SMagenda - Base de Conhecimento (RAG para Chatbot)

Gerado em 2026-01-16 21:48:25. Raiz: C:\Users\Admin\Desktop\SMagenda. PDF sobrescreve o arquivo existente.

Este PDF combina (1) uma base de conhecimento em linguagem natural (perguntas/respostas, fluxos e regras) e (2) um apêndice com dump de arquivos de texto do repositório para máxima cobertura. Segredos/tokens são redigidos por padrão.

---

1) Resumo do produto

- Plataforma de agendamento com link público (cliente agenda no navegador, sem app).
- Painel para o estabelecimento: agenda, serviços, clientes, funcionários, configurações e pagamento.
- Automação de mensagens: confirmação, lembrete e cancelamento via WhatsApp (quando configurado).

Fontes principais: smagenda/src/views/public/LandingPage.tsx, PublicBookingPage.tsx, AppShell.tsx, PagamentoPage.tsx, WhatsappSettingsPage.tsx, MensagensSettingsPage.tsx.

---

2) Rotas do site e do app

- Público: / (landing), /agendar/:slug (link público), /ajuda, /termos, /privacidade.
- Autenticação: /login, /cadastro, /esqueci-senha, /resetar-senha, /onboarding.
- Painel (usuário): /dashboard, /servicos, /clientes, /relatorios, /pagamento, /funcionarios, /configuracoes/*.

Fonte: smagenda/src/App.tsx.

---

3) Planos e preços (site)

BASIC **(basic)**

R$ 34,99/mês

Ideal para começar com 1 profissional

- Agendamentos 60 por mês
- 1 profissional incluído
- Lembretes automáticos via WhatsApp
- Até 3 serviços
- Página pública personalizável

PRO **(pro)**

R$ 59,99/mês

Até 6 profissionais (4 inclusos + até 2 adicionais)

- 4 profissionais incluídos
- Serviços ilimitados
- Logo e fotos de serviços
- Relatórios
- Bloqueios recorrentes

EMPRESA **(enterprise)**

R$ 98,99/mês

Até 10 profissionais

- Até 10 profissionais
- Multi-unidades
- Agendamentos ilimitados
- Serviços ilimitados
- Para mais profissionais, fale com o suporte

Fonte: smagenda/src/views/public/LandingPage.tsx.

---

4) Planos e cobrança (dentro do painel)

- Pagamento abre checkout em produção via Stripe: PIX (30 dias) ou cartão (assinatura).
- Gerenciamento/cancelamento: via Billing Portal (Stripe), quando disponível.

BASIC **(basic)**

R$ 34,99/mês

- Agendamentos 60 por mês
- 1 profissional incluído
- Lembretes automáticos via WhatsApp
- Até 3 serviços
- Página pública personalizável
- Suporte por email
- pro
- PRO
- R$ 59,99/mês
- Até 6 profissionais (4 inclusos + até 2 adicionais)
- 4 profissionais incluídos
- Serviços ilimitados
- Logo e fotos de serviços
- Relatórios
- Bloqueios recorrentes
- Suporte via WhatsApp
- enterprise
- EMPRESA
- R$ 98,99/mês
- Até 10 profissionais
- Até 10 profissionais
- Multi-unidades
- Agendamentos ilimitados
- Serviços ilimitados
- Logo e fotos de serviços
- Para mais profissionais, fale com o suporte
- setup_completo
- Setup Completo
- R$ 150 (uma vez)
- Configuramos tudo para você
- Cadastramos serviços, fotos, horários
- Testamos envios do WhatsApp apos conexão
- Treinamos o cliente em 15 minutos

Fonte: smagenda/src/views/app/PagamentoPage.tsx.

---

5) Fluxo do cliente: agendar pelo link público

- Passos: 1) Serviço → 2) Profissional (se habilitado por plano e existirem funcionários) → 2/3) Quando (data e horário) → Confirmar.

- Campos: Nome, WhatsApp e Email (opcional). Regras por tipo de negócio podem exigir Placa (lava-jato) e Endereço (faxina/diarista).
- Regras: antecedência mínima padrão 120 minutos (pode vir do serviço). Janela máxima geral até 1 ano (com regras por serviço).

Fonte: smagenda/src/views/public/PublicBookingPage.tsx.

---

6) Painel (navegação e permissões)

- Menu padrão do usuário: Agenda, Meus Serviços, Clientes, (PRO+) Relatórios, Pagamento, Funcionários, WhatsApp, Mensagens Automáticas, Página Pública, Ajuda.
- Funcionário: menu reduzido (Minha Agenda + Ajuda). Gerente (funcionário admin) vê opções conforme permissões.

Fonte: smagenda/src/components/layout/AppShell.tsx.

---

7) WhatsApp: conectar e testar

- Status exibe: Recurso (habilitado), Configuração (OK/Pendente), Conexão (Conectado/Desconectado/Conectando).
- Conectar: botão 'Conectar (QR Code)' e escaneamento do QR no WhatsApp.
- Teste: informar número com DDD e mensagem para validar envio em produção.

Fonte: smagenda/src/views/app/WhatsappSettingsPage.tsx.

---

8) Mensagens automáticas (confirmação, lembrete, cancelamento)

- Preferências: enviar confirmação ao confirmar agendamento; enviar lembrete automático; enviar cancelamento (quando habilitado).
- Templates: mensagens são editáveis e existem modelos prontos por segmento (ex.: lava-jato, barbearia, pilates, faxina).

Variáveis disponíveis

Variável	Significado
`{nome}`	Cliente
`{data}`	Data
`{hora}`	Hora
`{servico}`	Serviço
`{preco}`	Preço
`{cliente_endereco}`	Endereço cliente
`{profissional_nome}`	Profissional
`{telefone_profissional}`	Telefone
`{unidade_nome}`	Unidade
`{unidade_endereco}`	Endereço unidade
`{unidade_telefone}`	Telefone unidade
`{endereco}`	Endereço (fallback)
`{nome_negocio}`	Nome do negócio

Templates padrão (texto)

```
CONFIRMAÇÃO
Olá {nome}!\n\nSeu agendamento foi confirmado:\n📅 {data} às {hora}\n✂️ {servico}\n💰 {preco}\n\nLocal:
{endereco}\n\nNos vemos em breve!\n{nome_negocio}

LEMBRETE
Oi {nome}!\n\nLembrete: você tem agendamento em {data} às {hora}.\n\nSe não puder comparecer, me
avise!\n{telefone_profissional}

CANCELAMENTO
Olá {nome}!\n\nSeu agendamento foi cancelado:\n📅 {data} às {hora}\n✂️ {servico}\n\nSe quiser remarcar, é só me
chamar.\n{nome_negocio}
```

Fonte: smagenda/src/views/app/MensagensSettingsPage.tsx.

---

9) Ajuda, suporte e documentos

- Email de suporte (default): suporte@smagenda.com (pode vir de env VITE_SUPORTE_EMAIL/VITE_SUPPORT_EMAIL).
- WhatsApp de suporte (default): (31) 9 7518-4428 (pode vir de env VITE_SUPORTE_WHATSAPP/VITE_SUPPORT_WHATSAPP_NUMBER).
- Página de ajuda contém FAQ e links para Termos/Privacidade.

Fonte: smagenda/src/views/public/AjudaPage.tsx.

---

10) FAQ do site (landing)

Não foi possível extrair as perguntas frequentes.

Fonte: smagenda/src/views/public/LandingPage.tsx.

11) Termos e privacidade (fontes)

- Termos: regras de uso e contato via canais informados no painel.
- Privacidade: descreve dados processados (agendamentos, dados técnicos), finalidades, compartilhamento com provedores e direitos LGPD.

Fontes: smagenda/src/views/public/TermosPage.tsx e PrivacidadePage.tsx.

---

12) Solução de problemas (mensagens do próprio sistema)

- Pagamento: 'A função payments não está publicada no Supabase' → fazer deploy da Edge Function payments.
- Pagamento: 'A Edge Function está exigindo JWT no gateway' → deploy com verify_jwt=false.
- WhatsApp: 'Cloudflare Tunnel indisponível' → verificar túnel/URL pública e Evolution API.
- Supabase: 'Sessão expirada' ou 'JWT pertence a outro projeto' → sair e entrar novamente.

Fontes: smagenda/src/views/app/PagamentoPage.tsx e WhatsappSettingsPage.tsx.

Apêndice A) Dump do repositório (arquivos de texto)

Total de arquivos incluídos no apêndice: **109**. Este apêndice é para cobertura total no RAG.

Índice do apêndice

- prospector/vite.config.ts
- ROTEIRO_VENDAS.md
- scheduling-system-doc (1).md
- scheduling-system-doc.md
- smagenda/docker-compose.evolution.yml
- smagenda/eslint.config.js
- smagenda/index.html
- smagenda/package.json
- smagenda/postcss.config.js
- smagenda/public/_headers
- smagenda/public/_redirects
- smagenda/public/sw.js
- smagenda/public/vite.svg
- smagenda/README.md
- smagenda/src/App.tsx
- smagenda/src/assets/react.svg
- smagenda/src/components/layout/AdminShell.tsx
- smagenda/src/components/layout/AppShell.tsx
- smagenda/src/components/ui/Badge.tsx
- smagenda/src/components/ui/Button.tsx
- smagenda/src/components/ui/Card.tsx
- smagenda/src/components/ui/Input.tsx
- smagenda/src/components/ui/TutorialOverlay.tsx
- smagenda/src/index.css
- smagenda/src/lib/dates.ts
- smagenda/src/lib/env.ts
- smagenda/src/lib/slug.ts
- smagenda/src/lib/supabase.ts
- smagenda/src/main.tsx
- smagenda/src/state/auth/AuthContext.ts
- smagenda/src/state/auth/AuthProvider.tsx
- smagenda/src/state/auth/RequireAuth.tsx
- smagenda/src/state/auth/types.ts
- smagenda/src/state/auth/useAuth.ts
- smagenda/src/views/admin/AdminBootstrapPage.tsx
- smagenda/src/views/admin/AdminClienteDetalhesPage.tsx
- smagenda/src/views/admin/AdminClientesPage.tsx
- smagenda/src/views/admin/AdminConfiguracoesPage.tsx
- smagenda/src/views/admin/AdminDashboardPage.tsx
- smagenda/src/views/admin/AdminLoginPage.tsx
- smagenda/src/views/admin/AdminLogsPage.tsx
- smagenda/src/views/admin/AdminWhatsappAvisosPage.tsx
- smagenda/src/views/app/ClienteDetalhesPage.tsx
- smagenda/src/views/app/ClientesPage.tsx
- smagenda/src/views/app/DashboardPage.tsx
- smagenda/src/views/app/FuncionarioAgendaPage.tsx
- smagenda/src/views/app/FuncionariosPage.tsx
- smagenda/src/views/app/MensagensSettingsPage.tsx
- smagenda/src/views/app/PagamentoPage.tsx
- smagenda/src/views/app/PaginaPublicaSettingsPage.tsx
- smagenda/src/views/app/RelatoriosPage.tsx
- smagenda/src/views/app/ServicosPage.tsx
- smagenda/src/views/app/WhatsappSettingsPage.tsx
- smagenda/src/views/auth/CadastroPage.tsx

- smagenda/src/views/auth/ForgotPasswordPage.tsx
- smagenda/src/views/auth/LoginPage.tsx
- smagenda/src/views/auth/OnboardingPage.tsx
- smagenda/src/views/auth/ResetPasswordPage.tsx
- smagenda/src/views/public/AjudaPage.tsx
- smagenda/src/views/public/LandingPage.tsx
- smagenda/src/views/public/PrivacidadePage.tsx
- smagenda/src/views/public/PublicBookingPage.tsx
- smagenda/src/views/public/TermosPage.tsx
- smagenda/supabase/functions/admin-create-funcionario/config.toml
- smagenda/supabase/functions/admin-create-funcionario/index.ts
- smagenda/supabase/functions/create-funcionario/config.toml
- smagenda/supabase/functions/create-funcionario/index.ts
- smagenda/supabase/functions/payments/config.toml
- smagenda/supabase/functions/payments/index.ts
- smagenda/supabase/functions/resend-domain/config.toml
- smagenda/supabase/functions/resend-domain/index.ts
- smagenda/supabase/functions/whatsapp-lembretes/config.toml
- smagenda/supabase/functions/whatsapp-lembretes/index.ts
- smagenda/supabase/functions/whatsapp/config.toml
- smagenda/supabase/functions/whatsapp/index.ts
- smagenda/tailwind.config.js
- smagenda/tsconfig.app.json
- smagenda/tsconfig.json
- smagenda/tsconfig.node.json
- smagenda/vercel.json
- smagenda/vite.config.ts

checklist_sistema

```
2026-01-08
- Link público: endereço obrigatório para tipo_negocio=faxina e envio via p_extras
- Edge Function payments: aceita cliente_endereco e persiste em p_extras.endereco
- RPC public_create_agendamento_publico: suporte a p_extras (jsonb) e coluna agendamentos.extras
- Página pública (config): tipo_negocio fica bloqueado após primeira definição não-vazia
- Agenda (usuário/funcionário): exibe extras.endereco quando presente no agendamento
- WhatsApp: variável {cliente_endereco} disponível nos templates e preenchida por agendamentos.extras.endereco
(prioridade sobre unidade/endereço do negócio)
- Link público: regras por serviço (buffers/antecedência/janela) aplicadas em slots e criação
- RPC public_get_ocupacoes: considera buffer_antes_min/buffer_depois_min por serviço
- Página pública: maxDays/minLeadMinutes exibidos conforme regras do serviço selecionado
- Página pública: email opcional (extras.email) para fallback de confirmação
- Cliente (detalhes): edição de extras (endereco/placa/email) por agendamento
- WhatsApp (confirmação): fallback real por e-mail via Resend quando necessário
- Serviços: suporte a dia_inteiro e capacidade diária de diárias (1 ou 2)
- Link público: calendário responsivo para selecionar dia em serviços dia_inteiro
- RPC public_get_slots_publicos: dia_inteiro usa capacidade_dia_inteiro e bloqueios reais
- RPC public_create_agendamento_publico: dia_inteiro reserva expediente completo e respeita capacidade
- Lint: removido setState em useEffect (capacidadeDiaInteiro e mês do calendário)
- Página pública: hora de dia_inteiro é derivada de diaInteiroDisponibilidade[data]

2026-01-11
- Funcionários: fallback quando coluna capacidade_dia_inteiro não existe (carrega sem ela e exibe SQL para corrigir)
- Admin (configurações): SQL agora cria constraint para capacidade_dia_inteiro (1 ou 2)
- Cadastro: termos de uso e política de privacidade com aceite obrigatório e registro no Supabase
- Público: página /ajuda com contato, FAQ e links para termos/privacidade
- Painel: link “Ajuda” no menu lateral (usuário e funcionário)
- Ajuda: quando autenticado, abre dentro do painel e volta ao dashboard/agenda
- Ajuda: fallback real de suporte (telefone/WhatsApp e email) com links tel/mailto
- Ajuda: WhatsApp mostra número em negrito e email com prefixo "Email:"
- Ajuda: WhatsApp exibido como "WhatsApp: (31) 9 7518-4428"
- Link público: RPC via REST sanitiza VITE_SUPABASE_URL e VITE_SUPABASE_ANON_KEY (remove aspas e barras finais)
- Link público: fallback de assinatura agora remove p_extras quando Supabase está com SQL antigo
- Link público: erro de slots diferencia ausência de public_get_ocupacoes e sugere SQL de ocupações/bloqueios
- Link público: erro de slots/criação em rotas com unidade sugere também SQL de Multi-unidades (EMPRESA)
- Link público: detecção de RPC ausente ficou mais precisa (evita falso-positivo em permission denied)
- Admin (configurações): SQL do link público inclui pgcrypto e NOTIFY pgrst para refresh imediato
- Admin (configurações): SQL do link público remove overload de RPCs e padroniza assinaturas com parâmetros default
- Link público: public_get_usuario_publico agora aceita p_unidade_slug e evita erro de "best candidate function"
- Link público: RPCs públicas agora aceitam p_unidade_id default e salvam agendamentos com unidade_id quando disponível
- Público: ajuste de lint no PublicBookingPage (remove helper não utilizado)
- Link público: hotfix SQL remove overloads de public_create_agendamento_publico (evita PGRST203)
- Clientes: lista e detalhes destacam quando um telefone tem múltiplos nomes
- Cliente (detalhes): ação "Ocultar cliente" anonimiza telefone/nome em todos os agendamentos do número
- Cliente (detalhes): ocultar cliente usa telefone vazio e nome "Cliente oculto" (NOT NULL)
- Trial: período padrão reduzido de 15 para 7 dias (data_vencimento)
- Funcionários: limite agora respeita o plano (enterprise ilimitado; pro 4–6; basic/free 1)
- Edge Functions: create-funcionario/admin-create-funcionario validam limite pelo plano
- Deploy: Edge Functions create-funcionario (v17) e admin-create-funcionario (v1) em ttecpztkoaxwttludgfl
- Funcionários: permissões agora incluem Atendente (além de Funcionário e Gerente)
- Funcionários: rótulo de "Admin" renomeado para "Gerente" (permissao continua 'admin')
- Link público: public_get_funcionarios_publicos filtra para exibir apenas permissao='funcionario'
- Link público: filtro agora exige permissao='funcionario' (não mostra null/gerente/atendente)
- Deploy: Edge Functions create-funcionario (v18) e admin-create-funcionario (v2) em ttecpztkoaxwttludgfl
- Dev (Windows): npm ci estava falhando por lock em binário do Rollup; removido arquivo travado e reinstaladas
dependências
- TypeScript: tsconfig.app inclui tipos de React/ReactDOM e composite para remover erros de JSX no editor
- Funcionários: SQL para atualizar constraint funcionarios_permissao_check e permitir permissao='atendente'
- Agenda (painel): filtro de profissionais mostra apenas permissao='funcionario'
- Agenda (funcionário): gerente/atendente com "Ver agenda" podem ver agenda completa com filtro por profissional
- RLS: gerente/atendente com pode_ver_agenda podem listar funcionarios/agendamentos/bloqueios do master
- UI: campos de senha agora têm botão Ver/Ocultar
- Agenda (funcionário): removida edição do próprio horário (apenas gerente ajusta no painel)
- Agenda (atendente): atendente não visualiza o card "Resumo do Dia"
- Admin (configurações): SQL bloqueia RPC funcionario_update_horarios (não permite salvar horário por funcionário)
- Agenda (atendente): agenda completa usa horários do profissional filtrado (slots corretos)
- Agenda (atendente): erros de permissão/RLS exibem mensagem detalhada com ação recomendada
- Agenda (atendente): modo Lista para visualizar agendamentos e bloqueios por período
- Agenda (atendente): vazio agora mostra período, profissional e filtros ativos
- Agenda (painel): preferência Grade/Lista fica salva por usuário no navegador
```

```
- Agenda (atendente): exibe erro quando funcionarios.usuario_master_id está vazio (não carrega agenda)
- Supabase RLS: corrigida recursão infinita em policy de funcionarios usando função security definer
- Gerente: Configurações (Mensagens automáticas / Página pública) agora usam usuário master
- Onboarding: gerente usa perfil do usuário master (com estado Carregando…)
- Gerente: acesso agora limitado às 7 permissões (agenda/criar/cancelar/bloquear/financeiro/serviços/clientes)
- Menu: gerente só vê páginas permitidas pelas permissões marcadas
- Painel (agenda): bloquear horário, confirmar/no-show/cancelar e preços respeitam permissões do gerente
- Configurações: Pagamento/Funcionários/WhatsApp/Mensagens/Página pública restritos ao usuário master
- Funcionários: capacidade de diárias por dia só aparece para tipo_negocio=faxina/diarista
- Funcionários: ícone de agendamentos usa tesoura só em barbearia (senão calendário)
- WhatsApp: botão “Conectar (código)” agora prioriza código de pareamento (não QR)
- WhatsApp: UI não exibe QRCode quando o usuário escolhe “Conectar (código)”
- WhatsApp: Edge Function tenta criar/conectar com qrcode=false no modo código
- WhatsApp: ocultado o modo “Conectar (código)” (mantido apenas QR Code)
 - Mensagens automáticas: UI mobile aprimorada (padding responsivo, botões full-width no celular, melhor hierarquia)
 - Mensagens automáticas: adicionados modelos prontos para lava_jatos/barbearia/manicure/pilates/faxina
 - Mensagens automáticas: ao selecionar modelo pronto, aplica confirmação e lembrete automaticamente
 - Página pública (config): removida validação visual do link público (tela mais limpa)
 - Página pública (config): slug duplicado exibe “Nome já está em uso. Escolha outro para o seu link público.”
 - Página pública (config): Unidades ativo apenas no plano enterprise (carrega/permite CRUD só nesse plano)
 - Página pública (config): Aparência permite cores em qualquer plano
 - Página pública (config): Logo e imagem de fundo liberados apenas para planos pro/enterprise (com bloqueio de UI e no
salvar)
 - Página pública (config): prévia de aparência agora mostra serviços reais do usuário (Supabase)

2026-01-11 (validação)
- Build (tsc -b + vite build): OK
- Lint (eslint .): OK
- Smoke-test em preview (vite preview): OK (home abre sem erros no browser)
- Varredura por mocks/simulações no front: não encontrado (mock/fake/stub/simula)

2026-01-11 (GitHub)
- Upload realizado para https://github.com/Joecabulo/smagenda (branch main)
- Commit enviado: b273437

2026-01-11 (Cloudflare)
- Hotfix: adicionado package.json na raiz para o Cloudflare Pages encontrar npm scripts
- Build no Cloudflare: "npm run build" (root) gerando artefatos em "dist" (raiz)
- Frontend: Vite outDir ajustado para "../dist" (gera dist na raiz do repositório)
 - Repo: adicionado .gitignore na raiz para ignorar dist/ e logs

2026-01-11 (site)
- Landing page pública adicionada na rota / com CTA para cadastro/login
- Landing page: rodapé usa nome de marca configurável (© {ano} {brand})
- Landing page: seção de diferenciais para posicionamento/exclusividade
- Landing page: botão Copiar usa domínio real (window.location.origin)
- Branding: suporte a VITE_PUBLIC_PRODUCT_NAME, VITE_PUBLIC_COMPANY_NAME e VITE_PUBLIC_COMPANY_LOGO_URL (com fallback)
- Landing page: header exibe empresa (logo/nome) + nome do sistema
- Landing page: seção Para quem é e FAQ
- Site: favicon atualizado para ícone SingleMotion (substitui vite.svg)
- Site: favicon atualizado para usar smagenda/public/favicon.png (logo SingleMotion)

2026-01-12
- Landing page: seção Planos com preços reais (BASIC/PRO/EMPRESA) e CTA para cadastro
- Landing page: destaque de pré-venda com validade até 08/02/2026
- Landing page: CTA “Falar no WhatsApp” usa número real via env (VITE_SUPORTE_WHATSAPP / VITE_SUPPORT_WHATSAPP*)
- Landing page: bloco de confiança (checkout Stripe, cancelamento, suporte) e FAQ com cobrança/comparação de planos
- Validação: lint (eslint .) e build (tsc -b + vite build) OK
- Plano EMPRESA: limite definido para até 10 profissionais (UI + Edge Functions + pagamentos)
- Pagamento: botão “Gerenciar / Cancelar” abre Stripe Billing Portal (assinaturas)
- Auth: UsuarioProfile passa a carregar stripe_customer_id do Supabase
- Validação: lint (eslint .) OK; build (tsc -b + vite build) OK
- Pagamento: erro invalid_action agora orienta deploy da Edge Function payments
- Deploy: Edge Function payments (v16) em ttecpztkoaxwttludgfl
- Deploy: Edge Function payments (v17) com --no-verify-jwt em ttecpztkoaxwttludgfl
- Pagamento: Billing Portal agora retorna mensagem real do Stripe em erros
- Pagamento: suporta STRIPE_BILLING_PORTAL_CONFIGURATION (opcional)
- Deploy: Edge Function payments (v18) em ttecpztkoaxwttludgfl
- Pagamento: Billing Portal recria stripe_customer_id quando cliente não existe no Stripe
- Deploy: Edge Function payments (v19) em ttecpztkoaxwttludgfl
- Pagamento: PIX checkout resolve Price one-time real por plano (STRIPE_PRICE_{PLANO}_PIX ou fallback Stripe API)
- Deploy: Edge Function payments (v20) em ttecpztkoaxwttludgfl
```

```
- Roteiro de Vendas: adicionadas situações de venda consultiva (ex.: estúdio de yoga), mensagens prontas e objeções

2026-01-12 (prospector)
- Novo app separado "prospector" para importar estabelecimentos do Google Maps (Places API) por rua/cidade e segmentação
- Prospector: ações por estabelecimento (confirmado/aprovado/recusou/sem resposta/visitado) + contato (nome/número) +
exportação JSON/CSV
- Prospector: backend proxy (Express) para chamadas reais ao Google Places (exige GOOGLE_MAPS_API_KEY no server/.env)
- Prospector: UI redesenhada (tema escuro, painel de detalhes, filtros/ordenação, copiar/WhatsApp/tel, multi-ruas)
- Prospector: PWA instalável no Android (manifest + service worker + botão Instalar)
- Prospecção: nova tela /prospeccao para gerenciar estabelecimentos (status, contato, observações)
- Prospecção: Edge Function prospeccao importa estabelecimentos reais do Google Maps/Places por rua/termos
- Prospecção: SQL cria tabela public.prospeccao_estabelecimentos com RLS e índices
- Prospecção: correção de lint na ProspeccaoPage (useCallback e remoção de variável não usada)
- Prospector: botão "Importar Itabirito (cidade inteira)" via Places API real (proxy /api)
- Prospector: ordenação "Rua e número" com separação visual por rua
- Prospector: segmentos agora são persistidos/mesclados durante upsert/importações
- Prospector: api.ts melhora erro quando /api está offline (proxy 500 vazio)
- Prospector-server: carrega GOOGLE_MAPS_API_KEY via server/.env (sem dependências)
- Prospector (dev): iniciado prospector/server (8787) e vite dev (5173) para validar proxy /api
- Prospector: pré-check /api/health bloqueia importação sem GOOGLE_MAPS_API_KEY
- Prospector-server: criado prospector/server/.env placeholder (sem segredo)
- Prospector: preset "Todos (por agendamento)" agrega segmentos e importa em uma passada
- Prospector: barra de progresso com etapa atual durante importações

2026-01-12 (SMagenda)
- Tema alternativo “Prospector” implementado como flag real por usuário (usuarios.tema_prospector_habilitado)
- Super Admin: toggle em /admin/whatsapp para habilitar/desabilitar Tema Prospector em clientes selecionados
- Admin (configurações): novo bloco SQL “SQL do Tema Prospector (habilitação por cliente)” com trigger bloqueando update
fora do super admin
- AppShell aplica classe theme-prospector baseado no usuário logado (ou master do gerente), sem alterar tema padrão
- Build: corrigido erro TS5076 na ProspeccaoPage (precedência de ?? e ||)
- Lint: ajustado useEffect da ProspeccaoPage para evitar setState-in-effect
- Tema Prospector: dropdowns/inputs agora usam fundo escuro e texto claro (options legíveis)
- Super Admin: habilitar/desabilitar Tema Prospector movido para /admin/clientes (lista geral)
- Tema Prospector: Auth refresh automático em foco/visibilidade para refletir mudanças de flag
- PWA: SMagenda instalável (manifest + service worker + registro em produção)
- Prospecção: removida do SMagenda (tela/rota/menu/SQL/Edge Function)
- Validação: lint (eslint .) e build (tsc -b + vite build) OK após remoção

2026-01-14 (Prospector deploy)
- Prospector publicado como app separado no caminho /142555787po (Vite base + PWA ajustados)
- Deploy: build da raiz agora gera SMagenda (/) + Prospector (/142555787po) no mesmo dist
- Cloudflare Pages Functions: endpoints /api/* para Google Places/Geocode (sem servidor local)
- Cloudflare Pages Functions: adicionado /api/health para validar hasKey em produção
- Cloudflare redirects: fallback SPA para /142555787po e para /
- Prospector: importação padrão agora é por cidade + estado (UF), cobrindo a cidade inteira
- Prospector: raio de busca derivado do viewport do Geocode (fallback 14km) para resultado real
- Prospector: botão "Excluir tudo" para limpar importes salvos no dispositivo (localStorage)
- Segurança: removido prospector/server/.env do Git (chaves não devem ficar no repo)
- Prospector PWA: service worker não faz cache de /api/* (evita travar hasKey/erros antigos)

2026-01-14 (SMagenda)
- Página pública (config): removido campo de slug; URL pública somente leitura; slug gerado no salvar quando vazio
- Agenda: agendamento público inicia como "pendente"; ao confirmar vira "confirmado" e botão Confirmar some
- Agenda: filtro ganhou campo de Data (type=date) para pular direto para um dia
- Agenda: ao cancelar agendamento, envia aviso real via WhatsApp (Edge Function) e mostra feedback
- Mensagens automáticas: adicionados campo/toggle e modelos prontos de cancelamento (com variáveis reais)
- Admin (configurações): SQL do WhatsApp (automação + config global) inclui enviar_cancelamento e mensagem_cancelamento
- Edge Function whatsapp: envio de cancelamento respeita enviar_cancelamento e mensagem_cancelamento (fallback seguro)
- Mensagens automáticas: botões de “Modelos prontos” agora ficam em grid e com tamanho uniforme
- Link público: janela padrão de agendamento aumentada para 30 dias (mantém override por serviço)
- Link público: seleção de data agora abre calendário popup e preenche a data escolhida

2026-01-15
- Agenda (funcionário): notificação em tempo real quando entrar novo agendamento (Supabase Realtime)
- Agenda (funcionário): botão “Ativar notificações” usa Notification API (fallback visual no topo)
- WhatsApp: Edge Function whatsapp-lembretes aceita action=funcionario_novo_agendamento e envia ao profissional
- Admin (configurações): SQL adiciona trigger para avisar profissional no WhatsApp ao criar/atribuir agendamento
- Funcionários: removido bloco duplicado de botões “Cancelar/Salvar” no formulário
- WhatsApp (app): erro 530 do Cloudflare Tunnel agora mostra mensagem curta com dica de correção
```

```
2026-01-15 (validação geral)
- Root: npm run lint OK (smagenda + prospector)
- Root: npm run build OK (smagenda + prospector)
- Link público: removido limite curto e liberado até 1 ano (considera ano bissexto)
- Link público: calendário carrega disponibilidade real apenas do mês visível
- Link público: pagamento com taxa usa hora efetiva também em serviços dia_inteiro
- Link público: calendário popup desabilita dias sem horários disponíveis (consulta RPC real)
- Link público: texto de ajuda abaixo da data atualizado para orientar troca de mês e disponibilidade
- Admin (configurações): SQL do link público agora libera janela de 365 dias nas RPCs (slots/criação)
- Link público: removida seleção automática inicial (serviço/profissional/data) — exige clique do cliente
- Link público: quando data fica inválida, limpa seleção em vez de escolher outra automaticamente
- Link público: botões Continuar/Confirmar com mais destaque e efeito visual no clique
- Link público: pop-up exibido após envio real do agendamento (RPC/pagamento)

2026-01-17
- Documento PDF completo para RAG gerado a partir do repositório (texto de arquivos e configurações)
- Arquivo: SMagenda_RAG_Completo.pdf (raiz do projeto)
- Regras: ignora node_modules/dist/build/.git e redige tokens/chaves com padrão de segredo/JWT
```

CHECKLIST_SISTEMA.md

```
# Checklist do Sistema (SMagenda)

## 0) Visão geral (tudo que existe no sistema)

- [ ] Frontend (SPA): Vite/React (`smagenda/package.json:6-11`).
- [ ] Backend (BaaS): Supabase (Auth + Postgres + Storage + Edge Functions).
- [ ] Integrações externas:
  - [ ] Stripe (assinatura no cartão + pagamento único PIX + taxa de agendamento em standby) (`smagenda/supabase/functions/payments/index.ts:293-408`, `smagenda/supabase/functions/payments/index.ts:883-1105`).
  - [ ] Evolution API (WhatsApp via QR/pareamento) (`smagenda/supabase/functions/whatsapp/index.ts`).
  - [ ] Resend (validação de domínio e envio de teste) (`smagenda/supabase/functions/resend-domain/index.ts:145`).
- [ ] Deploy do frontend:
  - [ ] Vercel com rewrite de SPA (`smagenda/vercel.json:1-3`).
  - [ ] Headers de cache para assets (`smagenda/public/_headers:1-8`).

## Documentação de produto

- [ ] Documento multi-modal (setores) e roadmap: `DOCUMENTACAO_MULTI_MODAL.md`.

## 1) Repositório e comandos

- [ ] Git: repositório inicializado na raiz e branch `main` pronta para push.
- [ ] Rodar local: `npm run dev` (Vite) (`smagenda/package.json:7-11`).
- [ ] Validar qualidade: `npm run lint` (ESLint) (`smagenda/package.json:8-10`).
- [ ] Validar build: `npm run build` (TypeScript + Vite) (`smagenda/package.json:8-10`).
- [ ] Preview do build: `npm run preview` (`smagenda/package.json:10-10`).

## 2) Frontend (Vite/React)

### 2.1) Variáveis de ambiente (dados reais)

- [ ] Definir `VITE_SUPABASE_URL` e `VITE_SUPABASE_ANON_KEY` (obrigatórias) (`smagenda/src/lib/supabase.ts:4-15`).
- [ ] Confirmar que, sem essas envs, o app bloqueia e mostra tela de “Configuração necessária” (`smagenda/src/main.tsx:11-44`).
- [ ] (Opcional) Definir `VITE_SUPPORT_WHATSAPP_NUMBER` para botão “Suporte WhatsApp” (apenas PRO/TEAM/ENTERPRISE) (`smagenda/src/components/layout/AppShell.tsx:13-58`).
- [ ] Confirmar que o JWT pertence ao mesmo projeto do Supabase (mismatch) (`smagenda/src/lib/supabase.ts:34-41`).
- [ ] (Standby) Definir `VITE_ENABLE_TAXA_AGENDAMENTO=1` para exibir/ativar a taxa de agendamento (`smagenda/src/views/app/ServicosPage.tsx:51-53`, `smagenda/src/views/public/PublicBookingPage.tsx:330-366`).

### 2.2) Rotas e papéis (RBAC)

- [ ] Conferir rotas públicas de agendamento:
  - [ ] `/agendar/:slug` (página pública) (`smagenda/src/App.tsx:34-36`).
  - [ ] `/agendar/:slug/:unidadeSlug` (multi-unidades) (`smagenda/src/App.tsx:34-36`).
- [ ] Conferir rotas de autenticação:
  - [ ] `/login`, `/cadastro` (`smagenda/src/App.tsx:37-40`).
  - [ ] `/esqueci-senha`, `/resetar-senha` (`smagenda/src/App.tsx:38-40`).
  - [ ] `/onboarding` (apenas `usuario`) (`smagenda/src/App.tsx:41-48`).
- [ ] Conferir rotas do app (apenas `usuario`):
  - [ ] `/dashboard`, `/servicos`, `/clientes`, `/clientes/:telefone`, `/relatorios`, `/pagamento`, `/funcionarios` (`smagenda/src/App.tsx:50-105`).
  - [ ] `/configuracoes/whatsapp`, `/configuracoes/mensagens`, `/configuracoes/pagina-publica` (`smagenda/src/App.tsx:106-129`).
- [ ] Conferir rota de funcionário (apenas `funcionario`):
  - [ ] `/funcionario/agenda` (`smagenda/src/App.tsx:130-137`).
- [ ] Conferir rotas do Super Admin (apenas `super_admin`):
  - [ ] `/admin/login`, `/admin/dashboard`, `/admin/clientes`, `/admin/clientes/:id`, `/admin/whatsapp`, `/admin/configuracoes`, `/admin/logs` (`smagenda/src/App.tsx:140-188`).
- [ ] Confirmar que `/admin/bootstrap` só existe em DEV (`smagenda/src/App.tsx:139-140`).

### 2.3) Autenticação, perfis e impersonação

- [ ] Perfis suportados: `usuario`, `funcionario`, `super_admin` (`smagenda/src/state/auth/types.ts:55-58`).
- [ ] Resolução do perfil:
  - [ ] Tenta `super_admin`, depois `funcionarios`, depois `usuarios` (`smagenda/src/state/auth/AuthProvider.tsx:88-116`).
  - [ ] Se não existir `usuarios`, chama RPC `ensure_usuario_profile` (`smagenda/src/state/auth/AuthProvider.tsx:117-125`).
- [ ] Impersonação (Super Admin “virar cliente”) salva em `localStorage` (`smagenda/src/state/auth/AuthProvider.tsx:135-
```

```
205`).
- [ ] Bloqueio por cobrança (status/trial vencido) redireciona para `/pagamento`
(`smagenda/src/state/auth/RequireAuth.tsx:101-174`).

### 2.4) Onboarding do cliente

- [ ] Etapa 1: horário/dias/intervalo + logo (Storage bucket `logos`) (`smagenda/src/views/auth/OnboardingPage.tsx:72-
199`).
- [ ] Etapa 2: criação do primeiro serviço (`smagenda/src/views/auth/OnboardingPage.tsx:266-276`).
- [ ] Link público gerado: `${origin}/agendar/${slug}` (`smagenda/src/views/auth/OnboardingPage.tsx:44`).

### 2.5) Agenda (Dashboard)

- [ ] Criar/editar/cancelar agendamentos e bloqueios (UI principal) (`smagenda/src/views/app/DashboardPage.tsx`).
- [ ] Redirecionar checkout para `/pagamento` quando query tiver `checkout=success|cancel`
(`smagenda/src/views/app/DashboardPage.tsx:264-265`).

### 2.5.1) Agenda do funcionário

- [ ] Funcionário pode ajustar o próprio horário (início/fim/dias/intervalo) e salvar via RPC
`funcionario_update_horarios` (`smagenda/src/views/app/FuncionarioAgendaPage.tsx:793-824`).
- [ ] Card “Meu horário de funcionamento” fica no topo da tela do funcionário
(`smagenda/src/views/app/FuncionarioAgendaPage.tsx:834-1000`).
- [ ] Bloqueio de horários do funcionário continua separado do ajuste de horário (UI e SQL)
(`smagenda/src/views/app/FuncionarioAgendaPage.tsx`).

### 2.6) Serviços

- [ ] CRUD de serviços (preço/duração/cor) (`smagenda/src/views/app/ServicosPage.tsx`).
- [ ] (Standby) Taxa por serviço: UI/CRUD e leitura são liberadas só com `VITE_ENABLE_TAXA_AGENDAMENTO=1` (inclui
fallback para não exigir coluna quando desligado) (`smagenda/src/views/app/ServicosPage.tsx:51-140`,
`smagenda/src/views/app/ServicosPage.tsx:212-230`).
- [ ] (PRO+) Foto de serviço via Storage bucket `servicos` (políticas e trigger no SQL)
(`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).

### 2.7) Funcionários

- [ ] CRUD de funcionários no app (`smagenda/src/views/app/FuncionariosPage.tsx`).
- [ ] Criação via Edge Function `create-funcionario` (`smagenda/src/views/app/FuncionariosPage.tsx:310`,
`smagenda/supabase/functions/create-funcionario/index.ts:40-42`).
- [ ] Permissões por funcionário (flags no perfil) (`smagenda/src/state/auth/types.ts:33-45`).

### 2.8) Clientes e relatórios

- [ ] Listagem e resumo por telefone (`smagenda/src/views/app/ClientesPage.tsx`).
- [ ] Detalhes do cliente por `:telefone` (`smagenda/src/views/app/ClienteDetalhesPage.tsx`).
- [ ] Relatórios (export/financeiro) (`smagenda/src/views/app/RelatoriosPage.tsx`).

### 2.9) Configurações do cliente

- [ ] WhatsApp (conectar, status, desconectar, teste) (`smagenda/src/views/app/WhatsappSettingsPage.tsx`).
- [ ] Mensagens automáticas (templates e personalização) (`smagenda/src/views/app/MensagensSettingsPage.tsx:1-55`).
- [ ] Página pública: slug, cores, fundo, logo, tipo de negócio
(`smagenda/src/views/app/PaginaPublicaSettingsPage.tsx:111-260`).

### 2.10) Página pública (agendamento)

- [ ] Carregar dados via RPCs públicas (link público) (`smagenda/src/views/public/PublicBookingPage.tsx:150-215`).
- [ ] Labels dinâmicos por `tipo_negocio` (`smagenda/src/views/public/PublicBookingPage.tsx:62-70`).
- [ ] Implementado labels e opções de `tipo_negocio` adicionais: manicure, pilates e faxina
(`smagenda/src/views/public/PublicBookingPage.tsx`, `smagenda/src/views/app/PaginaPublicaSettingsPage.tsx`).
- [ ] Lava-jato: exigir placa e embutir no nome do cliente (compatibilidade)
(`smagenda/src/views/public/PublicBookingPage.tsx:137-138`, `smagenda/src/views/public/PublicBookingPage.tsx:326-352`).
- [ ] (Standby) Cobrança de taxa no agendamento público (checkout + retorno `paid=1&session_id=...`) é habilitada por
flag e usa a Edge Function `payments` (`smagenda/src/views/public/PublicBookingPage.tsx:333-862`,
`smagenda/supabase/functions/payments/index.ts:883-1105`).

### 2.11) Super Admin (painel)

- [ ] Login Super Admin (`smagenda/src/views/admin/AdminLoginPage.tsx`).
- [ ] Dashboard (visão geral) (`smagenda/src/views/admin/AdminDashboardPage.tsx`).
- [ ] Lista de clientes + impersonação (`smagenda/src/views/admin/AdminClientesPage.tsx`).
```

```
- [ ] Detalhe do cliente + gerar checkout + criar funcionário
(`smagenda/src/views/admin/AdminClienteDetalhesPage.tsx:246-248`,
`smagenda/src/views/admin/AdminClienteDetalhesPage.tsx:478-481`).
- [ ] Avisos WhatsApp (broadcast) (`smagenda/src/views/admin/AdminWhatsappAvisosPage.tsx`).
- [ ] Configurações (SQL “fonte da verdade” + Resend DNS checker)
(`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).
- [ ] Logs de auditoria (`smagenda/src/views/admin/AdminLogsPage.tsx`).

## 3) Supabase (Auth + Banco + Storage + RPC + RLS)

### 3.1) Supabase Auth (contas reais)

- [ ] Configurar Site URL e Redirect URLs para o domínio real do app (reset de senha e links de confirmação).
- [ ] Confirmar cadastro/login e status de confirmação consultando `/auth/v1/settings` (uso no frontend)
(`smagenda/src/views/auth/CadastroPage.tsx:60-67`, `smagenda/src/views/auth/LoginPage.tsx:241-246`).

### 3.2) Schema/RLS: “fonte da verdade”

- [ ] Aplicar os blocos SQL exibidos em `/admin/configuracoes` (é o schema do projeto).
- [ ] Blocos relevantes (mínimo para rodar 100%):
  - [ ] SQL base (tabelas app + RLS + helpers como `is_super_admin`).
  - [ ] SQL do link público (listar + agendar) + grants para `anon`.
  - [ ] SQL do horário do funcionário: cria a função `public.funcionario_update_horarios(...)` e libera `execute` para
`authenticated` (`smagenda/src/views/admin/AdminConfiguracoesPage.tsx:2283-2366`).
  - [ ] (Standby) SQL da taxa de agendamento (créditos): adiciona `servicos.taxa_agendamento`, cria
`taxa_agendamento_pagamentos` e atualiza RPCs públicas (`smagenda/src/views/admin/AdminConfiguracoesPage.tsx:3378-3390`,
`smagenda/src/views/admin/AdminConfiguracoesPage.tsx:777-923`).
  - [ ] SQL do Storage (logos) (`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).
  - [ ] SQL do Storage (fotos de serviços) (`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).
  - [ ] SQL de fotos nos serviços (PRO+) (`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).
  - [ ] SQL WhatsApp (Super Admin / habilitação / automação / cron / triggers).
  - [ ] SQL Cobrança (cron diário + trigger de status).
  - [ ] SQL de Logs de Auditoria (`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).

### 3.3) Tabelas principais (mínimo esperado)

- [ ] `public.usuarios` (perfil do cliente/empresa, plano, cobrança, configs, WhatsApp).
- [ ] `public.funcionarios` (perfil e permissões do funcionário).
- [ ] `public.servicos` (catálogo do cliente).
- [ ] `public.agendamentos` (agenda, status, flags de confirmação/lembrete).
- [ ] `public.bloqueios` (bloqueios de horário).
- [ ] `public.super_admin` (config global, inclusive Evolution API)
(`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).
- [ ] `public.unidades` (multi-unidades ENTERPRISE) (`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).
- [ ] `public.audit_logs` (auditoria) (`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).

### 3.4) RPCs públicas (link público)

- [ ] `public_get_usuario_publico` (com `tipo_negocio` e unidade quando aplicável)
(`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).
- [ ] `public_get_servicos_publicos`, `public_get_funcionarios_publicos`, `public_get_slots_publicos`,
`public_create_agendamento_publico`.
- [ ] (Standby) `public_get_servicos_publicos` retorna `taxa_agendamento` quando o SQL da taxa foi aplicado
(`smagenda/src/views/admin/AdminConfiguracoesPage.tsx:897-923`).

### 3.5) Storage (dados reais)

- [ ] Bucket `logos` público com write restrito por `auth.uid()`
(`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).
- [ ] Bucket `servicos` público com write restrito por `auth.uid()`
(`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).

### 3.6) Multi-unidades (ENTERPRISE)

- [ ] Aplicar SQL de multi-unidades e validar rota `/agendar/:slug/:unidadeSlug` (`smagenda/src/App.tsx:34-36`).

## 4) Edge Functions (Supabase)

### 4.1) Regras gerais

- [ ] Deploy de todas as funções em `smagenda/supabase/functions/`.
- [ ] Confirmar `verify_jwt=false` nos `config.toml` (ex.: `smagenda/supabase/functions/payments/config.toml:1`).
```

```
- [ ] Definir secrets padrão (mínimo): `SUPABASE_URL`, `SUPABASE_ANON_KEY`, `SERVICE_ROLE_KEY`/`SUPABASE_SERVICE_ROLE_KEY`.

### 4.2) `payments` (Stripe)

- [ ] Secrets: `STRIPE_SECRET_KEY`, `STRIPE_WEBHOOK_SECRET`, `SITE_URL`/`APP_URL` (`smagenda/supabase/functions/payments/index.ts:293-310`, `smagenda/supabase/functions/payments/index.ts:598-600`).
- [ ] (Standby) Secret=[REDACTED]
- [ ] Webhook Stripe apontando para `https://<PROJECT_REF>.supabase.co/functions/v1/payments`.
- [ ] Eventos mínimos: `checkout.session.completed`, `checkout.session.async_payment_succeeded`, `invoice.payment_succeeded`, `invoice.payment_failed`, `customer.subscription.updated`, `customer.subscription.deleted` (`smagenda/supabase/functions/payments/index.ts:332-408`).
- [ ] PIX (plano 30 dias): configurar `STRIPE_PRICE_<PLANO>_PIX` (Prices one-time em BRL) (`smagenda/supabase/functions/payments/index.ts:640`).
- [ ] UI cliente: botões “PIX (30 dias)” e “Cartão (assinatura)” (`smagenda/src/views/app/PagamentoPage.tsx:456-459`).
- [ ] Plano PRO: 4 profissionais inclusos + até 2 adicionais (máximo 6); acima disso, usar EMPRESA.
- [ ] Implementado PRO (4 inclusos, até 2 adicionais; máx 6): UI do cliente (`smagenda/src/views/app/PagamentoPage.tsx`) + UI do Super Admin (`smagenda/src/views/admin/AdminClienteDetalhesPage.tsx`) + validação/checkout na Edge Function (`smagenda/supabase/functions/payments/index.ts`).
- [ ] UI de planos: exibe aviso de pré-venda “válido até 08/02/2026” nos cards.
- [ ] UI Super Admin: gerar checkout PIX/cartão (`smagenda/src/views/admin/AdminClienteDetalhesPage.tsx:478-481`).

### 4.3) `whatsapp` (Evolution API)

- [ ] Secrets: `SUPABASE_URL`, `SUPABASE_ANON_KEY` e `SERVICE_ROLE_KEY` (necessária para config global) (`smagenda/supabase/functions/whatsapp/index.ts`).
- [ ] Config global (Super Admin): preencher `super_admin.whatsapp_api_url` e `super_admin.whatsapp_api_key` via SQL + UI.
- [ ] Validar que a URL da Evolution é pública (não localhost/IP privado) (`smagenda/supabase/functions/whatsapp/index.ts`).

### 4.4) `whatsapp-lembretes` (cron + cobrança)

- [ ] Secret=[REDACTED]
- [ ] Validar SQL do cron (pg_cron) e triggers de confirmação/lembrete/billing em `/admin/configuracoes`.

### 4.5) `resend-domain` (Email)

- [ ] Secret=[REDACTED]
- [ ] Validar domínio/DNS e enviar teste via checker no `/admin/configuracoes` (`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).

### 4.6) `create-funcionario` / `admin-create-funcionario`

- [ ] Secrets: `SUPABASE_URL`, `SUPABASE_ANON_KEY`, `SERVICE_ROLE_KEY` (`smagenda/supabase/functions/create-funcionario/index.ts:40-42`, `smagenda/supabase/functions/admin-create-funcionario/index.ts:36-38`).
- [ ] Confirmar chamada do app e do admin:
  - [ ] App chama `/functions/v1/create-funcionario` (`smagenda/src/views/app/FuncionariosPage.tsx:310`).
  - [ ] Admin chama `admin-create-funcionario` via helper `callFn` (`smagenda/src/views/admin/AdminClienteDetalhesPage.tsx:246`).

## 5) Cobrança e bloqueio de acesso

- [ ] Bloqueio no frontend por `status_pagamento` e por trial vencido (`smagenda/src/state/auth/RequireAuth.tsx:101-174`).
- [ ] Stripe webhook atualiza: `usuarios.status_pagamento`, `usuarios.data_vencimento`, `usuarios.plano` (`smagenda/supabase/functions/payments/index.ts:332-408`).

## 6) WhatsApp (Evolution API) — Infra (dados reais)

- [ ] Subir Evolution API com persistência de sessão (QR não pode “sumir”).
- [ ] Confirmar volumes e serviços (api + postgres + redis) (`smagenda/docker-compose.evolution.yml:4-50`).
- [ ] Configurar variáveis do Docker conforme `docker-compose.evolution.yml` (`smagenda/docker-compose.evolution.yml:11-45`).
- [ ] Teste obrigatório de reinício: conectar 1 WhatsApp, reiniciar containers/host, confirmar que volta conectado.

## 7) Observabilidade, operação e segurança

- [ ] Logs de auditoria ativos e visíveis em `/admin/logs` (SQL `audit_logs`) (`smagenda/src/views/admin/AdminConfiguracoesPage.tsx`).
- [ ] Separar ambientes (DEV/PROD): cada frontend aponta para seu Supabase e Stripe correspondentes.
- [ ] Gerenciar secrets apenas via Supabase/Vercel (nunca em código).
```

```
- [ ] Backups/retention do Postgres do Supabase (ou estratégia equivalente).
- [ ] Confirmar headers/cache no deploy (assets com cache longo; HTML sem cache) (`smagenda/public/_headers:1-8`).

### 7.1) Deploy (Cloudflare Pages)

- [ ] Build command: `npm run build` (na raiz do repositório).
- [ ] Build output directory: `dist` (na raiz do repositório).

## 8) “Não usar simulação em produção”

- [ ] Stripe em modo Live: `STRIPE_SECRET_KEY` e `STRIPE_WEBHOOK_SECRET` do Live + webhook Live.
- [ ] PIX real habilitado no Stripe e Prices PIX reais configurados.
- [ ] Evolution API real: URL pública acessível e API Key correta.
- [ ] Storage real: buckets/policies reais no projeto Supabase de produção.
- [ ] (Standby) Só ativar taxa de agendamento após: aplicar o SQL da taxa + configurar Stripe Live + definir
`VITE_ENABLE_TAXA_AGENDAMENTO=1` (frontend) e `ENABLE_BOOKING_FEE=1` (Edge Function)
(`smagenda/src/views/public/PublicBookingPage.tsx:333-862`, `smagenda/supabase/functions/payments/index.ts:883-1105`).
```

DOCUMENTACAO_MULTI_MODAL.md

```markdown
# Documentação Multi-Modal (SMagenda)

Este documento define o que o SMagenda precisa cobrir para ser um sistema de agendamento realmente “multi-modal” (multi-setor), atendendo desde profissionais autônomos (ex.: faxineira) até negócios com equipe e estrutura (ex.: estúdio de pilates, salão, barbearia, estética, manicure, etc.).

O objetivo é servir como “fonte da verdade” do produto: quais funcionalidades devem existir, como elas se conectam, e quais critérios validam a proposta de valor em produção (dados reais).

## 1) Princípios do produto

1. **Núcleo único + variações configuráveis**
   - Evitar “um sistema diferente para cada nicho”.
   - Preferir configurações por tipo de negócio (`tipo_negocio`), campos dinâmicos e regras por serviço/profissional.

2. **Experiência do cliente final (página pública) é prioridade**
   - A página pública é a principal alavanca de conversão.
   - Fluxo curto, rápido, com confirmações e no-show controlável.

3. **Operação do dono/secretaria é prioridade**
   - Ajuste de agenda, bloqueios, remarcação e comunicação não podem ser trabalhosos.

4. **Tudo validado com uso real**
   - Integrações reais (WhatsApp, pagamentos), métricas e logs.
   - Sem simulação como “feature final”; sandbox serve só para desenvolvimento.

## 2) Públicos-alvo (personas) e cenários

### 2.1) Autônomo (solo)
- Ex.: faxineira, manicure que atende em casa, barbeiro solo.
- Necessidades:
  - Agenda simples e rápida.
  - Página pública com link/WhatsApp.
  - Lembretes e confirmação.
  - Regras de disponibilidade e deslocamento (quando atendimento é externo).

### 2.2) Pequena equipe (2 a 6 profissionais)
- Ex.: barbearia, salão, estética.
- Necessidades:
  - Múltiplos profissionais.
  - Distribuição e/ou escolha do profissional.
  - Serviços com durações e preços diferentes.
  - Políticas de cancelamento/no-show.

### 2.3) Estúdio com recursos/capacidade (agenda “com restrições”)
- Ex.: pilates, funcional, estúdio com salas/equipamentos.
- Necessidades:
  - Agendamento por **turma/aula** ou por **capacidade**.
  - Recursos compartilhados: sala/equipamento com limite por horário.
  - Pacotes/mensalidades e controle de créditos/aulas.

## 3) Matriz de requisitos por setor (o que muda)

### 3.1) Itens comuns (todos os setores)
- Cliente consegue agendar sozinho por link público.
- Profissional/dono consegue criar/editar/cancelar e bloquear horários.
- Comunicação automática (mensagens de confirmação/lembrete) com opt-in.
- Relatórios mínimos (agenda, no-show, receita por período).

### 3.2) Itens que variam por setor (via configurações)
- **Campos do agendamento** (ex.: endereço, observações, placa, alergias, convênio, nível do aluno).
- **Regras de disponibilidade** (tempo de deslocamento, buffers, antecedência mínima).
- **Tipo de serviço** (individual, grupo, recorrente, pacote).
- **Capacidade/recurso** (sala/equipamento/vagas).

## 4) Funcionalidades “core” (MVP multi-modal)

### 4.1) Cadastros
1. **Serviços**
   - Nome, descrição, duração, preço, cor, ativo.
```

```
   - Regras por serviço:
     - Buffer antes/depois.
     - Antecedência mínima/máxima.
     - Permite escolher profissional ou auto-alocação.
     - Permite atendimento externo (quando aplicável).

2. **Profissionais (e permissões)**
   - Perfil do profissional, horário de trabalho, intervalos e dias.
   - Permissões: ver agenda, criar agendamentos, cancelar, ver financeiro, etc.

3. **Unidades/locais (quando houver)**
   - Multi-unidades para negócios maiores (e link público por unidade).

### 4.2) Agenda e operação
1. **Agenda do dono (dashboard)**
   - Visualização por dia/semana.
   - Criar/editar/cancelar agendamento.
   - Bloqueios recorrentes.

2. **Agenda do funcionário**
   - Visualização filtrada pelo próprio profissional.
   - Ações conforme permissões.

3. **Gestão de clientes**
   - Histórico por telefone.
   - Observações.

### 4.3) Página pública (conversão)
1. **Fluxo de agendamento**
   - Escolha de serviço → (profissional/unidade) → horário → dados do cliente → confirmação.
   - Labels por `tipo_negocio`.

2. **Regras de UX**
   - Mostrar só horários realmente disponíveis.
   - Carregamento rápido.
   - Confirmação clara e simples.

### 4.4) Mensageria (redução de no-show)
1. **Confirmação automática**
   - Mensagem de confirmação pós-agendamento.
   - Opção de enviar/ não enviar por configuração.

2. **Lembrete automático**
   - Configurar “horas antes”.
   - Templates por negócio.

3. **Canal mínimo recomendado**
   - WhatsApp (produção) como canal principal.
   - E-mail como fallback quando WhatsApp não estiver configurado.

### 4.5) Cobrança do plano (SaaS)
1. **Checkout**
   - PIX (30 dias) e cartão (assinatura).
   - Plano define limites (profissionais, agendamentos/mês etc.).

2. **Bloqueio por inadimplência**
   - Restringir acesso após vencimento/trial.

## 5) Funcionalidades “diferenciadoras” (para validar multi-modal de verdade)

### 5.1) Agendamento com pagamento/deposito (anti no-show)
- Opções por serviço:
  - Sem pagamento.
  - Depósito/sinal.
  - Pagamento integral.
- Política de cancelamento e reembolso.
- Registro do status do pagamento no agendamento.

### 5.2) No-show protection (cartão em arquivo / cobrança por falta)
- Reduz muito no-show em estética e serviços de maior ticket.
- Exige UX e comunicação claros.
```

```
### 5.3) Pacotes/mensalidades/créditos (essencial para pilates)
- Exemplos:
  - “8 aulas/mês” (expira no mês).
  - “10 sessões” (expira em 90 dias).
  - Mensalidade recorrente.
- Regras:
  - Consumo automático ao confirmar presença.
  - Reposição/estorno conforme política.

### 5.4) Aulas em grupo (capacidade)
- Criar “turmas” com:
  - Capacidade por horário.
  - Lista de alunos.
  - Lista de espera.
  - Check-in/presença.

### 5.5) Recursos (salas/equipamentos)
- Reservar recurso junto com o agendamento.
- Evitar overbooking de sala/equipamento.

### 5.6) Atendimento externo (faxina e domiciliar)
- Campos obrigatórios: endereço, referência.
- Regras:
  - Tempo mínimo entre atendimentos.
  - Zona de atendimento (quando aplicável).
  - Taxa de deslocamento opcional.

### 5.7) Personalização por segmento
- Templates de mensagens por setor.
- Campos do formulário público por setor.
- Labels no público por setor.

## 6) “Completo” para o público: o que costuma ser decisivo

### 6.1) Para barbearia/salão/manicure/estética
- Escolha do profissional e serviços com variações.
- Fotos/portfólio.
- Depósito/anti no-show.
- Confirmação/lembrete no WhatsApp.
- Remarcação simples.

### 6.2) Para faxina e serviços domiciliares
- Endereço + duração variável.
- Regras de deslocamento/buffer.
- Pagamento (sinal ou pós-serviço) e comunicação clara.

### 6.3) Para pilates/estúdio
- Turmas e capacidade.
- Mensalidades e créditos.
- Controle de presença.
- Recursos (salas/equipamentos).

## 7) Métricas de validação (para provar que multi-modal funciona)

### 7.1) Métricas do funil (página pública)
- Visitas no link público → seleção de serviço → seleção de horário → agendamento criado.
- Taxa de conversão por setor.
- Tempo médio para concluir agendamento.

### 7.2) Métricas de operação
- No-show rate.
- Remarcações e cancelamentos.
- Tempo de agenda ociosa.

### 7.3) Métricas financeiras
- Receita por período (quando houver pagamento integrado).
- Ticket médio por serviço.
- Receita recorrente (planos e mensalidades).

## 8) Roadmap recomendado (para validar rápido sem inflar complexidade)

### Fase 1 — Multi-modal mínimo (conversão + agenda + WhatsApp)
```

```
- Campos dinâmicos no agendamento por `tipo_negocio`.
- Templates de WhatsApp por setor.
- Regras de disponibilidade por serviço (buffers, antecedência).
- Relatórios mínimos.

### Fase 2 — No-show e monetização de fluxo
- Depósito/sinal por serviço.
- Políticas de cancelamento.
- Pagamento online opcional.

### Fase 3 — Estúdio (pilates) completo
- Turmas/capacidade + lista de espera.
- Pacotes/mensalidades/créditos.
- Recursos (salas/equipamentos).

## 9) Critérios de “pronto para produção” (dados reais)

1. WhatsApp funcionando com infraestrutura real (Evolution API) e logs.
2. Pagamentos funcionando em modo Live quando ativados.
3. Auditoria e métricas mínimas coletadas.
4. Fluxos críticos testados: agendar, cancelar, remarcar, lembrete, confirmação.
```

erro.md

```
import {createHotContext as __vite__createHotContext} from "/@vite/client";
import.meta.hot = __vite__createHotContext("/src/views/app/PagamentoPage.tsx");
import __vite__cjsImport0_react_jsxDevRuntime from "/node_modules/.vite/deps/react_jsx-dev-runtime.js?v=9875b0cf";
const jsxDEV = __vite__cjsImport0_react_jsxDevRuntime["jsxDEV"];
var _s = $RefreshSig$();
import __vite__cjsImport1_react from "/node_modules/.vite/deps/react.js?v=9875b0cf";
const useEffect = __vite__cjsImport1_react["useEffect"];
const useMemo = __vite__cjsImport1_react["useMemo"];
const useState = __vite__cjsImport1_react["useState"];
import {useLocation, useNavigate} from "/node_modules/.vite/deps/react-router-dom.js?v=9875b0cf";
import {AppShell} from "/src/components/layout/AppShell.tsx?t=1767394914655";
import {Badge} from "/src/components/ui/Badge.tsx";
import {Button} from "/src/components/ui/Button.tsx";
import {Card} from "/src/components/ui/Card.tsx";
import {Input} from "/src/components/ui/Input.tsx";
import {checkJwtProject, supabase, supabaseEnv} from "/src/lib/supabase.ts?t=1767388062616";
import {useAuth} from "/src/state/auth/useAuth.ts";
function safeJson(value) {
    try {
        return JSON.stringify(value);
    } catch {
        return null;
    }
}
async function callPaymentsFn(body) {
    if (!supabaseEnv.ok) {
        return {
            ok: false,
            status: 0,
            body: {
                error: "missing_supabase_env"
            }
        };
    }
    const supabaseUrl = String(supabaseEnv.values.VITE_SUPABASE_URL ?? "").trim().replace(/^['"`\s]+|['"`\s]+$/g,
"").replace(/\/+$/g, "");
    const supabaseAnonKey = String(supabaseEnv.values.VITE_SUPABASE_ANON_KEY ?? "").trim().replace(/^['"`\s]+|
['"`\s]+$/g, "");
    const fnUrl = `${supabaseUrl}/functions/v1/payments`;
    const tryRefresh = async () => {
        const {data: refreshed, error: refreshErr} = await supabase.auth.refreshSession();
        if (refreshErr)
            return null;
        return refreshed.session ?? null;
    }
    ;
    const {data: sessionData} = await supabase.auth.getSession();
    let session = sessionData.session;
    const now = Math.floor(Date.now() / 1e3);
    const expiresAt = session?.expires_at ?? null;
    if (session && (!expiresAt || expiresAt <= now + 60)) {
        const refreshed = await tryRefresh();
        if (refreshed)
            session = refreshed;
    }
    if (session) {
        const {error: userErr} = await supabase.auth.getUser();
        const userErrMsg = typeof userErr?.message === "string" ? userErr.message : "";
        if (userErr && /invalid\s+jwt/i.test(userErrMsg)) {
            const refreshed = await tryRefresh();
            if (refreshed)
                session = refreshed;
        }
    }
    const token = session?.access_token ?? null;
    if (!token) {
        return {
            ok: false,
            status: 401,
            body: {
```

```
                error: "session_expired"
            }
        };
    }
    const tokenProject = checkJwtProject(token, supabaseUrl);
    if (!tokenProject.ok) {
        await supabase.auth.signOut().catch( () => void 0);
        return {
            ok: false,
            status: 401,
            body: {
                error: "jwt_project_mismatch",
                iss: tokenProject.iss,
                expected: tokenProject.expectedPrefix
            }
        };
    }
    const callFetch = async (jwt) => {
        let res;
        try {
            res = await fetch(fnUrl, {
                method: "POST",
                headers: {
                    "Content-Type": "application/json",
                    apikey=[REDACTED]
                    Authorization: `Bearer ${jwt}`
                },
                body: JSON.stringify(body)
            });
        } catch (e) {
            const msg = e instanceof Error ? e.message : "Falha de rede";
            return {
                ok: false,
                status: 0,
                body: {
                    error: "network_error",
                    message: msg
                }
            };
        }
        const text = await res.text();
        let parsed = null;
        try {
            parsed = text ? JSON.parse(text) : null;
        } catch {
            parsed = text;
        }
        if (!res.ok && res.status === 404) {
            const raw = typeof parsed === "string" ? parsed : text;
            if (typeof raw === "string" && raw.includes("Requested function was not found")) {
                return {
                    ok: false,
                    status: 404,
                    body: {
                        error: "function_not_deployed",
                        message: "A função payments não está publicada no Supabase. Faça deploy da Edge Function
`payments` no seu projeto."
                    }
                };
            }
        }
        if (!res.ok && res.status === 401 && parsed && typeof parsed === "object" && parsed.message === "Invalid JWT" &&
parsed.code === 401) {
            return {
                ok: false,
                status: 401,
                body: {
                    error: "supabase_gateway_invalid_jwt"
                }
            };
        }
        if (!res.ok)
```

```
            console.error("payments error", {
                status: res.status,
                body: parsed,
                body_json: safeJson(parsed)
            });
        if (!res.ok)
            return {
                ok: false,
                status: res.status,
                body: parsed
            };
        return {
            ok: true,
            status: res.status,
            body: parsed
        };
    }
    ;
    const first = await callFetch(token);
    if (!first.ok && first.status === 401) {
        const refreshed = await tryRefresh();
        const nextToken = refreshed?.access_token ?? null;
        if (!nextToken) {
            await supabase.auth.signOut().catch( () => void 0);
            return {
                ok: false,
                status: 401,
                body: {
                    error: "invalid_jwt"
                }
            };
        }
        const nextProject = checkJwtProject(nextToken, supabaseUrl);
        if (!nextProject.ok) {
            await supabase.auth.signOut().catch( () => void 0);
            return {
                ok: false,
                status: 401,
                body: {
                    error: "jwt_project_mismatch",
                    iss: nextProject.iss,
                    expected: nextProject.expectedPrefix
                }
            };
        }
        return callFetch(nextToken);
    }
    return first;
}
async function createCheckoutPagamento(usuarioId, item, metodo, funcionariosTotal) {
    const payload = {
        action: "create_checkout",
        usuario_id: usuarioId,
        plano: item,
        metodo
    };
    if (typeof funcionariosTotal === "number" && Number.isFinite(funcionariosTotal))
        payload.funcionarios_total = funcionariosTotal;
    return callPaymentsFn(payload);
}
async function syncCheckoutSessionPagamento(sessionId, usuarioId) {
    const payload = {
        action: "sync_checkout_session",
        session_id: sessionId
    };
    if (usuarioId)
        payload.usuario_id = usuarioId;
    return callPaymentsFn(payload);
}
export function PagamentoPage() {
    _s();
    const {appPrincipal, refresh} = useAuth();
```

```
    const location = useLocation();
    const navigate = useNavigate();
    const usuario = appPrincipal?.kind === "usuario" ? appPrincipal.profile : null;
    const usuarioId = usuario?.id ?? null;
    const [checkoutNotice,setCheckoutNotice] = useState(null);
    const [userSelectedPlan,setUserSelectedPlan] = useState(null);
    const [selectedService,setSelectedService] = useState(null);
    const [creatingCheckout,setCreatingCheckout] = useState(false);
    const [error,setError] = useState(null);
    const [funcionariosTotal,setFuncionariosTotal] = useState(null);
    const formatCheckoutError = (status, body) => {
        if (typeof body === "string" && body.trim())
            return body;
        if (body && typeof body === "object") {
            const obj = body;
            const err = typeof obj.error === "string" ? obj.error : null;
            const message = typeof obj.message === "string" && obj.message.trim() ? obj.message.trim() : null;
            if (message) {
                const stripeStatus = typeof obj.stripe_status === "number" && Number.isFinite(obj.stripe_status) ?
obj.stripe_status : null;
                if (err === "stripe_error" && stripeStatus)
                    return `Stripe (HTTP ${stripeStatus}): ${message}`;
                return message;
            }
            if (err === "missing_supabase_env")
                return "Configuração do Supabase ausente no ambiente.";
            if (err === "network_error")
                return typeof obj.message === "string" && obj.message.trim() ? obj.message : "Falha de rede ao iniciar
pagamento.";
            if (err === "session_expired" || err === "invalid_jwt")
                return "Sessão expirada no Supabase. Saia e entre novamente.";
            if (err === "jwt_project_mismatch")
                return "Sessão do Supabase pertence a outro projeto. Saia e entre novamente.";
            if (err === "supabase_gateway_invalid_jwt")
                return "A Edge Function está exigindo JWT no gateway. Faça deploy com verify_jwt=false.";
            if (err === "function_not_deployed")
                return "A função payments não está publicada no Supabase.";
            const asJson = safeJson(body);
            if (err && asJson)
                return `Erro ao iniciar pagamento: ${err} (HTTP ${status}): ${asJson}`;
            if (err)
                return `Erro ao iniciar pagamento: ${err} (HTTP ${status}).`;
            if (asJson)
                return `Erro ao iniciar pagamento (HTTP ${status}): ${asJson}`;
        }
        return `Erro ao iniciar pagamento (HTTP ${status}).`;
    }
    ;
    const canShowFree = Boolean(usuario && usuario.plano === "free" && usuario.status_pagamento === "trial" &&
usuario.free_trial_consumido !== true);
    const plans = useMemo( () => [...canShowFree ? [{
        key: "free",
        title: "FREE",
        priceLabel: "R$ 0/mês",
        subtitle: "Para testar",
        bullets: ["Até 30 agendamentos por mês", "1 profissional", "Lembretes manuais (link do WhatsApp)", "Suporte por
email"]
    }] : [], {
        key: "basic",
        title: "BASIC",
        priceLabel: "R$ 34,99/mês",
        subtitle: "",
        bullets: ["Agendamentos 60 por mês", "1 profissional incluído", "Lembretes automáticos via WhatsApp", "Até 3
serviços", "Página pública personalizável", "Suporte por email"]
    }, {
        key: "pro",
        title: "PRO",
        priceLabel: "R$ 59,99/mês",
        subtitle: "Até 12 profissionais (8 inclusos + até 4 adicionais de R$ 7)",
        bullets: ["Até 8 profissionais incluídos", "Serviços ilimitados", "Logo e fotos de serviços", "Relatórios",
"Bloqueios recorrentes", "Suporte via WhatsApp"]
    }, {
```

```
        key: "enterprise",
        title: "EMPRESA",
        priceLabel: "R$ 98,99/mês",
        subtitle: "Ilimitado",
        bullets: ["Profissionais ilimitados", "Multi-unidades", "Agendamentos ilimitados", "Serviços ilimitados", "Logo e fotos de serviços", "Suporte via WhatsApp"]
    }], [canShowFree]);
    const services = useMemo( () => [{
        key: "setup_completo",
        title: "Setup Completo",
        priceLabel: "R$ 150 (uma vez)",
        bullets: ["Você configura tudo para o cliente", "Cadastra serviços, fotos, horários", "Conecta WhatsApp", "Testa envios", "Treina o cliente em 15 minutos"]
    }, {
        key: "consultoria_hora",
        title: "Consultoria por Hora",
        priceLabel: "R$ 80/hora",
        bullets: ["Ajuda com configurações avançadas", "Sugestões de otimização", "Dúvidas gerais"]
    }], []);
    const currentPlan = useMemo( () => {
        const current = (usuario?.plano ?? "").trim().toLowerCase();
        if (current === "free")
            return canShowFree ? "free" : "basic";
        if (current === "enterprise")
            return "enterprise";
        if (current === "team")
            return "pro";
        if (current === "basic" || current === "pro")
            return current;
        return "basic";
    }
    , [canShowFree, usuario?.plano]);
    const selectedPlan = userSelectedPlan ?? currentPlan;
    const defaultFuncionariosTotal = useMemo( () => {
        const raw = usuario?.limite_funcionarios;
        const n = typeof raw === "number" && Number.isFinite(raw) && raw > 0 ? Math.floor(raw) : 1;
        return Math.max(1, Math.min(200, n));
    }
    , [usuario?.limite_funcionarios]);
    const includedPro = 8;
    const maxPro = 12;
    const defaultProFuncionariosTotal = useMemo( () => {
        return Math.min(maxPro, Math.max(includedPro, defaultFuncionariosTotal));
    }
    , [defaultFuncionariosTotal, includedPro, maxPro]);
    const effectiveFuncionariosTotal = funcionariosTotal ?? (selectedPlan === "pro" ? defaultProFuncionariosTotal : defaultFuncionariosTotal);
    useEffect( () => {
        const run = async () => {
            const search = location.search ?? "";
            if (!search)
                return;
            const params = new URLSearchParams(search);
            const checkout = (params.get("checkout") ?? "").trim().toLowerCase();
            if (checkout !== "success" && checkout !== "cancel")
                return;
            const item = (params.get("item") ?? params.get("plano") ?? "").trim().toLowerCase() || null;
            setCheckoutNotice({
                kind: checkout,
                item
            });
            const sessionId = (params.get("session_id") ?? "").trim();
            const usuarioIdFromParams = (params.get("usuario_id") ?? "").trim() || null;
            const first = await refresh();
            const refreshedUsuarioId = first?.kind === "usuario" ? first.profile.id : null;
            const effectiveUsuarioId = refreshedUsuarioId ?? usuarioIdFromParams;
            if (checkout === "success") {
                if (sessionId) {
                    const sync = await syncCheckoutSessionPagamento(sessionId, effectiveUsuarioId);
                    if (!sync.ok) {
                        setError(formatCheckoutError(sync.status, sync.body));
                    } else {
```

```
                    await refresh();
                }
            }
            let tries = 0;
            while (tries < 10) {
                await new Promise( (r) => setTimeout(r, 2e3));
                const next = await refresh();
                if (next?.kind === "usuario" && next.profile.status_pagamento === "ativo")
                    break;
                tries += 1;
            }
        }
        navigate("/pagamento", {
            replace: true
        });
    }
    ;
    run().catch( () => void 0);
}
, [location.search, navigate, refresh]);
const startPlanCheckout = async (metodo) => {
    if (!usuarioId)
        return;
    const plan = selectedPlan;
    if (!plan || plan === "free") {
        setError("Selecione um plano válido.");
        return;
    }
    const requested = plan === "pro" ? Math.floor(effectiveFuncionariosTotal || includedPro) : 1;
    if (plan === "pro" && requested > maxPro) {
        setUserSelectedPlan("enterprise");
        setFuncionariosTotal(null);
        setError("Para mais de 12 profissionais, selecione o plano EMPRESA.");
        return;
    }
    const total = plan === "pro" ? Math.max(includedPro, Math.min(maxPro, requested)) : 1;
    setCreatingCheckout(true);
    setError(null);
    const res = await createCheckoutPagamento(usuarioId, plan, metodo, total);
    if (!res.ok) {
        setError(formatCheckoutError(res.status, res.body));
        setCreatingCheckout(false);
        return;
    }
    const body = res.body;
    const url = typeof body.url === "string" ? body.url : null;
    if (!url) {
        setError("A função não retornou o link de checkout.");
        setCreatingCheckout(false);
        return;
    }
    window.location.href = url;
}
;
const startServiceCheckout = async (metodo) => {
    if (!usuarioId || !selectedService)
        return;
    setCreatingCheckout(true);
    setError(null);
    const res = await createCheckoutPagamento(usuarioId, selectedService, metodo);
    if (!res.ok) {
        setError(formatCheckoutError(res.status, res.body));
        setCreatingCheckout(false);
        return;
    }
    const body = res.body;
    const url = typeof body.url === "string" ? body.url : null;
    if (!url) {
        setError("A função não retornou o link de checkout.");
        setCreatingCheckout(false);
        return;
    }
```

```
        window.location.href = url;
    }
    ;
    if (!usuario) {
        return /* @__PURE__ */
        jsxDEV(AppShell, {
            children: /* @__PURE__ */
            jsxDEV("div", {
                className: "text-slate-700",
                children: "Acesso restrito."
            }, void 0, false, {
                fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                lineNumber: 395,
                columnNumber: 9
            }, this)
        }, void 0, false, {
            fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
            lineNumber: 394,
            columnNumber: 7
        }, this);
    }
    const formatStatusPagamento = (value) => {
        const v = String(value ?? "").trim().toLowerCase();
        if (!v)
            return "—";
        if (v === "ativo")
            return "Ativo";
        if (v === "trial")
            return "Trial";
        if (v === "inadimplente")
            return "Inadimplente";
        if (v === "suspenso")
            return "Suspenso";
        if (v === "cancelado")
            return "Cancelado";
        return value;
    }
    ;
    const statusTone = usuario.status_pagamento === "inadimplente" ? "red" : usuario.status_pagamento === "ativo" ?
"green" : "slate";
    const planoLabel = (planoRaw) => {
        const p = String(planoRaw ?? "").trim().toLowerCase();
        if (p === "enterprise")
            return "EMPRESA";
        if (p === "team")
            return "PRO";
        if (p === "pro")
            return "PRO";
        if (p === "basic")
            return "BASIC";
        if (p === "free")
            return "FREE";
        return planoRaw;
    }
    ;
    return /* @__PURE__ */
    jsxDEV(AppShell, {
        children: /* @__PURE__ */
        jsxDEV("div", {
            className: "space-y-6",
            children: [checkoutNotice ? /* @__PURE__ */
            jsxDEV("div", {
                className: ["rounded-xl border p-4 text-sm", checkoutNotice.kind === "success" ? "border-emerald-200 bg-
emerald-50 text-emerald-800" : "border-amber-200 bg-amber-50 text-amber-900"].join(" "),
                children: /* @__PURE__ */
                jsxDEV("div", {
                    className: "flex flex-wrap items-center justify-between gap-3",
                    children: [/* @__PURE__ */
                    jsxDEV("div", {
                        className: "space-y-1",
                        children: [/* @__PURE__ */
                        jsxDEV("div", {
```

```
                        className: "font-semibold",
                        children: checkoutNotice.kind === "success" ? "Pagamento concluído" : "Pagamento cancelado"
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 435,
                        columnNumber: 17
                    }, this), /* @__PURE__ */
                    jsxDEV("div", {
                        children: [checkoutNotice.item ? `Item: ${checkoutNotice.item.toUpperCase()}. ` : "",
"Status: ", formatStatusPagamento(usuario.status_pagamento), "."]
                    }, void 0, true, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 436,
                        columnNumber: 17
                    }, this)]
                }, void 0, true, {
                    fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                    lineNumber: 434,
                    columnNumber: 15
                }, this), /* @__PURE__ */
                jsxDEV("div", {
                    className: "flex items-center gap-2",
                    children: [/* @__PURE__ */
                    jsxDEV(Button, {
                        variant: "secondary",
                        onClick: () => void refresh(),
                        children: "Atualizar"
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 441,
                        columnNumber: 17
                    }, this), /* @__PURE__ */
                    jsxDEV(Button, {
                        variant: "secondary",
                        onClick: () => setCheckoutNotice(null),
                        children: "Fechar"
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 444,
                        columnNumber: 17
                    }, this)]
                }, void 0, true, {
                    fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                    lineNumber: 440,
                    columnNumber: 15
                }, this)]
            }, void 0, true, {
                fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                lineNumber: 433,
                columnNumber: 13
            }, this)
        }, void 0, false, {
            fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
            lineNumber: 427,
            columnNumber: 9
        }, this) : null, error ? /* @__PURE__ */
        jsxDEV("div", {
            className: "rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700",
            children: error
        }, void 0, false, {
            fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
            lineNumber: 452,
            columnNumber: 18
        }, this) : null, /* @__PURE__ */
        jsxDEV(Card, {
            children: /* @__PURE__ */
            jsxDEV("div", {
                className: "p-6 space-y-4",
                children: [/* @__PURE__ */
                jsxDEV("div", {
                    className: "flex flex-wrap items-start justify-between gap-3",
                    children: [/* @__PURE__ */
```

```
                jsxDEV("div", {
                    children: [/* @__PURE__ */
                    jsxDEV("div", {
                        className: "text-sm font-semibold text-slate-900",
                        children: "Pagamento"
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 458,
                        columnNumber: 17
                    }, this), /* @__PURE__ */
                    jsxDEV("div", {
                        className: "text-sm text-slate-600",
                        children: ["Plano atual: ", planoLabel(usuario.plano)]
                    }, void 0, true, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 459,
                        columnNumber: 17
                    }, this)]
                }, void 0, true, {
                    fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                    lineNumber: 457,
                    columnNumber: 15
                }, this), /* @__PURE__ */
                jsxDEV("div", {
                    className: "flex items-center gap-2",
                    children: [/* @__PURE__ */
                    jsxDEV(Badge, {
                        tone: usuario.ativo ? "green" : "red",
                        children: usuario.ativo ? "Conta: Ativa" : "Conta: Inativa"
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 464,
                        columnNumber: 17
                    }, this), /* @__PURE__ */
                    jsxDEV(Badge, {
                        tone: statusTone,
                        children: ["Pagamento: ", formatStatusPagamento(usuario.status_pagamento)]
                    }, void 0, true, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 465,
                        columnNumber: 17
                    }, this), /* @__PURE__ */
                    jsxDEV(Button, {
                        variant: "secondary",
                        onClick: () => void refresh(),
                        children: "Atualizar status"
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 466,
                        columnNumber: 17
                    }, this)]
                }, void 0, true, {
                    fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                    lineNumber: 463,
                    columnNumber: 15
                }, this)]
            }, void 0, true, {
                fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                lineNumber: 456,
                columnNumber: 13
            }, this), /* @__PURE__ */
            jsxDEV("div", {
                className: "grid grid-cols-1 gap-3 sm:grid-cols-2 lg:grid-cols-3",
                children: plans.map( (p) => {
                    const selected = selectedPlan === p.key;
                    const clickable = p.key !== "free";
                    const best = p.key === "pro";
                    return /* @__PURE__ */
                    jsxDEV("button", {
                        type: "button",
                        onClick: () => {
                            if (!clickable)
```

```
                                        return;
                                    setUserSelectedPlan(p.key);
                                    setFuncionariosTotal(null);
                                }
                                ,
                                className: ["text-left rounded-xl border bg-white p-4 transition", best ? selected ?
"border-slate-900 ring-2 ring-slate-900/10 bg-amber-50" : "border-amber-300 hover:bg-amber-50" : selected ? "border-
slate-900 ring-2 ring-slate-900/10" : "border-slate-200 hover:bg-slate-50", clickable ? "" : "cursor-default"].join("
"),
                                children: [/* @__PURE__ */
                                jsxDEV("div", {
                                    className: "flex items-start justify-between gap-3",
                                    children: [/* @__PURE__ */
                                    jsxDEV("div", {
                                        children: [/* @__PURE__ */
                                        jsxDEV("div", {
                                            className: "text-sm font-semibold text-slate-900",
                                            children: p.title
                                        }, void 0, false, {
                                            fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                            lineNumber: 500,
                                            columnNumber: 25
                                        }, this), /* @__PURE__ */
                                        jsxDEV("div", {
                                            className: "text-xs text-slate-600",
                                            children: [p.priceLabel, p.subtitle ? ` • ${p.subtitle}` : ""]
                                        }, void 0, true, {
                                            fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                            lineNumber: 501,
                                            columnNumber: 25
                                        }, this)]
                                    }, void 0, true, {
                                        fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                        lineNumber: 499,
                                        columnNumber: 23
                                    }, this), /* @__PURE__ */
                                    jsxDEV("div", {
                                        className: "flex items-center gap-2",
                                        children: [best ? /* @__PURE__ */
                                        jsxDEV(Badge, {
                                            tone: "yellow",
                                            children: "Melhor opção"
                                        }, void 0, false, {
                                            fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                            lineNumber: 504,
                                            columnNumber: 33
                                        }, this) : null, selected ? /* @__PURE__ */
                                        jsxDEV(Badge, {
                                            tone: "slate",
                                            children: "Selecionado"
                                        }, void 0, false, {
                                            fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                            lineNumber: 505,
                                            columnNumber: 37
                                        }, this) : null]
                                    }, void 0, true, {
                                        fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                        lineNumber: 503,
                                        columnNumber: 23
                                    }, this)]
                                }, void 0, true, {
                                    fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                    lineNumber: 498,
                                    columnNumber: 21
                                }, this), /* @__PURE__ */
```

```
                              jsxDEV("div", {
                                  className: "mt-3 space-y-1",
                                  children: p.bullets.map( (b) => /* @__PURE__ */
                                  jsxDEV("div", {
                                      className: "text-xs text-slate-700",
                                      children: ["- ", b]
                                  }, b, true, {
                                      fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                      lineNumber: 510,
                                      columnNumber: 23
                                  }, this))
                              }, void 0, false, {
                                  fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                  lineNumber: 508,
                                  columnNumber: 21
                              }, this)]
                          }, p.key, true, {
                              fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                              lineNumber: 478,
                              columnNumber: 19
                          }, this);
                      }
                      )
                  }, void 0, false, {
                      fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                      lineNumber: 472,
                      columnNumber: 13
                  }, this), selectedPlan === "pro" ? /* @__PURE__ */
                  jsxDEV("div", {
                      className: "grid grid-cols-1 gap-3 sm:grid-cols-2",
                      children: [/* @__PURE__ */
                      jsxDEV(Input, {
                          label: "Profissionais",
                          type: "number",
                          min: includedPro,
                          max: maxPro,
                          value: String(effectiveFuncionariosTotal),
                          onChange: (e) => {
                              const raw = e.target.value;
                              const n = raw.trim() === "" ? includedPro : Number(raw);
                              if (!Number.isFinite(n))
                                  return;
                              const i = Math.floor(n);
                              if (i > maxPro) {
                                  setUserSelectedPlan("enterprise");
                                  setFuncionariosTotal(null);
                                  setError("Para mais de 12 profissionais, selecione o plano EMPRESA.");
                                  return;
                              }
                              const clamped = Math.max(includedPro, Math.min(maxPro, i));
                              setFuncionariosTotal(clamped);
                          }
                      }, void 0, false, {
                          fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                          lineNumber: 522,
                          columnNumber: 17
                      }, this), /* @__PURE__ */
                      jsxDEV("div", {
                          className: "rounded-xl border border-slate-200 bg-slate-50 p-4 text-xs text-slate-700 flex
items-center",
                          children: "O checkout calcula 8 profissionais inclusos + adicional por profissional acima de
8 (máximo 12 no PRO)."
                      }, void 0, false, {
                          fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                          lineNumber: 543,
                          columnNumber: 17
                      }, this)]
                  }, void 0, true, {
                      fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                      lineNumber: 521,
```

```
                    columnNumber: 13
                }, this) : null, /* @__PURE__ */
                jsxDEV("div", {
                    className: "flex flex-wrap justify-end gap-2",
                    children: [/* @__PURE__ */
                    jsxDEV(Button, {
                        variant: "secondary",
                        onClick: () => void startPlanCheckout("pix"),
                        disabled: creatingCheckout || selectedPlan === "free",
                        children: creatingCheckout ? "Abrindo…" : "PIX (30 dias)"
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 550,
                        columnNumber: 15
                    }, this), /* @__PURE__ */
                    jsxDEV(Button, {
                        onClick: () => void startPlanCheckout("card"),
                        disabled: creatingCheckout || selectedPlan === "free",
                        children: creatingCheckout ? "Abrindo…" : "Cartão (assinatura)"
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 553,
                        columnNumber: 15
                    }, this)]
                }, void 0, true, {
                    fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                    lineNumber: 549,
                    columnNumber: 13
                }, this)]
            }, void 0, true, {
                fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                lineNumber: 455,
                columnNumber: 11
            }, this)
        }, void 0, false, {
            fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
            lineNumber: 454,
            columnNumber: 9
        }, this), /* @__PURE__ */
        jsxDEV(Card, {
            children: /* @__PURE__ */
            jsxDEV("div", {
                className: "p-6 space-y-4",
                children: [/* @__PURE__ */
                jsxDEV("div", {
                    children: [/* @__PURE__ */
                    jsxDEV("div", {
                        className: "text-sm font-semibold text-slate-900",
                        children: "Serviços"
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 563,
                        columnNumber: 15
                    }, this), /* @__PURE__ */
                    jsxDEV("div", {
                        className: "text-sm text-slate-600",
                        children: "Contrate serviços avulsos para configuração e suporte."
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 564,
                        columnNumber: 15
                    }, this)]
                }, void 0, true, {
                    fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                    lineNumber: 562,
                    columnNumber: 13
                }, this), /* @__PURE__ */
                jsxDEV("div", {
                    className: "grid grid-cols-1 gap-3 sm:grid-cols-2",
                    children: services.map( (s) => {
                        const selected = selectedService === s.key;
                        return /* @__PURE__ */
```

```
                            jsxDEV("button", {
                                type: "button",
                                onClick: () => setSelectedService(s.key),
                                className: ["text-left rounded-xl border bg-white p-4 transition", selected ? "border-
slate-900 ring-2 ring-slate-900/10" : "border-slate-200 hover:bg-slate-50"].join(" "),
                                children: [/* @__PURE__ */
                                jsxDEV("div", {
                                    className: "flex items-start justify-between gap-3",
                                    children: [/* @__PURE__ */
                                    jsxDEV("div", {
                                        children: [/* @__PURE__ */
                                        jsxDEV("div", {
                                            className: "text-sm font-semibold text-slate-900",
                                            children: s.title
                                        }, void 0, false, {
                                            fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                            lineNumber: 582,
                                            columnNumber: 25
                                        }, this), /* @__PURE__ */
                                        jsxDEV("div", {
                                            className: "text-xs text-slate-600",
                                            children: s.priceLabel
                                        }, void 0, false, {
                                            fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                            lineNumber: 583,
                                            columnNumber: 25
                                        }, this)]
                                    }, void 0, true, {
                                        fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                        lineNumber: 581,
                                        columnNumber: 23
                                    }, this), selected ? /* @__PURE__ */
                                    jsxDEV(Badge, {
                                        tone: "slate",
                                        children: "Selecionado"
                                    }, void 0, false, {
                                        fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                        lineNumber: 585,
                                        columnNumber: 35
                                    }, this) : null]
                                }, void 0, true, {
                                    fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                    lineNumber: 580,
                                    columnNumber: 21
                                }, this), /* @__PURE__ */
                                jsxDEV("div", {
                                    className: "mt-3 space-y-1",
                                    children: s.bullets.map( (b) => /* @__PURE__ */
                                    jsxDEV("div", {
                                        className: "text-xs text-slate-700",
                                        children: ["- ", b]
                                    }, b, true, {
                                        fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                        lineNumber: 589,
                                        columnNumber: 23
                                    }, this))
                                }, void 0, false, {
                                    fileName:
"C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                    lineNumber: 587,
                                    columnNumber: 21
                                }, this)]
                            }, s.key, true, {
                                fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                                lineNumber: 571,
                                columnNumber: 19
```

```
                        }, this);
                    }
                    )
                }, void 0, false, {
                    fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                    lineNumber: 567,
                    columnNumber: 13
                }, this), /* @__PURE__ */
                jsxDEV("div", {
                    className: "flex flex-wrap justify-end gap-2",
                    children: [/* @__PURE__ */
                    jsxDEV(Button, {
                        variant: "secondary",
                        onClick: () => void startServiceCheckout("pix"),
                        disabled: creatingCheckout || !selectedService,
                        children: creatingCheckout ? "Abrindo…" : "Pagar com PIX"
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 600,
                        columnNumber: 15
                    }, this), /* @__PURE__ */
                    jsxDEV(Button, {
                        onClick: () => void startServiceCheckout("card"),
                        disabled: creatingCheckout || !selectedService,
                        children: creatingCheckout ? "Abrindo…" : "Pagar com cartão"
                    }, void 0, false, {
                        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                        lineNumber: 603,
                        columnNumber: 15
                    }, this)]
                }, void 0, true, {
                    fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                    lineNumber: 599,
                    columnNumber: 13
                }, this)]
            }, void 0, true, {
                fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
                lineNumber: 561,
                columnNumber: 11
            }, this)
        }, void 0, false, {
            fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
            lineNumber: 560,
            columnNumber: 9
        }, this)]
    }, void 0, true, {
        fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
        lineNumber: 425,
        columnNumber: 7
    }, this)
}, void 0, false, {
    fileName: "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx",
    lineNumber: 424,
    columnNumber: 5
}, this);
}
_s(PagamentoPage, "CSY4+SwL8YVqV+gk+qleO6U6S34=", false, function() {
    return [useAuth, useLocation, useNavigate];
});
_c = PagamentoPage;
var _c;
$RefreshReg$(_c, "PagamentoPage");
import*as RefreshRuntime from "/@react-refresh";
const inWebWorker = typeof WorkerGlobalScope !== "undefined" && self instanceof WorkerGlobalScope;
if (import.meta.hot && !inWebWorker) {
    if (!window.$RefreshReg$) {
        throw new Error("@vitejs/plugin-react can't detect preamble. Something is wrong.");
    }
    RefreshRuntime.__hmr_import(import.meta.url).then( (currentExports) => {

RefreshRuntime.registerExportsForReactRefresh("C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx"
, currentExports);
```

```
        import.meta.hot.accept( (nextExports) => {
            if (!nextExports)
                return;
            const invalidateMessage =
RefreshRuntime.validateRefreshBoundaryAndEnqueueUpdate("C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/Pagamento
Page.tsx", currentExports, nextExports);
            if (invalidateMessage)
                import.meta.hot.invalidate(invalidateMessage);
        }
        );
    }
    );
}
function $RefreshReg$(type, id) {
    return RefreshRuntime.register(type, "C:/Users/Admin/Desktop/SMagenda/smagenda/src/views/app/PagamentoPage.tsx " +
id);
}
function $RefreshSig$() {
    return RefreshRuntime.createSignatureFunctionForTransform();
}
```

```
//#
sourceMappingURL=data:application/json;base64,eyJ2ZXJzaW9uIjozLCJtYXBwaW5ncyI6IkFBMFlROztBQTFZUixTQUFTQSxXQUFXQyxTQUFTQy
xnQkFBZ0I7QUFDN0MsU0FBU0MsYUFBYUMsbUJBQW1CO0FBQ3pDLFNBQVNDLGdCQUFnQjtBQUN6QixTQUFTQyxhQUFhO0FBQ3RCLFNBQVNDLGNBQWM7QUFDdk
IsU0FBU0MsWUFBWTtBQUNyQixTQUFTQyxhQUFhO0FBQ3RCLFNBQVNDLGlCQUFpQkMsVUFBVUMsbUJBQW1CO0FBQ3ZELFNBQVNDLGVBQWU7QUFJeEIsU0FBU0
MsU0FBU0MsT0FBZ0I7QUFDaEMsTUFBSSTtBQUNGLFdBQU9DLEtBQUtDLFVBQVVGLEtBQUs7QUFBQSxFQUM3QixRQUFRO0FBQ04sV0FBTztBQUFBLEVBQ1Q7QU
FDRjtBQUVBLGVBQWVHLGVBQWVDLE1BQWtEO0FBQzlFLE1BQUksQ0FBQ1AsWUFBWVEsSUFBSTtBQUNuQixXQUFPLEVBQUVBLElBQUksT0FBZ0JDLFFBQVEsR0
FBR0YsTUFBTSxFQUFFRyxPQUFPLHVCQUF1QixFQUFFO0FBQUEsRUFDbEY7QUFFQSxRQUFNQyxjQUFjQyxPQUFPWixZQUFZYSxPQUFPQyxxQkFBcUIsRUFBRS
xFQUNsRUMsS0FBSyxFQUNMQyxRQUFRLHdCQUF3QixFQUFFLEVBQ2xDQSxRQUFRLFNBQVMsRUFBRTtBQUN0QixRQUFNQyxrQkFBa0JMLE9BQU9aLFlBQVlhLE
9BQU9LLDBCQUEwQixFQUFFLEVBQzNFSCxLQUFLLEVBQ0xDLFFBQVEsd0JBQXdCLEVBQUU7QUFDckMsUUFBTUcsUUFBUSxHQUFHUixXQUFXXO0FBRTVCLFFBQU
1TLGFBQWEsWUFBWTtBQUM3QixVQUFNLEVBQUVDLE1BQU1DLFdBQVdaLE9BQU9hLFdBQVcsSUFBSSxNQUFNeEIsU0FBU3lCLEtBQUtDLGVBQWU7QUFDbEYsUU
FBSUYsV0FBWSxRQUFPO0FBQ3ZCLFdBQU9VLEtBQUdBQU9ELFVBQVVJLFdBQVc7QUFDbEMsUUFBSSxDQUFDUCxXQUFXUSxLQUFLO0FBQUEsRUFDaEQsUUFBTyx1Q0FBc0I7QUFDcEIsVUFBTUUsUUFBUSxHQUFHYixnQkFBZ0JnQixLQUFLd0JDLHNCQUFzQkM
FTeUIsS0FBS0ksV0FBVztBQUM3RCxNQUFJRixVQUFVQyxZQUFZRDtBQUMxQixRQUFNRyxNQUFNQyxLQUFLQyxNQUFNQyxLQUFLSCxJQUFJLElBQUksR0FBSzR0FBST
tBQUN4QyxRQUFNSSxZQUFZUCxTQUxjQUFjO0FBRXpDLE1BQUlSLFlBQVksQ08sYUFBYUEsYUFBYUosTUFBTSxLQUFLO0FBQ3BELFVBQU1QLFlBQV
ksTUFBTUYsV0FBVztBQUNuQyxRQUFJRSxVQUFXSSxXQUFVSjtBQUFBQSxFQUMzQjtBQUVBLE1BQUlJLFNBQVM7QUFDWCxVQUFNLEVBQUVoQixPQUFPeUIsUUFBUeUIsUU
FBUSxJQUFJLE1BQU1wQyxTQUFTS1ksUUFBUUMsTUFBTUN2RCxVQUFNNkRCxVQUFNQyxhQUFhLE9BQU9GLMNBQVNHLFlBQVksV0FBV0gsUUFBUUMsTUFBTM1RS
xRQUFJSCxXQUFXLGlCQUFpQkksS0FBS0YsYUFBYVQsUHO0FBQ2hELFlBQU1uLFlBQVksTUFBTVAsV0FBVztBQUNuQyxRQUFJRSxVQUFXSSxXQUFVSjtBQU
FBQSxJQUMzQjtBQUFBLEVBQ0Y7QUFFQSxRQUFNa0IsUUFBUUMsS0FBS0wsWUFBWTtBQUNyQyxNQUFJRyxNQUFNQSxLQUFLTyxLQUFLRSxVQUFVO0FBQ3ZCLFdBQVcsUU
lBQUksT0FBZ0JDLFFBQVEsS0FBS0MsTUFBTU4sT0FBT1FBQUEsRUFDdkQ7QUFFQSxRQUFNZ0MsWUFBWTBELEVBQVUsU1VBQUEsRXVBQUE,IAG,GAAO,KAAK,EAAE,CAAGBDLE
9BQU83QixXQUFXO0FBQ3ZELE1BQUksQ0FBQytCLGFBQWFsQyxJQUFJO0FBQ3BCLFVBQU1ULFNBQVN5QixLQUFLbUIsUUFBUSxFQUFFQyxNQUFNLE1BQU1DLE1BQU1DLE
1BQVM7QUFDbkQsV0FBTyxFQUFFckMsSUFBSSxPQUFnQkMsUUFBUSxLQUFLRixNQUFNLEVBQVHLE9BQU8sd0JBQXdCcb0MsS0FBS0osaUJBUkksSUFBSUgsTUFBSSxFQUFFRy
FBVUwsYUFBYU0sZUFBZSxFQUFFO0FBQUEsRUFDbEEo7QUFFQSxRQUFNQyxZQUFZLE9BQU9DLFFBQWdCO0FBQ3ZDLFFBQUlDO0FBQ0osUUFBSTtBQUNGNGQSxZQU
FNLE1BQU1qQyxPQUFPO0FBQUEsRUFDakJrQyxRQUFRO0FBQUEsRUFDUkMsUUFBQUEsRUFDUkMsUUFBQUEsRUFDUkMsUUFBQUEsRUFDWEMsT0FBTyxFQUFFRyx
tBQUFBQSxVQUNSdUMsZUFBZSxVQUFVVixHQUFHO0FBQUEsUUFDOUIsRUFDQUEsTUFBTSxLQUFLZ0IsS0FBS0EsWUFBWTtBQUNwQixRQUFNQyxLQUFLQSxLQUNLLENBQU
M7QUFBQSxRQUNJLFNBQVNILFFBQVNrRCxHQUFHO0FBQUFBLFVBQ2xDLENBQUMsRUFBRUQsRUFDZ0IsQ0FBQ1IsTUFBTSxZQUFZLENBQUMsQ0FBQyxFQUFFLEVBQUFBQSxRQUFBQSxVQUFVO0FBQ3ZCLFdBQU8sRUFBRVgsSUFBSSxPQUFnQkMsUUFBUSxLQUFLRixNQUFNLEVBQUVHLE9BQU8s
FBZ0JDLFFBQVEsSUFBSUYsTUFBTSxFQUFFRyxPQUFPLDBCQUEwQixFQUFFO0FBQUEsRUFDbEY7QUFFQSxRQUFNcUIsVUFBVSxNQUFNN0IsZ0JBQWdCLEtBQUssRUFBRXFCLE
lBQUksT0FBZ0JDLFFBQVEsS0FBS0MsTUFBTU4sT0FBT1FBQUEsRUFDdkQ7QUFFQSxRQUFNZ0MsWUFBWTBELEVBQVUsU1VBQUEsRXVBQUFBQUEsQ0FBQyxDQUFDLENBQUM7
FBVUwsYUFBYU0sZUFBZSxFQUFFO0FBQUEsRUFDbEY7QUFFQSxRQUFNQyxZQUFZLE9BQU9DLFFBQWdCO0FBQ3ZDLFFBQUlDO0FBQ0osUUFBSTtBQUNGLE1BQ1
9BQU83QixXQUFXO0FBQ3ZELE1BQUksQ0FBQytCLGFBQWFsQyxJQUFJO0FBQ3BCLFVBQU1ULFNBQVN5QixLQUFLbUIsUUFBUSxFQUFFQyxNQUFNLE1BQU1DLE1BQU1DLE
FPO0FBQUEsRUFDVUFDTHhELEElBQUk7QUFDQSxRQUFNT1EsVUFBQUEsRUFDUkMsUUFBUSxLQUFLRixNQUFNLEVBQUVHLE9BQU8sZ0JBQWdCLEVBQUU7QUFDNUQsUUFBTUYsUUFBUSxDQUFDQyxLQUFLSixpQkFBaUJKLFFBQVFKLE9BQU9pQixLQUFLQyxT
VBQ1g7QUFBQSxRQUNSdUNHQUFMUhRSx0VBQ0FDLENBQUNLLEtBQUtDLFlBQVk0QkMsV0FBV0csRUFBRUMsT0FBT0MsV0FBV1BBQUE7QUFBQSxRQUNYLENBQUMsRUFBRUQsRUFDZ0IsQ0FBQ1IsTUFBTSxZQUFZLENBQUMsQ0FBQyxFQUFFLEVBQUFBQSxRQUFBQSxVQUFVO0FBQ3ZCL
ZBQUVELE1BQU0sRUFBRUcsT0FBTyxrQkFBa0IsRUFBRTtBQUNuQyxVQUFNQyxLQUFLQSxLQUNLLENBQU1
FBQUEsRUFDVUFDTHhELEElBQUk7QUFDQSxRQUFNT1EsVUFBQUEsRUFDUkMsUUFBUSxLQUFLRixNQUFNLEVBQUVHLE9BQU8sZ0JBQWdCLEVBQUU7QUFDNUQsUUFBTUYsUUFBUSxDQUFDQyxLQUFLSixpQkFBaUJKLFFBQVFKLE9BQU9pQixLQUFLQyxT
xXQUFXbUIsZ0JBQWdCLEtBQUssRUFBRXFCLE1BQU1DLE1BQU1DLEtBQUtDLFlBQVk0QkMsV0FBV0csRUFBRUMsT0FBT0MsV0FBV1BBQUE7QUFBQSxRQUNYLENBQUMsRUFBRUQsRUFDZ0IsQ0FB
tBQUNuRCxhQUFPLEVBQUVBLElBQUksT0FBZ0JDLFFBQVEsS0FBS0YsTUFBTSxFQUFFRyxPQUFPLDRCQUE0QixFQUFFO0FBQ25EO0FBRUEsUUFBTUMsWUFBWSxPQUFPQyxRQUFnQjtBQUN2Q
xjQUFjeEUsZ0JBQWdCdUUsV0FBVzFELEFBQUEsRUFDdkQ7QUFFQSxRQUFNZ0MsWUFBWTBELEVBQVUsU1VBQUEsRXVBQUFBQUEsQ0FBQyxDQUFDLENBQUMsRUFBRUQsRUFDZ0IsQ0FBQ1IsTUFBTSxZQUFZLENBQUMsQ0FBQyxF
VBQUVDLE1BQU0sRUFBRUcsT0FBTyxrQkFBa0IsRUFBRTtBQUNuQyxVQUFNQyxLQUFLQSxLQUNLLENBQU1DLHZCQUF3QixFQUFFLEVBQ2xDQSxRQUFRLFNBQVMsRUFBRTtBQUN0QixRQUFNQyxrQkFBa0Jx3QkFBd0JvQyxLQUFLLEVBQU
FLd0IsWUFBWXhCLEtBQUtDLFlBQVk0QkMsV0FBV0csRUFBRUMsT0FBT0MsV0FBV1BBQUE7QUFBQSxRQUNYLENBQUMsRUFBRUQsRUFDZ0IsQ0FBQ1IsTUFBTSxZQUFZLENBQUMsQ0FBQyxFQUFFLEVBQU
EsU0FBT0Q7QUFDVDtBQUVBLFNBQVNDLGVBQWVDLEdBQWtCQyxLQUFhQyxNQUFNQyxNQUFNLEVBQUUsQ0FBQ0MsTUFBTUQsTUFBTUMsTUFBTUMsTUFBTUMsTUFBTUMsTUFBTUMsTUFBTUMsTUFBTUMs
JDLFlBQVlOLFdBQVdPLE9BQU9NLE1BQU1DLE1BQU1ELEtBQUtELE9BQU9HLE1BQU1JLEtBQUtDLFlBQVk0QkMsV0FBV0csRUFBRUMsT0FBT0MsV0FBV1BBQUE7QUFBQSxRQUNYLENBQUMsRUFBRUQsRU
FNLHFCQUFxQlQsRlxBJ7QUFDOUcsU0FBT3FELFlBQVlXWUdBQVUsZ0JBQWdCLEVBQUU7QUFDbEQsUUFBTUYsUUFBUSxDQUFDQyxLQUFLSixpQkFBaUJKLFFBQVFKLE9BQU9pQixLQUFLQyxTVBQ1g7QUFBQSxRQUNSdUNHQUFMUhRSx0VBQ0FD
MsRUFBRUMsUUFBUSxLQUFLRixNQUFNLEVBQUVHLE9BQU8sZ0JBQWdCLEVBQUU7QUFDNUQsUUFBTUYsUUFBUSxDQUFDQyxLQUFLSixpQkFBaUJKLFFBQVFKLE9BQU9pQixLQUFLQyxTVBQ1g7QUFBQSxRQUNSdUNHQUFMUhRSx0VBQ0FDLENBQ
9CO0FBbUJPLGdCQUFnQkMsSUFBSUMsUUFBUSxNQUFNLEVBQUVDLE1BQU1DLE1BQU1DLEtBQUtDLFlBQVk0QkMsV0FBV0csRUFBRUMsT0FBT0MsV0FBV1BBQUE7QUFBQSxRQUNYLENBQUMsRUFBRUQsRUFDZ0Is
duRyxZQUFZO0FBQ3pDLFFBQUlDLFVBQVU0QixXQUFVSjtBQUFBQSxFQUMzQjtBQUVBLE1BQUlJLFNBQVM7QUFDWCxVQUFNLEVBQUVoQixPQUFPeUIsUUFBUXlCLFFBQVEsSUFBSSxNQUFNcEMsU0FBU0tZQUFBLEtBQU
10QixZQUFZb0IsUUFBU0ksWUFBWTtBQUNqQyxNQUFJRyxNQUFNQSxLQUFLTyxLQUFLRSxVQUFVO0FBQ3ZCLFdBQU8sRUFBRXRCLElBQUksT0FBZ0JDLFFBQVEsSUFBSUYsTUFBTSxFQUFFRyxPQUFPLGlCQUFpQixFQUFF
tCQUFrQixFQUFFO0FBQ25CLFFBQU1DLEtBQUtDLFlBQVk0QkMsV0FBV0csRUFBRUMsT0FBT0MsV0FBV1BBQUE7QUFBQSxRQUNYLENBQUMsRUFBRUQsRUFDZ0IsQ0FBQ1IsTUFBTSxZQUFZLENBQUMsQ0FBQyxFQUFFLEVBQUFBQ
FBQU0sQ0FBQ2dILGtCQUFrQixFQUFFO0FBQ25CLFFBQU1qQyxLQUFLQSxLQUNLLENBQUM7QUFBQSxRQUNJLFNBQVNILFFBQVNrRCxHQUFHO0FBQUFBLFVBQ2xDLENBQUMsRUFBRUQsRUFDZ0IsQ0FBQ1IsTUFBTSxZQUFZ
k7QUFDdEQsUUFBTUYsUUFBUSxDQUFDcUYsbUJBQW1COEIsb0JBQW9CQyxLQUFLQyxZQUFZ4QkMsV0FBV0csRUFBRUMsT0FBT0MsV0FBV1BBQUE7QUFBQSxRQUNYLENBQUMsRUFBRUQsRUFDZ0YsQ0FBQ1IsTUFBTSxZQUFZLENBQUMsQ0FBQyxFQUFFLEVBQUFBQSxRQUFBQSxVQUFVO0FBQ3ZCL
```

```
FBa0I7QUFDN0QsUUFBSSxPQUFPQSxTQUFTLFlBQVlBLEtBQUtRLEtBQUssRUFBRyxRQUFPUjtBQUNwRCxRQUFJQSxRQUFRLE9BQU9BLFNBQVMsVUFBVTtBQU
NwQyxZQUFNb0csTUFBTXBHO0FBQ1osWUFBTXFHLE1BQU0sT0FBT0QsSUFBSWpHLFVBQVUsV0FBV2lHLElBQUlqRyxRQUFRO0FBQ3hELFlBQU00QixVQUFVLE
9BQU9xRSxJQUFJckUsWUFBWSxZQUFZcUUsSUFBSXJFLFFBQVF2QixLQUFLLElBQUk0RixJQUFJckUsUUFBUXZCLEtBQUssSUFBSTtBQUM3RixVQUFJdUIsU0
FBUztBQUNYLGNBQU11RSxlQUFlLE9BQU9GLElBQUlHLGtCQUFrQixZQUFZOUIsT0FBT0MsU0FBUzBCLElBQUlHLGFBQWEsSUFBSUgsSUFBSUcsZ0JBQWdCO0
FBQ3ZILFlBQUlGLFFBQVEsa0JBQWtCQyxhQUFjLFFBQU8sZ0JBQWdCQSxZQUFZLE1BQU12RSxPQUFPO0FBQzVGLGVBQU9BO0FBQUFBLE1BQ1Q7QUFDQSxVQU
FJc0UsUUFBUSx1QkFBd0IsUUFBTztBQUMzQyxVQUFJQSxRQUFRLGdCQUFpQixRQUFPLE9BQU9ELElBQUlyRSxZQUFZLFlBQVlxRSxJQUFJckUsUUFBUXZCLE
tBQUssSUFBSTRGLElBQUlyRSxVQUFVO0FBQzFHLFVBQUlzRSxRQUFRLHFCQUFxQkEsUUFBUSxjQUFlLFFBQU87QUFDL0QsVUFBSUEsUUFBUSx1QkFBd0IsUU
FBTztBQUMzQyxVQUFJQSxRQUFRLCtCQUFnQyxRQUFPO0FBQ25ELFVBQUlBLFFBQVEsd0JBQXlCLFFBQU87QUFDNUMsWUFBTUcsU0FBUzdHLFNBQVNLLElBQU
k7QUFDNUIsVUFBSXFHLE9BQU9HLE9BQVEsUUFBTyw4QkFBOEJILEdBQUcsVUFBVW5HLE1BQU0sTUFBTXNHLE1BQU07QUFDdkYsVUFBSUgsSUFBSyxRQUFPLD
hCQUE4QkEsR0FBRyxVQUFVbkcsTUFBTTtBQUNqRSxVQUFJc0csT0FBUSxRQUFPLG1DQUFtQ3RHLE1BQU0sTUFBTXNHLE1BQU07QUFBQSxJQUMxRTtBQUNBLF
dBQU8sbUNBQW1DdEcsTUFBTTtBQUFBLEVBQ2xEO0FBRUEsUUFBTXVHLGNBQWNDLFFBQVFyQixXQUFXQSxRQUFRYixVQUFVLFVBQVVhLFFBQVFzQixxQkFBcU
IsV0FBV3RCLFFBQVF1Qix5QkFBeUIsSUFBSTtBQUVoSixRQUFNQyxRQUFRL0g7QUFBQUEsSUFDWixNQUNFO0FBQUEsTUFDRSxHQUFJMkgsY0FDQTtBQUFBLF
FBQ0U7QUFBQSxVQUNFSyxLQUFLO0FBQUEsVUFDTEMsT0FBTztBQUFBLFVBQ1BDLFlBQVk7QUFBQSxVQUNaQyxVQUFVO0FBQUEsVUFDVkMsU0FBUyxDQUFDLC
tCQUErQixrQkFBa0Isd0NBQXdDLG1CQUFtQjtBQUFBLFFBQ3hIO0FBQUEsTUFBQyxJQUVIO0FBQUEsTUFDSjtBQUFBLFFBQ0VKLEtBQUs7QUFBQSxRQUNMQy
xPQUFPO0FBQUEsUUFDUEMsWUFBWTtBQUFBLFFBQ1pDLFVBQVU7QUFBQSxRQUNWQyxTQUFTLENBQUMsMkJBQTJCLDJCQUEyQixzQ0FBc0Msa0JBQWtCLGlDQU
FpQyxtQkFBbUI7QUFBQSxNQUM5SztBQUFBLE1BQ0E7QUFBQSxRQUNFSixLQUFLO0FBQUEsUUFDTEMsT0FBTztBQUFBLFFBQ1BDLFlBQVk7QUFBQSxRQUNaQy
xVQUFVO0FBQUEsUUFDVkMsU0FBUyxDQUFDLGlDQUFpQyx1QkFBdUIsNEJBQTRCLGNBQWMseUJBQXlCLHNCQUFzQjtBQUFBLE1BQzdKO0FBQUEsTUFDQTtBQU
FBLFFBQ0VKLEtBQUs7QUFBQSxRQUNMQyxPQUFPO0FBQUEsUUFDUEMsWUFBWTtBQUFBLFFBQ1pDLFVBQVU7QUFBQSxRQUNWQyxTQUFTLENBQUMsNEJBQTRCLG
tCQUFrQiwyQkFBMkIsdUJBQXVCLDRCQUE0QixzQkFBc0I7QUFBQSxNQUM5SjtBQUFBLElBQUM7QUFBQSxJQUVMLENBQUNULFdBQVc7QUFBQSxFQUNkO0FBRU
EsUUFBTVUsV0FBV3JJO0FBQUFBLElBQ2YsTUFDRTtBQUFBLE1BQ0UsRUFBRWdJLEtBQUssa0JBQWtCQyxPQUFPLGtCQUFrQkMsWUFBWSxvQkFBb0JFLFNBQV
MsQ0FBQyxzQ0FBc0Msc0NBQXNDLG9CQUFvQixnQkFBZ0IsZ0NBQWdDLEVBQUU7QUFBQSxNQUM5TyxFQUFFSixLQUFLLG9CQUFvQkMsT0FBTyx3QkFBd0JDLF
lBQVksY0FBY0UsU0FBUyxDQUFDLHFDQUFxQywyQkFBMkIsZ0JBQWdCLEVBQUU7QUFBQSxJQUFDO0FBQUEsSUFFckw7QUFBQSxFQUNGO0FBRUEsUUFBTUUsY0
FBY3RJLFFBQVFBLE1BQU07QUFDekMsVUFBTXVJLFdBQVdoQyxTQUFTYixTQUFTLElBQUloRSxLQUFLLEVBQUU4RyxZQUFZO0FBQzFELFFBQUlELFlBQVksT0
FBUSxRQUFPWixjQUFjLFNBQVM7QUFDdEQsUUFBSVksWUFBWSxhQUFjLFFBQU87QUFDckMsUUFBSUEsWUFBWSxPQUFRLFFBQU87QUFDL0IsUUFBSUEsWUFBWS
xXQUFXQSxZQUFZLE1BQU8sUUFBT0E7QUFDckQsV0FBTztBQUFBLEVBQ1QsR0FBRyxDQUFDWixhQUFhcEIsU0FBU2IsS0FBSyxDQUFDO0FBRWhDLFFBQU0rQy
xlQUFlNUIsb0JBQW9CeUI7QUFFekMsUUFBTUksMkJBQTJCMUksUUFBUSxNQUFNO0FBQzdDLFVBQU0wRSxNQUFNNkIsU0FBU29DO0FBQ3JCLFVBQU1DLElBQU
ksT0FBT2xFLFFBQVEsWUFBWWlCLE9BQU9DLFNBQVNsQixHQUFHLEtBQUtBLE1BQU0sSUFBSWpDLEtBQUtDLE1BQU1nQyxHQUFHLElBQUk7QUFDekYsV0FBT2
pDLEtBQUtvRyxJQUFJLEdBQUdwRyxLQUFLcUcsSUFBSSxLQUFLRixDQUFDLENBQUM7QUFBQSxFQUNyQyxHQUFHLENBQUNyQyxTQUFTb0MsbUJBQW1CLENBQU
M7QUFFakMsUUFBTUksY0FBYztBQUNwQixRQUFNQyxTQUFTO0FBQ2YsUUFBTUMsOEJBQThCakosUUFBUSxNQUFNO0FBQ2hELFdBQU95QyxLQUFLcUcsSUFBSU
UsUUFBUXZHLEtBQUtvRyxJQUFJRSxhQUFhTCx3QkFBd0IsQ0FBQztBQUFBLEVBQ3pFLEdBQUcsQ0FBQ0EsMEJBQTBCSyxhQUFhQyxNQUFNLENBQUM7QUFFbE
QsUUFBTUUsNkJBQTZCNUQsc0JBQXNCbUQsaUJBQWlCLFFBQVFRLDhCQUE4QlA7QUFFaEgzSSxZQUFVLE1BQU07QUFDZCxVQUFNb0osTUFBTSxZQUFZO0FBQ3
RCLFlBQU1DLFNBQVMvQyxTQUFTK0MsVUFBVTtBQUNsQyxVQUFJLENBQUNBLE9BQVE7QUFDYixZQUFNQyxTQUFTLElBQUlDLGdCQUFnQkYsTUFBTTtBQUN6Qy
xZQUFNRyxZQUFZRixPQUFPRyxJQUFJLFVBQVUsS0FBSyxJQUFJOUgsS0FBSyxFQUFFOEcsWUFBWTtBQUNuRSxVQUFJZSxhQUFhLGFBQWFBLGFBQWEsU0FBVT
tBQUVyRCxZQUFNbkUsUUFBUWlFLE9BQU9HLElBQUksTUFBTSxLQUFLSCxPQUFPRyxJQUFJLE9BQU8sS0FBSyxJQUFJOUgsS0FBSyxFQUFFOEcsWUFBWSxLQU
FLO0FBQ3ZGNUIsd0JBQWtCLEVBQUVKLE1BQU0rQyxVQUFVbkUsS0FBSyxDQUFDO0FBRTFDLFlBQU1XLGFBQWFzRCxPQUFPRyxJQUFJLFlBQVksS0FBSyxJQU
FJOUgsS0FBSztBQUN4RCxZQUFNK0gsdUJBQXVCSixPQUFPRyxJQUFJLFlBQVksS0FBSyxJQUFJOUgsS0FBSyxLQUFLO0FBRXZFLFlBQU1xRCxRQUFRLE1BQU
1xQixRQUFRO0FBQzVCLFlBQU1zRCxxQkFBcUIzRSxPQUFPeUIsU0FBUyxZQUFZekIsTUFBTTBCLFFBQVFDLEtBQUs7QUFDMUUsWUFBTWlELHFCQUFxQkQsc0
JBQXNCRDtBQUVqRCxVQUFJRixhQUFhLFdBQVc7QUFDMUIsWUFBSXhELFdBQVc7QUFDYixnQkFBTTZELE9BQU8sTUFBTTlELDZCQUE2QkMsV0FBVzRELGtCQU
FrQjtBQUM3RSxjQUFJLENBQUNDLEtBQUt6SSxJQUFJO0FBQ1pnRyxxQkFBU0Usb0JBQW9CdUMsS0FBS3hJLFFBQVF3SSxLQUFLMUksSUFBSSxDQUFDO0FBQU
EsVUFDdEQsT0FBTztBQUNMLGtCQUFNa0YsUUFBUTtBQUFBLFVBQ2hCO0FBQUEsUUFDRjtBQUVBLFlBQUl5RCxRQUFRO0FBQ1osZUFBT0EsUUFBUSxJQUFJO0
FBQ2pCLGdCQUFNLElBQUlDLFFBQVEsQ0FBQ0MsTUFBTUMsV0FBV0QsR0FBRyxHQUFJLENBQUM7QUFDNUMsZ0JBQU1FLE9BQU8sTUFBTTdELFFBQVE7QUFDM0
IsY0FBSTZELE1BQU16RCxTQUFTLGFBQWF5RCxLQUFLeEQsUUFBUW9CLHFCQUFxQixRQUFTO0FBQzNFZ0MsbUJBQVM7QUFBQSxRQUNYO0FBQUEsTUFDRjtBQU
VBdkQsZUFBUyxjQUFjLEVBQUUzRSxTQUFTLEtBQUssQ0FBQztBQUFBLElBQzFDO0FBQ0F3SCxRQUFJLEVBQUU1RixNQUFNLE1BQU1DLE1BQVM7QUFBQSxFQU
M3QixHQUFHLENBQUM2QyxTQUFTK0MsUUFBUTlDLFVBQVVGLE9BQU8sQ0FBQztBQUV2QyxRQUFNOEQsb0JBQW9CLE9BQU83RSxXQUEyQjtBQUMxRCxRQUFJLE
NBQUNGLFVBQVc7QUFDaEIsVUFBTWdGLE9BQU8xQjtBQUNiLFFBQUksQ0FBQzBCLFFBQVFBLFNBQVMsUUFBUTtBQUM1QmhELGVBQVMsNEJBQTRCO0FBQ3JDO0
FBQUEsSUFDRjtBQUVBLFVBQU1pRCxZQUFZRCxTQUFTLFFBQVExSCxLQUFLQyxNQUFNd0csOEJBQThCSCxXQUFXLElBQUk7QUFDM0YsUUFBSW9CLFNBQVMsU0
FBU0MsWUFBWXBCLFFBQVE7QUFDeENsQywwQkFBb0IsWUFBWTtBQUNoQ00sMkJBQXFCLElBQUk7QUFDekJELGVBQVMsMkRBQTJEO0FBQ3BFO0FBQUEsSUFDRj
tBQUVBLFVBQU1rRCxRQUFRRixTQUFTLFFBQVExSCxLQUFLb0csSUFBSUUsYUFBYXRHLEtBQUtxRyxJQUFJRSxRQUFRb0IsU0FBUyxDQUFDLElBQUk7QUFFcE
ZsRCx3QkFBb0IsSUFBSTtBQUN4QkMsYUFBUyxJQUFJO0FBQ2IsVUFBTXJELE1BQU0sTUFBTW9CLHdCQUF3QkMsV0FBV2dGLE1BQU05RSxRQUFRZ0YsS0FBSz
tBQUN4RSxRQUFJLENBQUN2RyxJQUFJM0MsSUFBSTtBQUNYZ0csZUFBU0Usb0JBQW9CdkQsSUFBSTFDLFFBQVEwQyxJQUFJNUMsSUFBSSxDQUFDO0FBQ2xEZ0
csMEJBQW9CLEtBQUs7QUFDekI7QUFBQSxJQUNGO0FBQ0EsVUFBTWhHLE9BQU80QyxJQUFJNUM7QUFDakIsVUFBTW9KLE1BQU0sT0FBT3BKLEtBQUtvSixRQU
FRLFdBQVdwSixLQUFLb0osTUFBTTtBQUN0RCxRQUFJLENBQUNBLEtBQUs7QUFDUm5ELGVBQVMsMkNBQTJDO0FBQ3BERCwwQkFBb0IsS0FBSztBQUN6QjtBQU
FBLElBQ0Y7QUFDQXFELFdBQU9sRSxTQUFTbUUsT0FBT0Y7QUFBQUEsRUFDekI7QUFFQSxRQUFNRyx1QkFBdUIsT0FBT3BGLFdBQTJCO0FBQzdELFFBQUksQ0
FBQ0YsYUFBYSxDQUFDNEIsZ0JBQWlCO0FBQ3BDRyx3QkFBb0IsSUFBSTtBQUN4QkMsYUFBUyxJQUFJO0FBQ2IsVUFBTXJELE1BQU0sTUFBTW9CLHdCQUF3Qk
MsV0FBVzRCLGlCQUFpQjFCLE1BQU07QUFDNUUsUUFBSSxDQUFDdkIsSUFBSTNDLElBQUk7QUFDWGdHLGVBQVNFLG9CQUFvQnZELElBQUkxQyxRQUFRMEMsSU
FBSTVDLElBQUksQ0FBQztBQUNsRGdHLDBCQUFvQixLQUFLO0FBQ3pCO0FBQUEsSUFDRjtBQUNBLFVBQU1oRyxPQUFPNEMsSUFBSTVDO0FBQ2pCLFVBQU1vSi
xNQUFNLE9BQU9wSixLQUFLb0osUUFBUSxXQUFXcEosS0FBS29KLE1BQU07QUFDdEQsUUFBSSxDQUFDQSxLQUFLO0FBQ1JuRCxlQUFTLDJDQUEyQztBQUNwRE
QsMEJBQW9CLEtBQUs7QUFDekI7QUFBQSxJQUNGO0FBQ0FxRCxXQUFPbEUsU0FBU21FLE9BQU9GO0FBQUFBLEVBQ3pCO0FBRUEsTUFBSSxDQUFDL0QsU0FBUz
tBQUNaLFdBQ0UsdUJBQUMsWUFDQyxpQ0FBQyxTQUFJLFdBQVUsa0JBQWlCLGdDQUFoQztBQUFBO0FBQUE7QUFBQTtBQUFBLFdBQWdELEtBRGxEO0FBQUE7QU
FBQTtBQUFBO0FBQUEsV0FFQTtBQUFBLEVBRUo7QUFFQSxRQUFNbUUsd0JBQXdCQSxDQUFDNUosVUFBa0I7QUFDL0MsVUFBTTZKLElBQUlwSixPQUFPVCxTQU
FTLEVBQUUsRUFBRVksS0FBSyxFQUFFOEcsWUFBWTtBQUNqRCxRQUFJLENBQUNtQyxFQUFHLFFBQU87QUFDZixRQUFJQSxNQUFNLFFBQVMsUUFBTztBQUMxQi
xRQUFJQSxNQUFNLFFBQVMsUUFBTztBQUMxQixRQUFJQSxNQUFNLGVBQWdCLFFBQU87QUFDakMsUUFBSUEsTUFBTSxXQUFZLFFBQU87QUFDN0IsUUFBSUEsTU
FBTSxZQUFhLFFBQU87QUFDOUIsV0FBTzdKO0FBQUFBLEVBQ1Q7QUFFQSxRQUFNOEosYUFBYXJFLFFBQVFzQixxQkFBcUIsaUJBQWlCLFFBQVF0QixRQUFRc0
IscUJBQXFCLFVBQVVBLFVBQVU7QUUxSCxRQUFNZ0QsYUFBYUEsQ0FBQ0MsYUFBcUI7QUFDdkMsVUFBTUMsSUFBSXhKLE9BQU91SixZQUFZLEVBQUUsRUFBRX
BKLEtBQUssRUFBRThHLFlBQVk7QUFDcEQsUUFBSXVDLE1BQU0sYUFBYyxRQUFPO0FBQy9CLFFBQUlBLE1BQU0sT0FBUSxRQUFPO0FBQ3pCLFFBQUlBLE1BQU
0sTUFBTyxRQUFPO0FBQ3hCLFFBQUlBLE1BQU0sUUFBUyxRQUFPO0FBQzFCLFFBQUlBLE1BQU0sT0FBUSxRQUFPO0FBQ3pCLFdBQU9EO0FBQUFBLEVBQ1Q7QU
FFQSxTQUNFLHVCQUFDLFlBQ0MsaUNBQUMsU0FBSSxXQUFVLGFBQ1puRTtBQUFBQSxxQkFDQztBQUFBLE1BQUM7QUFBQTtBQUFBLFFBQ0MsV0FBVztBQUFBLF
VBQ1Q7QUFBQSxVQUNBQSxlQUFlSCxTQUFTLFlBQVksc0RBQXNEO0FBQUEsUUFBNkMsRUFDdkl3RSxLQUFLLEdBQUc7QUFBQSxRQUVWLGlDQUFDLFNBQUksV0
FBVSxxREFDYjtBQUFBLGlDQUFDLFNBQUksV0FBVSxhQUNiO0FBQUEsbUNBQUMsU0FBSSxXQUFVLGlCQUFpQnJFLHlCQUFlSCxTQUFTLFlBQVksd0JBQXdCLH
lCQUE1RjtBQUFBO0FBQUE7QUFBQTtBQUFBLG1CQUFrSDtBQUFBLFlBQ2xILHVCQUFDLFNBQ0VHO0FBQUFBLDZCQUFldkIsT0FBTyxTQUFTdUIsZUFBZXZCLE
tBQUs2RixZQUFZLENBQUMsT0FBTztBQUFBLGNBQUc7QUFBQSxjQUFTUCxzQkFBc0JuRSxRQUFRc0IsZ0JBQWdCO0FBQUEsY0FBRTtBQUFBLGlCQUR0STtBQU
FBO0FBQUE7QUFBQTtBQUFBLG1CQUVBO0FBQUEsZUFKRjtBQUFBO0FBQUE7QUFBQTtBQUFBLGlCQUtBO0FBQUEsVUFDQSx1QkFBQyxTQUFJLFdBQVUsMkJBQ2
```

```
I7QUFBQSxtQ0FBQyxVQUFPLFNBQVEsYUFBWSxTQUFTLE1BQU0sS0FBS3pCLFFBQVEsR0FBRSx5QkFBMUQ7QUFBQTtBQUFBO0FBQUE7QUFBQSxtQkFFQTtBQU
FBLFlBQ0EsdUJBQUMsVUFBTyxTQUFRLGFBQVksU0FBUyxNQUFNUSxrQkFBa0IsSUFBSSxHQUFFLHNCQUFuRTtBQUFBO0FBQUFBO0FBQUE7QUFBQTtBQUFBLG1CQUVBO
FBQUEsZUFORjtBQUFBO0FBQUE7QUFBQTtBQUFBLGlCQU9BO0FBQUEsYUFkRjtBQUFBO0FBQUE7QUFBQTtBQUFBLGVBZUE7QUFBQTtBQUFBLE1BckJGO0FBQU
E7QUFBQTtBQUFBO0FBQUE7QUFBQTtBQUFBO0FBQUE7QUFBQSxJQXNCQSxJQUNFO0FBQUEsSUFFSHZGLFFBQVEsdUJBQUMsU0FBSSxXQUFVLDBFQUEwRUsbU
JBQXpGO0FBQUE7QUFBQTtBQUFBO0FBQUEsV0FBK0YsSUFBUztBQUFBLElElBRWpILHVCQUFDLFFBQ0MsaUNBQUMsU0FBSSxXQUFVLGlCQUNiO0FBQUEsNkJBQU
MsU0FBSSxXQUFVLG9EQUNiO0FBQUEsK0JBQUMsU0FDQztBQUFBLGlDQUFDLFNBQUksV0FBVSx3Q0FBdUMsUXREO0FBQUE7QUFBQTtBQUFBO0FBQUEsaU
JBQStEO0FBQUEsVUFDL0QsdUJBQUMsU0FBSSxXQUFVLDBCQUF3QjtBQUFBO0FBQUEsWUFDdkJ3SixXQUFXdEUsUUFBUWIsS0FBSztBQUFBLGVBQXhDO0FBQUE
E7QUFBQTtBQUFBO0FBQUEsaUJBRUE7QUFBQSxhQUpGO0FBQUE7QUFBQTtBQUFBO0FBQUEsZUFLQTtBQUFBLFFBQ0EsdUJBQUMsU0FBSSxXQUFVLDJCQUNiO0
FBQUEsaUNBQUMsU0FBSSxXQUFVLFNNQUFNYSxRQUFRMkUsUUFBUSxVQUFVLE9BQVEzRSxrQkFBUTJFLFFBQVEsaUJBQWlCLG9CQUFoRjtBQUFBO0FBQUFBO0FBQU
FBLGlCQUFpRztBQUFBLFVBQ2pHLHVCQUFDLFNBQU0sTUFBTSU4sWUFBWWTtBQUFBO0FBQUFBO0FBQUEsWUFBWUYsc0JBQXNCbkUsUUFBUXNCLGdCQUFnQjtBQUFBLGVBQW
5GO0FBQUE7QUFBQTtBQUFBO0FBQUEsaUJBQXFGO0FBQUEsVUFDckYsdUJBQUMsVUFBTyxTQUFRLGFBQVksU0FBUyxNQUFNLEtBQUt6QixRQUFRLEdBQUUsZ0
NBQTFEO0FBQUE7QUFBQTtBQUFBO0FBQUEsaUJBQUE7QUFBQSxhQUxGO0FBQUE7QUFBQTtBQUFBO0FBQUEsZUFNQTtBQUFBLFFBQ0EsdUJBQUMsU0FBSSxXQU
E7QUFBQSxhQWNBO0FBQUEsTUFFQSx1QkFBQyxTQUFJLFdBQVUsd0RBQ0oyQixnQkFBTW9ELElBQUksQ0FBQ0osTUFBTUksQ0FBQ0osTUFBTUNpdjQ7QUFBQ1NyXXM0MsaU
JBQWlCc0MsUkFBRS9DO0FBQ3BDLGNBQU1xRCxZQUFZLEFBQUFFL0MsUUFBUTNhRkUsbmdhcG9MkUsUUFBUTNhRkUsbOEjuqeQicmNjUXF0Q0ZOVnBQOQ9hG4EHNtBQUlFFBQVE7QUFDdkIsZUFDdEQ
FBLFVBQUM7QUFBQTtBQUFBLFlBRVdUNLU9FlBSztBQUFBLFlBQ0wsS0FBS0NVeRW4RCxLCCvC5kwXsgXV0UEVSQQUFBUXRQUFHTmxXQVBVFRXRQ0ZBQVQxQ
FBTzFLO0FBQ3JCLG9CQUFNOEgsSUFBSWxFLElBQUloRCxLQUFLLE1BQU0sS0FBS3hILLGNBQWNwRCxxQkFBQ3FRQ2RDZFRXXFgRCxPQUFDQVJRZaEbNQlVSjBkRPxBTWs9QR09ZLCBUQTZ0ZjdVUhd
FBTkqlDE0tBFQ1Mb3I5dDWHIrbsqPbM1fCF1VTVQQWrGVVBMa1TSXFRPQk5jU9UZ1cFzTT1U1nMtTYXcKxQ9AsZXhlPpQK1T4wN0x1nNWbxBtSV1e3TUQU0QUVQmdCbC
ZSI1vhKGNEJmVF5mVVEzlu5zZBKMUuTDlRgjR2lQUUJYZHcRXRCd2NOaHZvbVpZW3JTQUQzR0FCRUF4QXUFbyBDlZuN4lXQXJ4bRnNRoMk01RUJDUxmMSZXFUE8zRVUxUAYR
FBQSx5QkFBa0IsSUFBSSxHQUFFLHNCQUFuRTtBQUFBO0FBQUFBO0FBQUE7QUFBQTtBQUFBLG1CQUVBO
FBLFlBQ0EsdUJBQUMsVUFBTyxTQUFRLGFBQVksU0FBUyxNQUFNUSxrQkFBa0IsSUFBSSxHQUFFLHNCQUFuRTtBQUFBO
```

IsInVzZXJTZWxlY3RlZFBsYW4iLCJzZXRVc2VyU2VsZWN0ZWRQbGFuIiwic2VsZWN0ZWRTZXJ2aWNlIiwic2V0U2VsZWN0ZWRTZXJ2aWNlIiwiY3JlYXRpbm
dDaGVja291dCIsInNldENyZWF0aW5nQ2hlY2tvdXQiLCJzZXRFcnJvciIsInNldEZ1bmNpb25hcmlvc1RvdGFsIiwiZm9ybWF0Q2hlY2tvdXRFcnJvciIsIm
9iaiIsImVyciIsInN0cmlwZVN0YXR1cyIsInN0cmlwZV9zdGF0dXMiLCJhc0pzb24iLCJjYW5TaG93RnJlZSIsIkJvb2xlYW4iLCJzdGF0dXNfcGFnYW1lbn
RvIiwiZnJlZV90cmlhbF9jb25zdW1pZG8iLCJwbGFucyIsImtleSIsInRpdGxlIiwicHJpY2VMYWJlbCIsInN1YnRpdGxlIiwiYnVsbGV0cyIsInNlcnZpY2
VzIiwiY3VycmVudFBsYW4iLCJjdXJyZW50IiwidG9Mb3dlckNhc2UiLCJzZWxlY3RlZFBsYW4iLCJkZWZhdWx0RnVuY2lvbmFyaW9zVG90YWwiLCJsaW1pdG
VfZnVuY2lvbmFyaW9zIiwibiIsIm1heCIsIm1pbiIsImluY2x1ZGVkUHJvIiwibWF4UHJvIiwiZGVmYXVsdFByb0Z1bmNpb25hcmlvc1RvdGFsIiwiZWZmZW
N0aXZlRnVuY2lvbmFyaW9zVG90YWwiLCJydW4iLCJzZWFyY2giLCJwYXJhbXMiLCJVUkxTZWFyY2hQYXJhbXMiLCJjaGVja291dCIsImdldCIsInVzdWFyaW
9JZEZyb21QYXJhbXMiLCJyZWZyZXNoZWRVc3VhcmlvSWQiLCJlZmZlY3RpdmVVc3VhcmlvSWQiLCJzeW5jIiwidHJpZXMiLCJQcm9taXNlIiwiciIsInNldF
RpbWVvdXQiLCJuZXh0Iiwic3RhcnRQbGFuQ2hlY2tvdXQiLCJwbGFuIiwicmVxdWVzdGVkIiwidG90YWwiLCJ1cmwiLCJ3aW5kb3ciLCJocmVmIiwic3Rhcn
RTZXJ2aWNlQ2hlY2tvdXQiLCJmb3JtYXRTdGF0dXNQYWdhbWVudG8iLCJ2Iiwic3RhdHVzVG9uZSIsInBsYW5vTGFiZWwiLCJwbGFub1JhdyIsInAiLCJqb2
luIiwidG9VcHBlckNhc2UiLCJhdGl2byIsIm1hcCIsInNlbGVjdGVkIiwiY2xpY2thYmxlIiwiYmVzdCIsImIiLCJ0YXJnZXQiLCJpIiwiY2xhbXBlZCIsIn
MiLCJfYyIsIiRSZWZyZXNoUmVnJCJdLCJpZ25vcmVMaXN0IjpbXSwic291cmNlcyI6WyJQYWdhbWVudG9QYWdlLnRzeCJdLCJzb3VyY2VzQ29udGVudCI6Wy
JpbXBvcnQgeyB1c2VFZmZlY3QsIHVzZU1lbW8sIHVzZVN0YXRlIH0gZnJvbSAncmVhY3QnXG5pbXBvcnQgeyB1c2VMb2NhdGlvbiwgdXNlTmF2aWdhdGUgfS
Bmcm9tICdyZWFjdC1yb3V0ZXItZG9tJ1xuaW1wb3J0IHsgQXBwU2hlbGwgfSBmcm9tICcuLi8uLi9jb21wb25lbnRzL2xheW91dC9BcHBTaGVsbCdcbmltcG
9ydCB7IEJhZGdlIH0gZnJvbSAnLi4vLi4vY29tcG9uZW50cy91aS9CYWRnZSdcbmltcG9ydCB7IEJ1dHRvbiB9IGZyb20gJy4uLy4uL2NvbXBvbmVudHMvdW
kvQnV0dG9uJ1xuaW1wb3J0IHsgQ2FyZCB9IGZyb20gJy4uLy4uL2NvbXBvbmVudHMvdWkvQ2FyZCdcbmltcG9ydCB7IElucHV0IH0gZnJvbSAnLi4vLi4vY2
9tcG9uZW50cy91aS9JbnB1dCdcbmltcG9ydCB7IGNoZWNrSnd0UHJvamVjdCwgc3VwYWJhc2UsIHN1cGFiYXNlRW52IH0gZnJvbSAnLi4vLi4vbGliL3N1cG
FiYXNlJ1xuaW1wb3J0IHsgdXNlQXV0aCB9IGZyb20gJy4uLy4uL3N0YXRlL2F1dGgvdXNlQXV0aCdcblxudHlwZSBGblJlc3VsdCA9IHsgb2s6IHRydWU7IH
N0YXR1czogbnVtYmVyOyBib2R5OiB1bmtub3duIH0gfCB7IG9rOiBmYWxzZTsgc3RhdHVzOiBudW1iZXI7IGJvZHk6IHVua25vd24gfVxuXG5mdW5jdGlvbi
BzYWZlSnNvbih2YWx1ZTogdW5rbm93bikge1xuICB0cnkge1xuICAgIHJldHVybiBKU09OLnN0cmluZ2lmeSh2YWx1ZSlcbiAgfSBjYXRjaCB7XG4gICAgcm
V0dXJuIG51bGxcbiAgfVxufVxuXG5hc3luYyBmdW5jdGlvbiBjYWxsUGF5bWVudHNGbihib2R5OiBSZWNvcmQ8c3RyaW5nLCB1bmtub3duPik6IFByb21pc2
U8Rm5SZXN1bHQ+IHtcbiAgaWYgKCFzdXBhYmFzZUVudi5vaykge1xuICAgIHJldHVybiB7IG9rOiBmYWxzZSBhcyBjb25zdCwgc3RhdHVzOiAwLCBib2R5Oi
B7IGVycm9yOiAnbWlzc2luZ19zdXBhYmFzZV9lbnYnIH0gfVxuICB9XG5cbiAgY29uc3Qgc3VwYWJhc2VVcmwgPSBTdHJpbmcoc3VwYWJhc2VFbnYudmFsdW
VzLlZJVEVfU1VQQUJBU0VfVVJMID8/ICcnKVxuICAgIC50cmltKClcbiAgICAucmVwbGFjZSgvXlsnXCJgXFxzXSt8WydcImBcXHNdKyQvZywgJycpXG4gIC
AgLnJlcGxhY2UoL1xcLyskL2csICcnKVxuICBjb25zdCBzdXBhYmFzZUFub25LZXkgPSBTdHJpbmcoc3VwYWJhc2VFbnYudmFsdWVzLlZJVEVfU1VQQUJBU0
VfQU5PTl9LRVkgPz8gJycpXG4gICAgLnRyaW0oKVxuICAgIC5yZXBsYWNlKC9eWydcImBcXHNdK3xbJ1wiYFxcc10rJC9nLCAnJylcbiAgY29uc3QgZm5Vcm
wgPSBgJHtzdXBhYmFzZVVybH0vZnVuY3Rpb25zL3YxL3BheW1lbnRzYFxuXG4gIGNvbnN0IHRyeVJlZnJlc2ggPSBhc3luYyAoKSA9PiB7XG4gICAgY29uc3
QgeyBkYXRhOiByZWZyZXNoZWQsIGVycm9yOiByZWZyZXNoRXJyIH0gPSBhd2FpdCBzdXBhYmFzZS5hdXRoLnJlZnJlc2hTZXNzaW9uKClcbiAgICBpZiAocm
VmcmVzaEVycikgcmV0dXJuIG51bGxcbiAgICByZXR1cm4gcmVmcmVzaGVkLnNlc3Npb24gPz8gbnVsbFxuICB9XG5cbiAgY29uc3QgeyBkYXRhOiBzZXNzaW
9uRGF0YSB9ID0gYXdhaXQgc3VwYWJhc2UuYXV0aC5nZXRTZXNzaW9uKClcbiAgbGV0IHNlc3Npb24gPSBzZXNzaW9uRGF0YS5zZXNzaW9uXG4gIGNvbnN0IG
5vdyA9IE1hdGguZmxvb3IoRGF0ZS5ub3coKSAvIDEwMDApXG4gIGNvbnN0IGV4cGlyZXNBdCA9IHNlc3Npb24/LmV4cGlyZXNfYXQgPz8gbnVsbFxuXG4gIG
lmIChzZXNzaW9uICYmICghZXhwaXJlc0F0IHx8IGV4cGlyZXNBdCA8PSBub3cgKyA2MCkpIHtcbiAgICBjb25zdCByZWZyZXNoZWQgPSBhd2FpdCB0cnlSZW
ZyZXNoKClcbiAgICBpZiAocmVmcmVzaGVkKSBzZXNzaW9uID0gcmVmcmVzaGVkXG4gIH1cblxuICBpZiAoc2Vzc2lvbikge1xuICAgIGNvbnN0IHsgZXJyb3
I6IHVzZXJFcnIgfSA9IGF3YWl0IHN1cGFiYXNlLmF1dGguZ2V0VXNlcigpXG4gICAgY29uc3QgdXNlckVyck1zZyA9IHR5cGVvZiB1c2VyRXJyPy5tZXNzYW
dlID09PSAnc3RyaW5nJyA/IHVzZXJFcnIubWVzc2FnZSA6ICcnXG4gICAgaWYgKHVzZXJFcnIgJiYgL2ludmFsaWRcXHMrand0L2kudGVzdCh1c2VyRXJyTX
NnKSkge1xuICAgICAgY29uc3QgcmVmcmVzaGVkID0gYXdhaXQgdHJ5UmVmcmVzaCgpXG4gICAgICBpZiAocmVmcmVzaGVkKSBzZXNzaW9uID0gcmVmcmVzaG
VkXG4gICAgfVxuICB9XG5cbiAgY29uc3QgdG9rZW4gPSBzZXNzaW9uPy5hY2Nlc3NfdG9rZW4gPz8gbnVsbFxuICBpZiAoIXRva2VuKSB7XG4gICAgcmV0dX
JuIHsgb2s6IGZhbHNlIGFzIGNvbnN0LCBzdGF0dXM6IDQwMSwgYm9keTogeyBlcnJvcjogJ3Nlc3Npb25fZXhwaXJlZCcgfSB9XG4gIH1cblxuICBjb25zdC
B0b2tlblByb2plY3QgPSBjaGVja0p3dFByb2plY3QodG9rZW4sIHN1cGFiYXNlVXJsKVxuICBpZiAoIXRva2VuUHJvamVjdC5vaykge1xuICAgIGF3YWl0IH
N1cGFiYXNlLmF1dGguc2lnbk91dCgpLmNhdGNoKCgpID0+IHVuZGVmaW5lZClcbiAgICByZXR1cm4geyBvazogZmFsc2UgYXMgY29uc3QsIHN0YXR1czogND
AxLCBib2R5OiB7IGVycm9yOiAnand0X3Byb2plY3RfbWlzbWF0Y2gnLCBpc3M6IHRva2VuUHJvamVjdC5pc3MsIGV4cGVjdGVkOiB0b2tlblByb2plY3QuZX
hwZWN0ZWRQcmVmaXggfSB9XG4gIH1cblxuICBjb25zdCBjYWxsRmV0Y2ggPSBhc3luYyAoand0OiBzdHJpbmcpID0+IHtcbiAgICBsZXQgcmVzOiBSZXNwb2
5zZVxuICAgIHRyeSB7XG4gICAgICByZXMgPSBhd2FpdCBmZXRjaChmblVybCwge1xuICAgICAgICBtZXRob2Q6ICdQT1NUJyxcbiAgICAgICAgaGVhZGVycz
oge1xuICAgICAgICAgICdDb250ZW50LVR5cGUnOiAnYXBwbGljYXRpb24vanNvbicsXG4gICAgICAgICAgYXBpa2V5OiBzdXBhYmFzZUFub25LZXksXG4gIC
AgICAgICAgQXV0aG9yaXphdGlvbjogYEJlYXJlciAke2p3dH1gLFxuICAgICAgICB9LFxuICAgICAgICBib2R5OiBKU09OLnN0cmluZ2lmeShib2R5KSxcbi
AgICAgIH0pXG4gICAgfSBjYXRjaCAoZTogdW5rbm93bikge1xuICAgICAgY29uc3QgbXNnID0gZSBpbnN0YW5jZW9mIEVycm9yID8gZS5tZXNzYWdlIDogJ0
ZhbGhhIGRlIHJlZGUnXG4gICAgICByZXR1cm4geyBvazogZmFsc2UgYXMgY29uc3QsIHN0YXR1czogMCwgYm9keTogeyBlcnJvcjogJ25ldHdvcmtfZXJyb3
InLCBtZXNzYWdlOiBtc2cgfSB9XG4gICAgfVxuXG4gICAgY29uc3QgdGV4dCA9IGF3YWl0IHJlcy50ZXh0KClcbiAgICBsZXQgcGFyc2VkOiB1bmtub3duID
0gbnVsbFxuICAgIHRyeSB7XG4gICAgICBwYXJzZWQgPSB0ZXh0ID8gSlNPTi5wYXJzZSh0ZXh0KSA6IG51bGxcbiAgICB9IGNhdGNoIHtcbiAgICAgIHBhcn
NlZCA9IHRleHRcbiAgICB9XG5cbiAgICBpZiAoIXJlcy5vayAmJiByZXMuc3RhdHVzID09PSA0MDQpIHtcbiAgICAgIGNvbnN0IHJhdyA9IHR5cGVvZiBwYX
JzZWQgPT09ICdzdHJpbmcnID8gcGFyc2VkIDogdGV4dFxuICAgICAgaWYgKHR5cGVvZiByYXcgPT09ICdzdHJpbmcnICYmIHJhdy5pbmNsdWRlcygnUmVxdW
VzdGVkIGZ1bmN0aW9uIHdhcyBub3QgZm91bmQnKSkge1xuICAgICAgICByZXR1cm4ge1xuICAgICAgICAgIG9rOiBmYWxzZSBhcyBjb25zdCxcbiAgICAgIC
AgICBzdGF0dXM6IDQwNCxcbiAgICAgICAgICBib2R5OiB7XG4gICAgICAgICAgICBlcnJvcjogJ2Z1bmN0aW9uX25vdF9kZXBsb3llZCcsXG4gICAgICAgIC
AgICBtZXNzYWdlOiAnQSBmdW7Dp8OjbyBwYXltZW50cyBuw6NvIGVzdMOhIHB1YmxpY2FkYSBubyBTdXBhYmFzZS4gRmHDp2EgZGVwbG95IGRhIEVkZ2UgRn
VuY3Rpb24gYHBheW1lbnRzYCBubyBzZXUgcHJvamV0by4nLFxuICAgICAgICAgIH0sXG4gICAgICAgIH1cbiAgICAgIH1cbiAgICB9XG5cbiAgICBpZiAoXG
4gICAgICAhcmVzLm9rICYmXG4gICAgICByZXMuc3RhdHVzID09PSA0MDEgJiZcbiAgICAgIHBhcnNlZCAmJlxuICAgICAgdHlwZW9mIHBhcnNlZCA9PT0gJ2
9iamVjdCcgJiZcbiAgICAgIChwYXJzZWQgYXMgUmVjb3JkPHN0cmluZywgdW5rbm93bj4pLm1lc3NhZ2UgPT09ICdJbnZhbGlkIEpXVCcgJiZcbiAgICAgIC
hwYXJzZWQgYXMgUmVjb3JkPHN0cmluZywgdW5rbm93bj4pLmNvZGUgPT09IDQwMVxuICAgICkge1xuICAgICAgcmV0dXJuIHsgb2s6IGZhbHNlIGFzIGNvbn
N0LCBzdGF0dXM6IDQwMSwgYm9keTogeyBlcnJvcjogJ3N1cGFiYXNlX2dhdGV3YXlfaW52YWxpZF9qd3QnIH0gfVxuICAgIH1cblxuICAgIGlmICghcmVzLm
9rKSBjb25zb2xlLmVycm9yKCdwYXltZW50cyBlcnJvcicsIHsgc3RhdHVzOiByZXMuc3RhdHVzLCBib2R5OiBwYXJzZWQsIGJvZHlfanNvbjogc2FmZUpzb2
4ocGFyc2VkKSB9KVxuXG4gICAgaWYgKCFyZXMub2spIHJldHVybiB7IG9rOiBmYWxzZSBhcyBjb25zdCwgc3RhdHVzOiByZXMuc3RhdHVzLCBib2R5OiBwYX
JzZWQgfVxuICAgIHJldHVybiB7IG9rOiB0cnVlIGFzIGNvbnN0LCBzdGF0dXM6IHJlcy5zdGF0dXMsIGJvZHk6IHBhcnNlZCB9XG4gIH1cblxuICBjb25zdC
BmaXJzdCA9IGF3YWl0IGNhbGxGZXRjaCh0b2tlbilcbiAgaWYgKCFmaXJzdC5vayAmJiBmaXJzdC5zdGF0dXMgPT09IDQwMSkge1xuICAgIGNvbnN0IHJlZn
Jlc2hlZCA9IGF3YWl0IHRyeVJlZnJlc2goKVxuICAgIGNvbnN0IG5leHRUb2tlbiA9IHJlZnJlc2hlZD8uYWNjZXNzX3Rva2VuID8/IG51bGxcbiAgICBpZi
AoIW5leHRUb2tlbikge1xuICAgICAgYXdhaXQgc3VwYWJhc2UuYXV0aC5zaWduT3V0KCkuY2F0Y2goKCkgPT4gdW5kZWZpbmVkKVxuICAgICAgcmV0dXJuIH
sgb2s6IGZhbHNlIGFzIGNvbnN0LCBzdGF0dXM6IDQwMSwgYm9keTogeyBlcnJvcjogJ2ludmFsaWRfand0JyB9IH1cbiAgICB9XG4gICAgY29uc3QgbmV4dF
Byb2plY3QgPSBjaGVja0p3dFByb2plY3QobmV4dFRva2VuLCBzdXBhYmFzZVVybClcbiAgICBpZiAoIW5leHRQcm9qZWN0Lm9rKSB7XG4gICAgICBhd2FpdC
BzdXBhYmFzZS5hdXRoLnNpZ25PdXQoKS5jYXRjaCgoKSA9PiB1bmRlZmluZWQpXG4gICAgICByZXR1cm4geyBvazogZmFsc2UgYXMgY29uc3QsIHN0YXR1cz
ogNDAxLCBib2R5OiB7IGVycm9yOiAnand0X3Byb2plY3RfbWlzbWF0Y2gnLCBpc3M6IG5leHRQcm9qZWN0LmlzcywgZXhwZWN0ZWQ6IG5leHRQcm9qZWN0Lm
V4cGVjdGVkUHJlZml4IH0gfVxuICAgIH1cbiAgICByZXR1cm4gY2FsbEZldGNoKG5leHRUb2tlbilcbiAgfVxuXG4gIHJldHVybiBmaXJzdFxufVxuXG5hc3
luYyBmdW5jdGlvbiBjcmVhdGVDaGVja291dFBhZ2FtZW50byhcbiAgdXN1YXJpb0lkOiBzdHJpbmcsXG4gIGl0ZW06IHN0cmluZyxcbiAgbWV0b2RvOiAnY2
FyZCcgfCAncGl4JyxcbiAgZnVuY2lvbmFyaW9zVG90YWw/OiBudW1iZXIgfCBudWxsXG4pOiBQcm9taXNlPEZuUmVzdWx0PiB7XG4gIGNvbnN0IHBheWxvYW

```
Q6IFJlY29yZDxzdHJpbmcsIHVua25vd24+ID0geyBhY3Rpb246ICdjcmVhdGVfY2hlY2tvdXQnLCB1c3VhcmlvX2lkOiB1c3VhcmlvSWQsIHBsYW5vOiBpdG
VtLCBtZXRvZG8gfVxuICBpZiAodHlwZW9mIGZ1bmNpb25hcmlvc1RvdGFsID09PSAnbnVtYmVyJyAmJiBOdW1iZXIuaXNGaW5pdGUoZnVuY2lvbmFyaW9zVG
90YWwpKSBwYXlsb2FkLmZ1bmNpb25hcmlvc190b3RhbCA9IGZ1bmNpb25hcmlvc1RvdGFsXG4gIHJldHVybiBjYWxsUGF5bWVudHNGbihwYXlsb2FkKVxufV
xuXG5hc3luYyBmdW5jdGlvbiBzeW5jQ2hlY2tvdXRTZXNzaW9uUGFnYW1lbnRvKHNlc3Npb25JZDogc3RyaW5nLCB1c3VhcmlvSWQ6IHN0cmluZyB8IG51bG
wpOiBQcm9taXNlPEZuUmVzdWx0PiB7XG4gIGNvbnN0IHBheWxvYWQ6IFJlY29yZDxzdHJpbmcsIHVua25vd24+ID0geyBhY3Rpb246ICdzeW5jX2NoZWNrb3
V0X3Nlc3Npb24nLCBzZXNzaW9uX2lkOiBzZXNzaW9uSWQgfVxuICBpZiAodXN1YXJpb0lkKSBwYXlsb2FkLnVzdWFyaW9faWQgPSB1c3VhcmlvSWRcbiAgcm
V0dXJuIGNhbGxQYXltZW50c0ZuKHBheWxvYWQpXG59XG5cbnR5cGUgUGxhbktleSA9ICdmcmVlJyB8ICdiYXNpYycgfCAncHJvJyB8ICd0ZWFtJyB8ICdlbn
RlcnByaXNlJ1xuXG50eXBlIFBsYW5DYXJkID0ge1xuICBrZXk6IFBsYW5LZXlcbiAgdGl0bGU6IHN0cmluZ1xuICBwcmljZUxhYmVsOiBzdHJpbmdcbiAgc3
VidGl0bGU6IHN0cmluZ1xuICBidWxsZXRzOiBzdHJpbmdbXVxufVxuXG50eXBlIFNlcnZpY2VDYXJkID0ge1xuICBrZXk6ICdzZXR1cF9jb21wbGV0bycgfC
AnY29uc3VsdG9yaWFfaG9yYSdcbiAgdGl0bGU6IHN0cmluZ1xuICBwcmljZUxhYmVsOiBzdHJpbmdcbiAgYnVsbGV0czogc3RyaW5nW11cbn1cblxuZXhwb3
J0IGZ1bmN0aW9uIFBhZ2FtZW50b1BhZ2UoKSB7XG4gIGNvbnN0IHsgYXBwUHJpbmNpcGFsLCByZWZyZXNoIH0gPSB1c2VBdXRoKClcbiAgY29uc3QgbG9jYX
Rpb24gPSB1c2VMb2NhdGlvbigpXG4gIGNvbnN0IG5hdmlnYXRlID0gdXNlTmF2aWdhdGUoKVxuICBjb25zdCB1c3VhcmlvID0gYXBwUHJpbmNpcGFsPy5raW
5kID09PSAndXN1YXJpbycgPyBhcHBQcmluY2lwYWwucHJvZmlsZSA6IG51bGxcbiAgY29uc3QgdXN1YXJpb0lkID0gdXN1YXJpbz8uaWQgPz8gbnVsbFxuXG
4gIGNvbnN0IFtjaGVja291dE5vdGljZSwgc2V0Q2hlY2tvdXROb3RpY2VdID0gdXNlU3RhdGU8bnVsbCB8IHsga2luZDogJ3N1Y2Nlc3MnIHwgJ2NhbmNlbC
c7IGl0ZW06IHN0cmluZyB8IG51bGwgfT4obnVsbClcbiAgY29uc3QgW3VzZXJTZWxlY3RlZFBsYW4sIHNldFVzZXJTZWxlY3RlZFBsYW5dID0gdXNlU3RhdG
U8UGxhbktleSB8IG51bGw+KG51bGwpXG4gIGNvbnN0IFtzZWxlY3RlZFNlcnZpY2UsIHNldFNlbGVjdGVkU2VydmljZV0gPSB1c2VTdGF0ZTxTZXJ2aWNlQ2
FyZFsna2V5J10gfCBudWxsPihudWxsKVxuICBjb25zdCBbY3JlYXRpbmdDaGVja291dCwgc2V0Q3JlYXRpbmdDaGVja291dF0gPSB1c2VTdGF0ZShmYWxzZS
lcbiAgY29uc3QgW2Vycm9yLCBzZXRFcnJvcl0gPSB1c2VTdGF0ZTxzdHJpbmcgfCBudWxsPihudWxsKVxuICBjb25zdCBbZnVuY2lvbmFyaW9zVG90YWwsIH
NldEZ1bmNpb25hcmlvc1RvdGFsXSA9IHVzZVN0YXRlPG51bWJlciB8IG51bGw+KG51bGwpXG5cbiAgY29uc3QgZm9ybWF0Q2hlY2tvdXRFcnJvciA9IChzdG
F0dXM6IG51bWJlciwgYm9keTogdW5rbm93bikgPT4ge1xuICAgIGlmICh0eXBlb2YgYm9keSA9PT0gJ3N0cmluZycgJiYgYm9keS50cmltKCkpIHJldHVybi
Bib2R5XG4gICAgaWYgKGJvZHkgJiYgdHlwZW9mIGJvZHkgPT09ICdvYmplY3QnKSB7XG4gICAgICBjb25zdCBvYmogPSBib2R5IGFzIFJlY29yZDxzdHJpbm
csIHVua25vd24+XG4gICAgICBjb25zdCBlcnIgPSB0eXBlb2Ygb2JqLmVycm9yID09PSAnc3RyaW5nJyA/IG9iai5lcnJvciA6IG51bGxcbiAgICAgIGNvbn
N0IG1lc3NhZ2UgPSB0eXBlb2Ygb2JqLm1lc3NhZ2UgPT09ICdzdHJpbmcnICYmIG9iai5tZXNzYWdlLnRyaW0oKSA/IG9iai5tZXNzYWdlLnRyaW0oKSA6IG
51bGxcbiAgICAgIGlmIChtZXNzYWdlKSB7XG4gICAgICAgIGNvbnN0IHN0cmlwZVN0YXR1cyA9IHR5cGVvZiBvYmouc3RyaXBlX3N0YXR1cyA9PT0gJ251bW
JlcicgJiYgTnVtYmVyLmlzRmluaXRlKG9iai5zdHJpcGVfc3RhdHVzKSA/IG9iai5zdHJpcGVfc3RhdHVzIDogbnVsbFxuICAgICAgICBpZiAoZXJyID09PS
Anc3RyaXBlX2Vycm9yJyAmJiBzdHJpcGVTdGF0dXMpIHJldHVybiBgU3RyaXBlIChIVFRQICR7c3RyaXBlU3RhdHVzfSk6ICR7bWVzc2FnZX1gXG4gICAgIC
AgIHJldHVybiBtZXNzYWdlXG4gICAgICB9XG4gICAgICBpZiAoZXJyID09PSAnbWlzc2luZ19zdXBhYmFzZV9lbnYnKSByZXR1cm4gJ0NvbmZpZ3VyYCOnw6
NvIGRvIFN1cGFiYXNlIGF1c2VudGUgbm8gYW1iaWVudGUuJ1xuICAgICAgaWYgKGVyciA9PT0gJ25ldHdvcmtfZXJyb3InKSByZXR1cm4gdHlwZW9mIG9iai
5tZXNzYWdlID09PSAnc3RyaW5nJyAmJiBvYmoubWVzc2FnZS50cmltKCkgPyBvYmoubWVzc2FnZSA6ICdGYWxoYSBkZSByZWRlIGFvIGluaWNpYXIgcGFnYW
1lbnRvLidcbiAgICAgIGlmIChlcnIgPT09ICdzZXNzaW9uX2V4cGlyZWQnIHx8IGVyciA9PT0gJ2ludmFsaWRfand0JykgcmV0dXJuICdTZXNzw6NvIGV4cG
lyYWRhIG5vIFN1cGFiYXNlLiBTYWlhIGUgZW50cmUgbm92YW1lbnRlLidcbiAgICAgIGlmIChlcnIgPT09ICdqd3RfcHJvamVjdF9taXNtYXRjaCcpIHJldH
VybiAnU2Vzc8OjbyBkbyBTdXBhYmFzZSBwZXJ0ZW5jZSBhIG91dHJvIHByb2pldG8uIFNhaWEgZSBlbnRyZSBub3ZhbWVudGUuJ1xuICAgICAgaWYgKGVyci
A9PT0gJ3N1cGFiYXNlX2dhdGV3YXlfaW52YWxpZF9qd3QnKSByZXR1cm4gJ0EgRWRnZSBGdW5jdGlvbiBlc3TDoSBleGlnaW5kbyBKV1Qgbm8gZ2F0ZXdheS
4gRmHDp2EgZGVwbG95IGNvbSB2ZXJpZnlfand0PWZhbHNlLidcbiAgICAgIGlmIChlcnIgPT09ICdmdW5jdGlvbl9ub3RfZGVwbG95ZWQnKSByZXR1cm4gJ0
EgZnVuw6fDo28gcGF5bWVudHMgbsOjbyBlc3TDoSBwdWJsaWNhZGEgbm8gU3VwYWJhc2UuJ1xuICAgICAgY29uc3QgYXNKc29uID0gc2FmZUpzb24oYm9keS
lcbiAgICAgIGlmIChlcnIgJiYgYXNKc29uKSByZXR1cm4gYEVycm8gYW8gaW5pY2lhciBwYWdhbWVudG86ICR7ZXJyfSAoSFRUUCAke3N0YXR1c30pOiAke2
FzSnNvbn1gXG4gICAgICBpZiAoZXJyKSByZXR1cm4gYEVycm8gYW8gaW5pY2lhciBwYWdhbWVudG86ICR7ZXJyfSAoSFRUUCAke3N0YXR1c30pLmBcbiAgIC
AgIGlmIChhc0pzb24pIHJldHVybiBgRXJybyBhbyBpbmljaWFyIHBhZ2FtZW50byAoSFRUUCAke3N0YXR1c30pOiAke2FzSnNvbn1gXG4gICAgfVxuICAgIH
JldHVybiBgRXJybyBhbyBpbmljaWFyIHBhZ2FtZW50byAoSFRUUCAke3N0YXR1c30pLmBcbiAgfVxuXG4gIGNvbnN0IGNhblNob3dGcmVlID0gQm9vbGVhbi
h1c3VhcmlvICYmIHVzdWFyaW8ucGxhbm8gPT09ICdmcmVlJyAmJiB1c3VhcmlvLnN0YXR1c19wYWdhbWVudG8gPT09ICd0cmlhbCcgJiYgdXN1YXJpby5mcm
VlX3RyaWFsX2NvbnN1bWlkbyAhPT0gdHJ1ZSlcblxuICBjb25zdCBwbGFucyA9IHVzZU1lbW88UGxhbkNhcmRbXT4oXG4gICAgKCkgPT5cbiAgICAgIFtcbi
AgICAgICAgLi4uKGNhblNob3dGcmVlXG4gICAgICAgICAgPyBbXG4gICAgICAgICAgICAgIHtcbiAgICAgICAgICAgICAgICBrZXk6ICdmcmVlJyBhcyBjb2
5zdCxcbiAgICAgICAgICAgICAgICB0aXRsZTogJ0ZSRUUnLFxuICAgICAgICAgICAgICAgIHByaWNlTGFiZWw6ICdSJCAwL23DqnMnLFxuICAgICAgICAgIC
AgICAgIHN1YnRpdGxlOiAnUGFyYSB0ZXN0YXInLFxuICAgICAgICAgICAgICAgIGJ1bGxldHM6IFsnQXTDqSAzMCBhZ2VuZGFtZW50b3MgcG9yIG3DqnMnLC
AnMSBwcm9maXNzaW9uYWwnLCAnTGVtYnJldGVzIG1hbnVhaXMgKGxpbmsgZG8gV2hhdHNBcHApJywgJ1N1cG9ydGUgcG9yIGVtYWlsJ10sXG4gICAgICAgIC
AgICAgIH0sXG4gICAgICAgICAgICBdXG4gICAgICAgICAgOiBbXSksXG4gICAgICAgIHtcbiAgICAgICAgICBrZXk6ICdiYXNpYycsXG4gICAgICAgICAgdG
l0bGU6ICdCQVNJQycsXG4gICAgICAgICAgcHJpY2VMYWJlbDogJ1IkIDM0LDk5L23DqnMnLFxuICAgICAgICAgIHN1YnRpdGxlOiAnJyxcbiAgICAgICAgIC
BidWxsZXRzOiBbJ0FnZW5kYW1lbnRvcyA2MCBwb3IgbcOqcycsICcxIHByb2Zpc3Npb25hbCBpbmNsdcOtZG8nLCAnTGVtYnJldGVzIGF1dG9tw6F0aWNvcy
B2aWEgV2hhdHNBcHAnLCAnQXTDqSAzIHNlcnZpw6dvcycsICdQw6FnaW5hIHDDumJsaWNhIHBlcnNvbmFsaXrDoXZlbCcsICdTdXBvcnRlIHBvciBlbWFpbC
ddLFxuICAgICAgICB9LFxuICAgICAgICB7XG4gICAgICAgICAga2V5OiAncHJvJyxcbiAgICAgICAgICB0aXRsZTogJ1BSTycsXG4gICAgICAgICAgcHJpY2
VMYWJlbDogJ1IkIDU5LDk5L23DqnMnLFxuICAgICAgICAgIHN1YnRpdGxlOiAnQXTDqSAxMiBwcm9maXNzaW9uYWlzICg4IGluY2x1c29zICsgYXTDqSA0IG
FkaWNpb25haXMgZGUgUiQgNyknLFxuICAgICAgICAgIGJ1bGxldHM6IFsnQXTDqSA4IHByb2Zpc3Npb25haXMgaW5jbHXDrWRvcycsICdTZXJ2acOnb3MgaW
xpbWl0YWRvcycsICdMb2dvIGUgZm90b3MgZGUgc2VydmnDp29zJywgJ1JlbGF0w7NyaW9zJywgJ0Jsb3F1ZWlvcyByZWNvcnJlbnRlcycsICdTdXBvcnRlIH
ZpYSBXaGF0c0FwcCddLFxuICAgICAgICB9LFxuICAgICAgICB7XG4gICAgICAgICAga2V5OiAnZW50ZXJwcmlzZScsXG4gICAgICAgICAgdGl0bGU6ICdFTV
BSRVNBJyxcbiAgICAgICAgICBwcmljZUxhYmVsOiAnUiQgOTgsOTkvbcOqcycsXG4gICAgICAgICAgc3VidGl0bGU6ICdJbGltaXRhZG8nLFxuICAgICAgIC
AgIGJ1bGxldHM6IFsnUHJvZmlzc2lvbmFpcyBpbGltaXRhZG9zJywgJ011bHRpLXVuaWRhZGVzJywgJ0FnZW5kYW1lbnRvcyBpbGltaXRhZG9zJywgJ1Nlcn
Zpw6dvcyBpbGltaXRhZG9zJywgJ0xvZ28gZSBmb3RvcyBkZSBzZXJ2acOnb3MnLCAnU3Vwb3J0ZSB2aWEgV2hhdHNBcHAnXSxcbiAgICAgICAgfSxcbiAgIC
AgIF0gc2F0aXNmaWVzIFBsYW5DYXJkW10sXG4gICAgW2NhblNob3dGcmVlXVxuICApXG5cbiAgY29uc3Qgc2VydmljZXMgPSB1c2VNZW1vPFNlcnZpY2VDYX
JkW10+KFxuICAgICgpID0+XG4gICAgICBbXG4gICAgICAgIHsga2V5OiAnc2V0dXBfY29tcGxldG8nLCB0aXRsZTogJ1NldHVwIENvbXBsZXRvJywgcHJpY2
VMYWJlbDogJ1IkIDE1MCAodW1hIHZleiknLCBidWxsZXRzOiBbJ1ZvY8OqIGNvbmZpZ3VyYSB0dWRvIHBhcmEgbyBjbGllbnRlJywgJ0NhZGFzdHJhIHNlcn
Zpw6dvcywgZm90b3MsIGhvcsOhcmlvcycsICdDb25lY3RhIFdoYXRzQXBwJywgJ1Rlc3RhIGVudmlvcycsICdUcmVpbmEgbyBjbGllbnRlIGVtIDE1IG1pbn
V0b3MnXSB9LFxuICAgICAgICB7IGtleTogJ2NvbnN1bHRvcmlhX2hvcmEnLCB0aXRsZTogJ0NvbnN1bHRvcmlhIHBvciBIb3JhJywgcHJpY2VMYWJlbDogJ1
IkIDgwL2hvcmEnLCBidWxsZXRzOiBbJ0FqdWRhIGNvbSBjb25maWd1cmHDp8O1ZXMgYXZhbsOnYWRhcycsICdTdWdlc3TDtWVzIGRlIG90aW1pemHDp8Ojby
csICdEw7p2aWRhcyBnZXJhaXMnXSB9LFxuICAgICAgXSBzYXRpc2ZpZXMgU2VydmljZUNhcmRbXSxcbiAgICBbXVxuICApXG5cbiAgY29uc3QgY3VycmVudF
BsYW4gPSB1c2VNZW1vPFBsYW5LZXk+KCgpID0+IHtcbiAgICBjb25zdCBjdXJyZW50ID0gKHVzdWFyaW8/LnBsYW5vID8/ICcnKS50cmltKCkudG9Mb3dlck
Nhc2UoKSBhcyBQbGFuS2V5XG4gICAgaWYgKGN1cnJlbnQgPT09ICdmcmVlJykgcmV0dXJuIGNhblNob3dGcmVlID8gJ2ZyZWUnIDogJ2Jhc2ljJ1xuICAgIG
lmIChjdXJyZW50ID09PSAnZW50ZXJwcmlzZScpIHJldHVybiAnZW50ZXJwcmlzZSdcbiAgICBpZiAoY3VycmVudCA9PT0gJ3RlYW0nKSByZXR1cm4gJ3Byby
dcbiAgICBpZiAoY3VycmVudCA9PT0gJ2Jhc2ljJyB8fCBjdXJyZW50ID09PSAncHJvJykgcmV0dXJuIGN1cnJlbnRcbiAgICByZXR1cm4gJ2Jhc2ljJ1xuIC
B9LCBbY2FuU2hvd0ZyZWUsIHVzdWFyaW8/LnBsYW5vXSlcblxuICBjb25zdCBzZWxlY3RlZFBsYW4gPSB1c2VyU2VsZWN0ZWRQbGFuID8/IGN1cnJlbnRQbG
FuXG5cbiAgY29uc3QgZGVmYXVsdEZ1bmNpb25hcmlvc1RvdGFsID0gdXNlTWVtbygoKSA9PiB7XG4gICAgY29uc3QgcmF3ID0gdXN1YXJpbz8ubGltaXRlX2
Z1bmNpb25hcmlvc1xuICAgIGNvbnN0IG4gPSB0eXBlb2YgcmF3ID09PSAnbnVtYmVyJyAmJiBOdW1iZXIuaXNGaW5pdGUocmF3KSAmJiByYXcgPiAwID8gTW
F0aC5mbG9vcihyYXcpIDogMVxuICAgIHJldHVybiBNYXRoLm1heCgxLCBNYXRoLm1pbigyMDAsIG4pKVxuICB9LCBbdXN1YXJpbz8ubGltaXRlX2Z1bmNpb2
```

```
5hcmlvc10pXG5cbiAgY29uc3QgaW5jbHVkZWRQcm8gPSA4XG4gIGNvbnN0IG1heFBybyA9IDEyXG4gIGNvbnN0IGRlZmF1bHRQcm9GdW5jaW9uYXJpb3NUb3
RhbCA9IHVzZU1lbW8oKCkgPT4ge1xuICAgIHJldHVybiBNYXRoLm1pbihtYXhQcm8sIE1hdGgubWF4KGluY2x1ZGVkUHJvLCBkZWZhdWx0RnVuY2lvbmFyaW
9zVG90YWwpKVxuICB9LCBbZGVmYXVsdEZ1bmNpb25hcmlvc1RvdGFsLCBpbmNsdWRlZFBybywgbWF4UHJvXSlcblxuICBjb25zdCBlZmZlY3RpdmVGdW5jaW
9uYXJpb3NUb3RhbCA9IGZ1bmNpb25hcmlvc1RvdGFsID8/IChzZWxlY3RlZFBsYW4gPT09ICdwcm8nID8gZGVmYXVsdFByb0Z1bmNpb25hcmlvc1RvdGFsID
ogZGVmYXVsdEZ1bmNpb25hcmlvc1RvdGFsKVxuXG4gIHVzZUVmZmVjdCgoKSA9PiB7XG4gICAgY29uc3QgcnVuID0gYXN5bmMgKCkgPT4ge1xuICAgICAgY2
9uc3Qgc2VhcmNoID0gbG9jYXRpb24uc2VhcmNoID8/ICcnXG4gICAgICBpZiAoIXNlYXJjaCkgcmV0dXJuXG4gICAgICBjb25zdCBwYXJhbXMgPSBuZXcgVV
JMU2VhcmNoUGFyYW1zKHNlYXJjaClcbiAgICAgIGNvbnN0IGNoZWNrb3V0ID0gKHBhcmFtcy5nZXQoJ2NoZWNrb3V0JykgPz8gJycpLnRyaW0oKS50b0xvd2
VyQ2FzZSgpXG4gICAgICBpZiAoY2hlY2tvdXQgIT09ICdzdWNjZXNzJyAmJiBjaGVja291dCAhPT0gJ2NhbmNlbCcpIHJldHVyblxuXG4gICAgICBjb25zdC
BpdGVtID0gKHBhcmFtcy5nZXQoJ2l0ZW0nKSA/PyBwYXJhbXMuZ2V0KCdwbGFubycpID8/ICcnKS50cmltKCkudG9Mb3dlckNhc2UoKSB8fCBudWxsXG4gIC
AgICBzZXRDaGVja291dE5vdGljZSh7IGtpbmQ6IGNoZWNrb3V0LCBpdGVtIH0pXG5cbiAgICAgIGNvbnN0IHNlc3Npb25JZCA9IChwYXJhbXMuZ2V0KCdzZX
NzaW9uX2lkJykgPz8gJycpLnRyaW0oKVxuICAgICAgY29uc3QgdXN1YXJpb0lkRnJvbVBhcmFtcyA9IChwYXJhbXMuZ2V0KCd1c3VhcmlvX2lkJykgPz8gJy
cpLnRyaW0oKSB8fCBudWxsXG5cbiAgICAgIGNvbnN0IGZpcnN0ID0gYXdhaXQgcmVmcmVzaCgpXG4gICAgICBjb25zdCByZWZyZXNoZWRVc3VhcmlvSWQgPS
BmaXJzdD8ua2luZCA9PT0gJ3VzdWFyaW8nID8gZmlyc3QucHJvZmlsZS5pZCA6IG51bGxcbiAgICAgIGNvbnN0IGVmZmVjdGl2ZVVzdWFyaW9JZCA9IHJlZn
Jlc2hlZFVzdWFyaW9JZCA/PyB1c3VhcmlvSWRGcm9tUGFyYW1zXG5cbiAgICAgIGlmIChjaGVja291dCA9PT0gJ3N1Y2Nlc3MnKSB7XG4gICAgICAgIGlmIC
hzZXNzaW9uSWQpIHtcbiAgICAgICAgICBjb25zdCBzeW5jID0gYXdhaXQgc3luY0NoZWNrb3V0U2Vzc2lvblBhZ2FtZW50byhzZXNzaW9uSWQsIGVmZmVjdG
l2ZVVzdWFyaW9JZClcbiAgICAgICAgICBpZiAoIXN5bmMub2spIHtcbiAgICAgICAgICAgIHNldEVycm9yKGZvcm1hdENoZWNrb3V0RXJyb3Ioc3luYy5zdG
F0dXMsIHN5bmMuYm9keSkpXG4gICAgICAgICAgfSBlbHNlIHtcbiAgICAgICAgICAgIGF3YWl0IHJlZnJlc2goKVxuICAgICAgICAgIH1cbiAgICAgICAgfV
xuXG4gICAgICAgIGxldCB0cmllcyA9IDBcbiAgICAgICAgd2hpbGUgKHRyaWVzIDwgMTApIHtcbiAgICAgICAgICBhd2FpdCBuZXcgUHJvbWlzZSgocikgPT
4gc2V0VGltZW91dChyLCAyMDAwKSlcbiAgICAgICAgICBjb25zdCBuZXh0ID0gYXdhaXQgcmVmcmVzaCgpXG4gICAgICAgICAgaWYgKG5leHQ/LmtpbmQgPT
09ICd1c3VhcmlvJyAmJiBuZXh0LnByb2ZpbGUuc3RhdHVzX3BhZ2FtZW50byA9PT0gJ2F0aXZvJykgYnJlYWtcbiAgICAgICAgICB0cmllcyArPSAxXG4gIC
AgICAgIH1cbiAgICAgIH1cblxuICAgICAgbmF2aWdhdGUoJy9wYWdhbWVudG8nLCB7IHJlcGxhY2U6IHRydWUgfSlcbiAgICB9XG4gICAgcnVuKCkuY2F0Y2
goKCkgPT4gdW5kZWZpbmVkKVxuICB9LCBbbG9jYXRpb24uc2VhcmNoLCBuYXZpZ2F0ZSwgcmVmcmVzaF0pXG5cbiAgY29uc3Qgc3RhcnRQbGFuQ2hlY2tvdX
QgPSBhc3luYyAobWV0b2RvOiAnY2FyZCcgfCAncGl4JykgPT4ge1xuICAgIGlmICghdXN1YXJpb0lkKSByZXR1cm5cbiAgICBjb25zdCBwbGFuID0gc2VsZW
N0ZWRQbGFuXG4gICAgaWYgKCFwbGFuIHx8IHBsYW4gPT09ICdmcmVlJykge1xuICAgICAgc2V0RXJyb3IoJ1NlbGVjaW9uZSB1bSBwbGFubyB2w6FsaWRvLi
cpXG4gICAgICByZXR1cm5cbiAgICB9XG5cbiAgICBjb25zdCByZXF1ZXN0ZWQgPSBwbGFuID09PSAncHJvJyA/IE1hdGguZmxvb3IoZWZmZWN0aXZlRnVuY2
lvbmFyaW9zVG90YWwgfHwgaW5jbHVkZWRQcm8pIDogMVxuICAgIGlmIChwbGFuID09PSAncHJvJyAmJiByZXF1ZXN0ZWQgPiBtYXhQcm8pIHtcbiAgICAgIH
NldFVzZXJTZWxlY3RlZFBsYW4oJ2VudGVycHJpc2UnKVxuICAgICAgc2V0RnVuY2lvbmFyaW9zVG90YWwobnVsbClcbiAgICAgIHNldEVycm9yKCdQYXJhIG
1haXMgZGUgMTIgcHJvZmlzc2lvbmFpcywgc2VsZWNpb25lIG8gcGxhbm8gRU1QUkVTQS4nKVxuICAgICAgcmV0dXJuXG4gICAgfVxuXG4gICAgY29uc3QgdG
90YWwgPSBwbGFuID09PSAncHJvJyA/IE1hdGgubWF4KGluY2x1ZGVkUHJvLCBNYXRoLm1pbihtYXhQcm8sIHJlcXVlc3RlZCkpIDogMVxuXG4gICAgc2V0Q3
JlYXRpbmdDaGVja291dCh0cnVlKVxuICAgIHNldEVycm9yKG51bGwpXG4gICAgY29uc3QgcmVzID0gYXdhaXQgY3JlYXRlQ2hlY2tvdXRQYWdhbWVudG8odX
N1YXJpb0lkLCBwbGFuLCBtZXRvZG8sIHRvdGFsKVxuICAgIGlmICghcmVzLm9rKSB7XG4gICAgICBzZXRFcnJvcihmb3JtYXRDaGVja291dEVycm9yKHJlcy
5zdGF0dXMsIHJlcy5ib2R5KSlcbiAgICAgIHNldENyZWF0aW5nQ2hlY2tvdXQoZmFsc2UpXG4gICAgICByZXR1cm5cbiAgICB9XG4gICAgY29uc3QgYm9keS
A9IHJlcy5ib2R5IGFzIFJlY29yZDxzdHJpbmcsIHVua25vd24+XG4gICAgY29uc3QgdXJsID0gdHlwZW9mIGJvZHkudXJsID09PSAnc3RyaW5nJyA/IGJvZH
kudXJsIDogbnVsbFxuICAgIGlmICghdXJsKSB7XG4gICAgICBzZXRFcnJvcignQSBmdW7Dp8OjbyBuw6NvIHJldG9ybm91IG8gbGluayBkZSBjaGVja291dC
4nKVxuICAgICAgc2V0Q3JlYXRpbmdDaGVja291dChmYWxzZSlcbiAgICAgIHJldHVyblxuICAgIH1cbiAgICB3aW5kb3cubG9jYXRpb24uaHJlZiA9IHVybF
xuICB9XG5cbiAgY29uc3Qgc3RhcnRTZXJ2aWNlQ2hlY2tvdXQgPSBhc3luYyAobWV0b2RvOiAnY2FyZCcgfCAncGl4JykgPT4ge1xuICAgIGlmICghdXN1YX
Jpb0lkIHx8ICFzZWxlY3RlZFNlcnZpY2UpIHJldHVyblxuICAgIHNldENyZWF0aW5nQ2hlY2tvdXQodHJ1ZSlcbiAgICBzZXRFcnJvcihudWxsKVxuICAgIG
NvbnN0IHJlcyA9IGF3YWl0IGNyZWF0ZUNoZWNrb3V0UGFnYW1lbnRvKHVzdWFyaW9JZCwgc2VsZWN0ZWRTZXJ2aWNlLCBtZXRvZG8pXG4gICAgaWYgKCFyZX
Mub2spIHtcbiAgICAgIHNldEVycm9yKGZvcm1hdENoZWNrb3V0RXJyb3IocmVzLnN0YXR1cywgcmVzLmJvZHkpKVxuICAgICAgc2V0Q3JlYXRpbmdDaGVja2
91dChmYWxzZSlcbiAgICAgIHJldHVyblxuICAgIH1cbiAgICBjb25zdCBib2R5ID0gcmVzLmJvZHkgYXMgUmVjb3JkPHN0cmluZywgdW5rbm93bj5cbiAgIC
Bjb25zdCB1cmwgPSB0eXBlb2YgYm9keS51cmwgPT09ICdzdHJpbmcnID8gYm9keS51cmwgOiBudWxsXG4gICAgaWYgKCF1cmwpIHtcbiAgICAgIHNldEVycm
9yKCdBIGZ1bsOnw6NvIG7Do28gcmV0b3Jub3UgbyBsaW5rIGRlIGNoZWNrb3V0LicpXG4gICAgICBzZXRDcmVhdGluZ0NoZWNrb3V0KGZhbHNlKVxuICAgIC
AgcmV0dXJuXG4gICAgfVxuICAgIHdpbmRvdy5sb2NhdGlvbi5ocmVmID0gdXJsXG4gIH1cblxuICBpZiAoIXVzdWFyaW8pIHtcbiAgICByZXR1cm4gKFxuIC
AgICAgPEFwcFNoZWxsPlxuICAgICAgICA8ZGl2IGNsYXNzTmFtZT1cInRleHQtc2xhdGUtNzAwXCI+QWNlc3NvIHJlc3RyaXRvLjwvZGl2PlxuICAgICAgPC
9BcHBTaGVsbD5cbiAgICApXG4gIH1cblxuICBjb25zdCBmb3JtYXRTdGF0dXNQYWdhbWVudG8gPSAodmFsdWU6IHN0cmluZykgPT4ge1xuICAgIGNvbnN0IH
YgPSBTdHJpbmcodmFsdWUgPz8gJycpLnRyaW0oKS50b0xvd2VyQ2FzZSgpXG4gICAgaWYgKCF2KSByZXR1cm4gJ+KAlCdcbiAgICBpZiAodiA9PT0gJ2F0aX
ZvJykgcmV0dXJuICdBdGl2bydcbiAgICBpZiAodiA9PT0gJ3RyaWFsJykgcmV0dXJuICdUcmlhbCdcbiAgICBpZiAodiA9PT0gJ2luYWRpbXBsZW50ZScpIH
JldHVybiAnSW5hZGltcGxlbnRlJ1xuICAgIGlmICh2ID09PSAnc3VzcGVuc28nKSByZXR1cm4gJ1N1c3BlbnNvJ1xuICAgIGlmICh2ID09PSAnY2FuY2VsYW
RvJykgcmV0dXJuICdDYW5jZWxhZG8nXG4gICAgcmV0dXJuIHZhbHVlXG4gIH1cblxuICBjb25zdCBzdGF0dXNUb25lID0gdXN1YXJpby5zdGF0dXNfcGFnYW
1lbnRvID09PSAnaW5hZGltcGxlbnRlJyA/ICdyZWQnIDogdXN1YXJpby5zdGF0dXNfcGFnYW1lbnRvID09PSAnYXRpdm8nID8gJ2dyZWVuJyA6ICdzbGF0ZS
dcblxuICBjb25zdCBwbGFub0xhYmVsID0gKHBsYW5vUmF3OiBzdHJpbmcpID0+IHtcbiAgICBjb25zdCBwID0gU3RyaW5nKHBsYW5vUmF3ID8/ICcnKS50cm
ltKCkudG9Mb3dlckNhc2UoKVxuICAgIGlmIChwID09PSAnZW50ZXJwcmlzZScpIHJldHVybiAnRU1QUkVTQSdcbiAgICBpZiAocCA9PT0gJ3RlYW0nKSByZX
R1cm4gJ1BSTydcbiAgICBpZiAocCA9PT0gJ3BybycpIHJldHVybiAnUFJPJ1xuICAgIGlmIChwID09PSAnYmFzaWMnKSByZXR1cm4gJ0JBU0lDJ1xuICAgIG
lmIChwID09PSAnZnJlZScpIHJldHVybiAnRlJFRSdcbiAgICByZXR1cm4gcGxhbm9SYXdcbiAgfVxuXG4gIHJldHVybiAoXG4gICAgPEFwcFNoZWxsPlxuIC
AgICAgPGRpdiBjbGFzc05hbWU9XCJzcGFjZS15LTZcIj5cbiAgICAgICAge2NoZWNrb3V0Tm90aWNlID8gKFxuICAgICAgICAgIDxkaXZcbiAgICAgICAgIC
AgIGNsYXNzTmFtZT17W1xuICAgICAgICAgICAgICAncm91bmRlZC14bCBib3JkZXIgcC00IHRleHQtc20nLFxuICAgICAgICAgICAgICBjaGVja291dE5vdG
ljZS5raW5kID09PSAnc3VjY2VzcycgPyAnYm9yZGVyLWVtZXJhbGQtMjAwIGJnLWVtZXJhbGQtNTAgdGV4dC1lbWVyYWxkLTgwMCcgOiAnYm9yZGVyLWFtYm
VyLTIwMCBiZy1hbWJlci01MCB0ZXh0LWFtYmVyLTkwMCcsXG4gICAgICAgICAgICBdLmpvaW4oJyAnKX1cbiAgICAgICAgICA+XG4gICAgICAgICAgICA8ZG
l2IGNsYXNzTmFtZT1cImZsZXggZmxleC13cmFwIGl0ZW1zLWNlbnRlciBqdXN0aWZ5LWJldHdlZW4gZ2FwLTNcIj5cbiAgICAgICAgICAgICAgPGRpdiBjbG
Fzc05hbWU9XCJzcGFjZS15LTFcIj5cbiAgICAgICAgICAgICAgICA8ZGl2IGNsYXNzTmFtZT1cImZvbnQtc2VtaWJvbGRcIj57Y2hlY2tvdXROb3RpY2Uua2
luZCA9PT0gJ3N1Y2Nlc3MnID8gJ1BhZ2FtZW50byBjb25jbHXDrWRvJyA6ICdQYWdhbWVudG8gY2FuY2VsYWRvJ308L2Rpdj5cbiAgICAgICAgICAgICAgIC
A8ZGl2PlxuICAgICAgICAgICAgICAgICAge2NoZWNrb3V0Tm90aWNlLml0ZW0gPyBgSXRlbTogJHtjaGVja291dE5vdGljZS5pdGVtLnRvVXBwZXJDYXNlKC
l9LiBgIDogJyd9U3RhdHVzOiB7Zm9ybWF0U3RhdHVzUGFnYW1lbnRvKHVzdWFyaW8uc3RhdHVzX3BhZ2FtZW50byl9LlxuICAgICAgICAgICAgICAgIDwvZG
l2PlxuICAgICAgICAgICAgICA8L2Rpdj5cbiAgICAgICAgICAgICAgPGRpdiBjbGFzc05hbWU9XCJmbGV4IGl0ZW1zLWNlbnRlciBnYXAtMlwiPlxuICAgIC
AgICAgICAgIDxCdXR0b24gdmFyaWFudD1cInNlY29uZGFyeVwiIG9uQ2xpY2s9eygpID0+IHZvaWQgcmVmcmVzaCgpfT5cbiAgICAgICAgICAgICAgICAgIC
AgIEF0dWFsaXphclxuICAgICAgICAgICAgICAgIDwvQnV0dG9uPlxuICAgICAgICAgICAgICAgIDxCdXR0b24gdmFyaWFudD1cInNlY29uZGFyeVwiIG9uQ2
xpY2s9eygpID0+IHNldENoZWNrb3V0Tm90aWNlKG51bGwpfT5cbiAgICAgICAgICAgICAgICAgIEZlY2hhclxuICAgICAgICAgICAgICAgIDwvQnV0dG9uPl
xuICAgICAgICAgICAgICA8L2Rpdj5cbiAgICAgICAgICAgIDwvZGl2PlxuICAgICAgICAgIDwvZGl2PlxuICAgICAgICApIDogbnVsbH1cblxuICAgICAgIC
B7ZXJyb3IgPyA8ZGl2IGNsYXNzTmFtZT1cInJvdW5kZWQteGwgYm9yZGVyIGJvcmRlci1yb3NlLTIwMCBiZy1yb3NlLTUwIHAtMyB0ZXh0LXNtIHRleHQtcm
9zZS03MDBcIj57ZXJyb3J9PC9kaXY+IDogbnVsbH1cblxuICAgICAgICA8Q2FyZD5cbiAgICAgICAgICA8ZGl2IGNsYXNzTmFtZT1cInAtNiBzcGFjZS15LT
RcIj5cbiAgICAgICAgICAgIDxkaXYgY2xhc3NOYW1lPVwiZmxleCBmbGV4LXdyYXAgaXRlbXMtc3RhcnQganVzdGlmeS1iZXR3ZWVuIGdhcC0zXCI+XG4gIC
AgICAgICAgICAgIDxkaXY+XG4gICAgICAgICAgICAgICAgPGRpdiBjbGFzc05hbWU9XCJ0ZXh0LXNtIGZvbnQtc2VtaWJvbGQgdGV4dC1zbGF0ZS05MDBcIj
```

```
5QYWdhbWVudG88L2Rpdj5cbiAgICAgICAgICAgICAgICA8ZGl2IGNsYXNzTmFtZT1cInRleHQtc20gdGV4dC1zbGF0ZS02MDBcIj5cbiAgICAgICAgICAgIC
AgICAgIFBsYW5vIGF0dWFsOiB7cGxhbm9MYWJlbCh1c3VhcmlvLnBsYW5vKX1cbiAgICAgICAgICAgICAgICA8L2Rpdj5cbiAgICAgICAgICAgICAgPC9kaX
Y+XG4gICAgICAgICAgICAgIDxkaXYgY2xhc3NOYW1lPVwiZmxleCBpdGVtcy1jZW50ZXIgZ2FwLTJcIj5cbiAgICAgICAgICAgICAgICA8QmFkZ2UgdG9uZT
17dXN1YXJpby5hdGl2byA/ICdncmVlbicgOiAncmVkJ30+e3VzdWFyaW8uYXRpdm8gPyAnQ29udGE6IEF0aXZhJyA6ICdDb250YTogSW5hdGl2YSd9PC9CYW
RnZT5cbiAgICAgICAgICAgICAgICA8QmFkZ2UgdG9uZT17c3RhdHVzVG9uZX0+UGFnYW1lbnRvOiB7Zm9ybWF0U3RhdHVzUGFnYW1lbnRvKHVzdWFyaW8uc3
RhdHVzX3BhZ2FtZW50byl9PC9CYWRnZT5cbiAgICAgICAgICAgICAgICA8QnV0dG9uIHZhcmlhbnQ9XCJzZWNvbmRhcnlcIiBvbkNsaWNrPXsoKSA9PiB2b2
lkIHJlZnJlc2goKX0+XG4gICAgICAgICAgICAgICAgICBBdHVhbGl6YXIgc3RhdHVzXG4gICAgICAgICAgICAgICAgPC9CdXR0b24+XG4gICAgICAgICAgIC
AgIDwvZGl2PlxuICAgICAgICAgICAgPC9kaXY+XG5cbiAgICAgICAgICAgIDxkaXYgY2xhc3NOYW1lPVwiZ3JpZCBncmlkLWNvbHMtMSBnYXAtMyBzbTpncm
lkLWNvbHMtMiBsZzpncmlkLWNvbHMtM1wiPlxuICAgICAgICAgICAgICB7cGxhbnMubWFwKChwKSA9PiB7XG4gICAgICAgICAgICAgICAgY29uc3Qgc2VsZW
N0ZWQgPSBzZWxlY3RlZFBsYW4gPT09IHAua2V5XG4gICAgICAgICAgICAgICAgY29uc3QgY2xpY2thYmxlID0gcC5rZXkgIT09ICdmcmVlJ1xuICAgICAgIC
AgICAgICAgIGNvbnN0IGJlc3QgPSBwLmtleSA9PT0gJ3BybydcbiAgICAgICAgICAgICAgICByZXR1cm4gKFxuICAgICAgICAgICAgICAgICAgPGJ1dHRvbl
xuICAgICAgICAgICAgICAgICAgICBrZXk9e3Aua2V5fVxuICAgICAgICAgICAgICAgICAgICB0eXBlPVwiYnV0dG9uXCJcbiAgICAgICAgICAgICAgICAgIC
Agb25DbGljaz17KCkgPT4ge1xuICAgICAgICAgICAgICAgICAgICAgIGlmICghY2xpY2thYmxlKSByZXR1cm5cbiAgICAgICAgICAgICAgICAgICAgICBzZX
RVc2VyU2VsZWN0ZWRQbGFuKHAua2V5KVxuICAgICAgICAgICAgICAgICAgICAgIHNldEZ1bmNpb25hcmlvc1RvdGFsKG51bGwpXG4gICAgICAgICAgICAgIC
AgICAgIH19XG4gICAgICAgICAgICAgICAgICAgIGNsYXNzTmFtZT17W1xuICAgICAgICAgICAgICAgICAgICAgICd0ZXh0LWxlZnQgcm91bmRlZC14bCBib3
JkZXIgYmctd2hpdGUgcC00IHRyYW5zaXRpb24nLFxuICAgICAgICAgICAgICAgICAgICAgIGJlc3RcbiAgICAgICAgICAgICAgICAgICAgICAgID8gc2VsZW
N0ZWRcbiAgICAgICAgICAgICAgICAgICAgICAgICAgPyAnYm9yZGVyLXNsYXRlLTkwMCByaW5nLTIgcmluZy1zbGF0ZS05MDAvMTAgYmctYW1iZXItNTAnXG
4gICAgICAgICAgICAgICAgICAgICAgICAgIDogJ2JvcmRlci1hbWJlci0zMDAgaG92ZXI6YmctYW1iZXItNTAnXG4gICAgICAgICAgICAgICAgICAgICAgIC
A6IHNlbGVjdGVkXG4gICAgICAgICAgICAgICAgICAgICAgICAgID8gJ2JvcmRlci1zbGF0ZS05MDAgcmluZy0yIHJpbmctc2xhdGUtOTAwLzEwJ1xuICAgIC
AgICAgICAgICAgICAgICAgICA6ICdib3JkZXItc2xhdGUtMjAwIGhvdmVyOmJnLXNsYXRlLTUwJyxcbiAgICAgICAgICAgICAgICAgICBjbGlja2
FibGUgPyAnJyA6ICdjdXJzb3ItZGVmYXVsdCcsXG4gICAgICAgICAgICAgICAgICAgIF0uam9pbignICcpfVxuICAgICAgICAgICAgICAgICAgPlxuICAgIC
AgICAgICAgICAgICA8ZGl2IGNsYXNzTmFtZT1cImZsZXggaXRlbXMtc3RhcnQganVzdGlmeS1iZXR3ZWVuIGdhcC0zXCI+XG4gICAgICAgICAgICAgIC
AgICAgICAgPGRpdj5cbiAgICAgICAgICAgICAgICAgICAgICAgIDxkaXYgY2xhc3NOYW1lPVwidGV4dC1zbSBmb250LXNlbWlib2xkIHRleHQtc2xhdGUtOT
AwXCI+e3AudGl0bGV9PC9kaXY+XG4gICAgICAgICAgICAgICAgICAgICAgICA8ZGl2IGNsYXNzTmFtZT1cInRleHQteHMgdGV4dC1zbGF0ZS02MDBcIj57cC
5wcmljZUxhYmVsfXtwLnN1YnRpdGxlID8gYCDigKIgJHtwLnN1YnRpdGxlfWAgOiAnJ308L2Rpdj5cbiAgICAgICAgICAgICAgICAgICAgICA8L2Rpdj5cbi
AgICAgICAgICAgICAgICAgICA8ZGl2IGNsYXNzTmFtZT1cImZsZXggaXRlbXMtY2VudGVyIGdhcC0yXCI+XG4gICAgICAgICAgICAgICAgICAgICAgICAgIC
B7YmVzdCA/IDxCYWRnZSB0b25lPVwieWVsbG93XCI+TWVsaG9yIG9ww6fDo288L0JhZGdlPiA6IG51bGx9XG4gICAgICAgICAgICAgICAgICAgICAgICB7c2
VsZWN0ZWQgPyA8QmFkZ2UgdG9uZT1cInNsYXRlXCI+U2VsZWNpb25hZG88L0JhZGdlPiA6IG51bGx9XG4gICAgICAgICAgICAgICAgICAgICAgPC9kaXY+XG
4gICAgICAgICAgICAgICAgIDwvZGl2PlxuICAgICAgICAgICAgICAgICAgICA8ZGl2IGNsYXNzTmFtZT1cIm10LTMgc3BhY2UteS0xXCI+XG4gICAgICAgIC
AgICAgICAgICAgICAge3AuYnVsbGV0cy5tYXAoKGIpID0+IChcbiAgICAgICAgICAgICAgICAgICAgIDxkaXYga2V5PXtifSBjbGFzc05hbWU9XC
J0ZXh0LXhzIHRleHQtc2xhdGUtNzAwXCI+XG4gICAgICAgICAgICAgICAgICAgICC0ge2J9XG4gICAgICAgICAgICAgICAgICAgICA8L2Rpdj
5cbiAgICAgICAgICAgICAgICAgICApKX1cbiAgICAgICAgICAgICAgICAgPC9kaXY+XG4gICAgICAgICAgICAgICA8L2J1dHRvbj5cbiAgIC
AgICAgICAgICApXG4gICAgICAgICAgIH0pfVxuICAgICAgICAgPC9kaXY+XG5cbiAgICAgICAgIHtzZWxlY3RlZFBsYW4gPT09ICdwcm
8nID8gKFxuICAgICAgICAgICA8ZGl2IGNsYXNzTmFtZT1cImdyaWQgZ3JpZC1jb2xzLTEgZ2FwLTMgc206Z3JpZC1jb2xzLTJcIj5cbiAgICAgIC
AgICAgICA8SW5wdXRcbiAgICAgICAgICAgICAgIGxhYmVsPVwiUHJvZmlzc2lvbmFpc1wiXG4gICAgICAgICAgICAgICB0eXBlPVwibnVtYmVyXC
JcbiAgICAgICAgICAgICAgIG1pbj17aW5jbHVkZWRQcm99XG4gICAgICAgICAgICAgICBtYXg9e21heFByb31cbiAgICAgICAgICAgICAgIH
ZhbHVlPXtTdHJpbmcoZWZmZWN0aXZlRnVuY2lvbmFyaW9zVG90YWwpfVxuICAgICAgICAgICAgICAgb25DaGFuZ2U9eyhlKSA9PiB7XG4gICAgIC
AgICAgICAgIGNvbnN0IHJhdyA9IGUudGFyZ2V0LnZhbHVlXG4gICAgICAgICAgICAgICAgIGNvbnN0IG4gPSByYXcudHJpbSgpID09PSAnJyA/IG
luY2x1ZGVkUHJvIDogTnVtYmVyKHJhdylcbiAgICAgICAgICAgICAgICAgaWYgKCFOdW1iZXIuaXNGaW5pdGUobikpIHJldHVyblxuICAgICAgIC
AgICAgICBjb25zdCBpID0gTWF0aC5mbG9vcihuKVxuICAgICAgICAgICAgICAgICBpZiAoaSA+IG1heFBybykge1xuICAgICAgICAgICAgIC
AgICAgIHNldFVzZXJTZWxlY3RlZFBsYW4oJ2VudGVycHJpc2UnKVxuICAgICAgICAgICAgICAgICAgIHNldEZ1bmNpb25hcmlvc1RvdGFsKG51bGwpXG
4gICAgICAgICAgICAgICAgICAgc2V0RXJyb3IoJ1BhcmEgbWFpcyBkZSAxMiBwcm9maXNzaW9uYWlzLCBzZWxlY2lvbmUgbyBwbGFubyBFTVBSRVNBLi
cpXG4gICAgICAgICAgICAgICAgICAgcmV0dXJuXG4gICAgICAgICAgICAgICAgIH1cbiAgICAgICAgICAgICAgICAgY29uc3QgY2xhbXBlZC
A9IE1hdGgubWF4KGluY2x1ZGVkUHJvLCBNYXRoLm1pbihtYXhQcm8sIGkpKVxuICAgICAgICAgICAgICAgICBzZXRGdW5jaW9uYXJpb3NUb3RhbChjbG
FtcGVkKVxuICAgICAgICAgICAgICAgfX1cbiAgICAgICAgICAgICAvPlxuICAgICAgICAgICAgIDxkaXYgY2xhc3NOYW1lPVwicm91bmRlZC
14bCBib3JkZXIgYm9yZGVyLXNsYXRlLTIwMCBiZy1zbGF0ZS01MCBwLTQgdGV4dC14cyB0ZXh0LXNsYXRlLTcwMCBmbGV4IGl0ZW1zLWNlbnRlclwiPlxuIC
AgICAgICAgICAgICAgTyBjaGVja291dCBjYWxjdWxhIDggcHJvZmlzc2lvbmFpcyBpbmNsdXNvcyArIGFkaWNpb25hbCBwb3IgcHJvZmlzc2lvbmFsIG
FjaW1hIGRlIDggKG3DoXhpbW8gMTIgbm8gUFJPKS5cbiAgICAgICAgICAgICA8L2Rpdj5cbiAgICAgICAgICAgPC9kaXY+XG4gICAgICAgIC
ApIDogbnVsbH1cblxuICAgICAgICAgPGRpdiBjbGFzc05hbWU9XCJmbGV4IGZsZXgtd3JhcCBqdXN0aWZ5LWVuZCBnYXAtMlwiPlxuICAgICAgIC
AgICA8QnV0dG9uIHZhcmlhbnQ9XCJzZWNvbmRhcnlcIiBvbkNsaWNrPXsoKSA9PiB2b2lkIHN0YXJ0UGxhbkNoZWNrb3V0KCdwaXgnKX0gZGlzYWJsZWQ9e2
NyZWF0aW5nQ2hlY2tvdXQgfHwgc2VsZWN0ZWRQbGFuID09PSAnZnJlZSd9PlxuICAgICAgICAgICAgIHtjcmVhdGluZ0NoZWNrb3V0ID8gJ0FicmluZG
/igKYnIDogJ1BJWCAoMzAgZGlhcyknfVxuICAgICAgICAgICA8L0J1dHRvbj5cbiAgICAgICAgICAgPEJ1dHRvbiBvbkNsaWNrPXsoKSA9PiB2b2
lkIHN0YXJ0UGxhbkNoZWNrb3V0KCdjYXJkJyl9IGRpc2FibGVkPXtjcmVhdGluZ0NoZWNrb3V0IHx8IHNlbGVjdGVkUGxhbiA9PT0gJ2ZyZWUnfT5cbiAgIC
AgICAgICAgICB7Y3JlYXRpbmdDaGVja291dCA/ICdBYnJpbmRv4oCmJyA6ICdDYXJ0w6NvIChhc3NpbmF0dXJhKSd9XG4gICAgICAgICAgIDwvQn
V0dG9uPlxuICAgICAgICAgPC9kaXY+XG4gICAgICAgPC9kaXY+XG4gICAgIDwvQ2FyZD5cblxuICAgICA8Q2FyZD5cbiAgICAgIC
A8ZGl2IGNsYXNzTmFtZT1cInAtNiBzcGFjZS15LTRcIj5cbiAgICAgICAgIDxkaXY+XG4gICAgICAgICAgIDxkaXYgY2xhc3NOYW1lPVwidGV4dC
1zbSBmb250LXNlbWlib2xkIHRleHQtc2xhdGUtOTAwXCI+U2VydmnDp29zPC9kaXY+XG4gICAgICAgICAgIDxkaXYgY2xhc3NOYW1lPVwidGV4dC1zbS
B0ZXh0LXNsYXRlLTYwMFwiPkNvbnRyYXRlIHNlcnZpw6dvcyBhdnVsc29zIHBhcmEgY29uZmlndXJhw6fDo28gZSBzdXBvcnRlLjwvZGl2PlxuICAgIC
AgICAgPC9kaXY+XG5cbiAgICAgICAgIDxkaXYgY2xhc3NOYW1lPVwiZ3JpZCBncmlkLWNvbHMtMSBnYXAtMyBzbTpncmlkLWNvbHMtMlwiPlxuICAgIC
AgICAgICB7c2VydmljZXMubWFwKChzKSA9PiB7XG4gICAgICAgICAgICAgY29uc3Qgc2VsZWN0ZWQgPSBzZWxlY3RlZFNlcnZpY2UgPT09IHMua2
V5XG4gICAgICAgICAgICAgcmV0dXJuIChcbiAgICAgICAgICAgICAgIDxidXR0b25cbiAgICAgICAgICAgICAgICAga2V5PXtzLmtleX1cbi
AgICAgICAgICAgICAgICAgdHlwZT1cImJ1dHRvblwiXG4gICAgICAgICAgICAgICAgIG9uQ2xpY2s9eygpID0+IHNldFNlbGVjdGVkU2VydmljZS
hzLmtleSl9XG4gICAgICAgICAgICAgICAgIGNsYXNzTmFtZT17W1xuICAgICAgICAgICAgICAgICAgICd0ZXh0LWxlZnQgcm91bmRlZC14bCBib3
JkZXIgYmctd2hpdGUgcC00IHRyYW5zaXRpb24nLFxuICAgICAgICAgICAgICAgICAgIHNlbGVjdGVkID8gJ2JvcmRlci1zbGF0ZS05MDAgcmluZy0yIH
Jpbmctc2xhdGUtOTAwLzEwJyA6ICdib3JkZXItc2xhdGUtMjAwIGhvdmVyOmJnLXNsYXRlLTUwJyxcbiAgICAgICAgICAgICAgICAgXS5qb2luKCcgJy
l9XG4gICAgICAgICAgICAgICA+XG4gICAgICAgICAgICAgICAgIDxkaXYgY2xhc3NOYW1lPVwiZmxleCBpdGVtcy1zdGFydCBqdXN0aWZ5LWJldH
dlZW4gZ2FwLTNcIj5cbiAgICAgICAgICAgICAgICAgICA8ZGl2PlxuICAgICAgICAgICAgICAgICAgICAgPGRpdiBjbGFzc05hbWU9XCJ0ZXh0LX
NtIGZvbnQtc2VtaWJvbGQgdGV4dC1zbGF0ZS05MDBcIj57cy50aXRsZX08L2Rpdj5cbiAgICAgICAgICAgICAgICAgICAgIDxkaXYgY2xhc3NOYW1lPV
widGV4dC14cyB0ZXh0LXNsYXRlLTYwMFwiPntzLnByaWNlTGFiZWx9PC9kaXY+XG4gICAgICAgICAgICAgICAgICAgPC9kaXY+XG4gICAgICAgIC
AgICAgICAge3NlbGVjdGVkID8gPEJhZGdlIHRvbmU9XCJzbGF0ZVwiPlNlbGVjaW9uYWRvPC9CYWRnZT4gOiBudWxsfVxuICAgICAgICAgICAgIC
AgICA8L2Rpdj5cbiAgICAgICAgICAgICAgICAgPGRpdiBjbGFzc05hbWU9XCJtdC0zIHNwYWNlLXktMVwiPlxuICAgICAgICAgICAgICAgICAgIH
tzLmJ1bGxldHMubWFwKChiKSA9PiAoXG4gICAgICAgICAgICAgICAgICAgICA8ZGl2IGtleT17Yn0gY2xhc3NOYW1lPVwidGV4dC14cyB0ZXh0LXNsYX
```

```
RlLTcwMFwiPlxuICAgICAgICAgICAgICAgICAgICAgICAgICAtIHtifVxuICAgICAgICAgICAgICAgICAgICAgPC9kaXY+XG4gICAgICAgICAgICAgIC
AgICAgICAgKSl9XG4gICAgICAgICAgICAgICAgICAgIDwvZGl2PlxuICAgICAgICAgICAgICAgICAgPC9idXR0b24+XG4gICAgICAgICAgICAgKVxuIC
AgICAgICAgICAgICB9KX1cbiAgICAgICAgICAgIDwvZGl2PlxuXG4gICAgICAgICAgICA8ZGl2IGNsYXNzTmFtZT1cImZsZXggZmxleC13cmFwIGp1c3RpZn
ktZW5kIGdhcC0yXCI+XG4gICAgICAgICAgICAgIDxCdXR0b24gdmFyaWFudD1cInNlY29uZGFyeVwiIG9uQ2xpY2s9eygpID0+IHZvaWQgc3RhcnRTZXJ2aW
NlQ2hlY2tvdXQoJ3BpeCcpfSBkaXNhYmxlZD17Y3JlYXRpbmdDaGVja291dCB8fCAhc2VsZWN0ZWRTZXJ2aWNlfT5cbiAgICAgICAgICAgICAgICB7Y3JlYX
RpbmdDaGVja291dCA/ICdBYnJpbmRv4oCmJyA6ICdQYWdhciBjb20gUElYJ31cbiAgICAgICAgICAgICAgPC9CdXR0b24+XG4gICAgICAgICAgICAgIDxCdX
R0b24gb25DbGljaz17KCkgPT4gdm9pZCBzdGFydFNlcnZpY2VDaGVja291dCgnY2FyZCcpfSBkaXNhYmxlZD17Y3JlYXRpbmdDaGVja291dCB8fCAhc2VsZW
N0ZWRTZXJ2aWNlfT5cbiAgICAgICAgICAgICAgICB7Y3JlYXRpbmdDaGVja291dCA/ICdBYnJpbmRv4oCmJyA6ICdQYWdhciBjb20gY2FydMOjbyd9XG4gIC
AgICAgICAgICAgIDwvQnV0dG9uPlxuICAgICAgICAgICAgPC9kaXY+XG4gICAgICAgICAgPC9kaXY+XG4gICAgICAgIDwvQ2FyZD5cbiAgICAgIDwvZGl2Pl
xuICAgIDwvQXBwU2hlbGw+XG4gIClcbn1cbiJdLCJmaWxlIjoiQzovVXNlcnMvQWRtaW4vRGVza3RvcC9TTWFnZW5kYS9zbWFnZW5kYS9zcmMvdmlld3MvYX
BwL1BhZ2FtZW50b1BhZ2UudHN4In0=
```

functions/api/_utils.js

```
export function json(data, init) {
  const status = init?.status ?? 200
  const headers = new Headers(init?.headers)
  if (!headers.has('content-type')) headers.set('content-type', 'application/json; charset=utf-8')
  return new Response(JSON.stringify(data), { status, headers })
}

export function getApiKey(context) {
  return String(context?.env?.GOOGLE_MAPS_API_KEY ?? '').trim()
}

export async function forwardJson(url) {
  const upstream = await fetch(url)
  const text = await upstream.text()
  let parsed
  try {
    parsed = text ? JSON.parse(text) : null
  } catch {
    parsed = { raw: text }
  }
  return json(parsed, { status: upstream.status })
}
```

functions/api/geocode.js

```
import { forwardJson, getApiKey, json } from './_utils'

export async function onRequestGet(context) {
  const apiKey=[REDACTED]
  if (!apiKey) return json({ error: 'missing_env', message: 'Defina GOOGLE_MAPS_API_KEY no Cloudflare Pages' }, {
status: 500 })

  const address = String(context?.request?.url ? new URL(context.request.url).searchParams.get('address') : '').trim()
  if (!address) return json({ error: 'invalid_address' }, { status: 400 })

  const url = new URL('https://maps.googleapis.com/maps/api/geocode/json')
  url.searchParams.set('key', apiKey)
  url.searchParams.set('address', address)
  return await forwardJson(url.toString())
}
```

functions/api/health.js

```
import { getApiKey, json } from './_utils'

export async function onRequestGet(context) {
  const apiKey=[REDACTED]
  return json({ ok: true, hasKey: Boolean(apiKey) })
}
```

functions/api/places/details.js

```
import { forwardJson, getApiKey, json } from '../_utils'

export async function onRequestGet(context) {
  const apiKey=[REDACTED]
  if (!apiKey) return json({ error: 'missing_env', message: 'Defina GOOGLE_MAPS_API_KEY no Cloudflare Pages' }, {
status: 500 })

  const u = new URL(context.request.url)
  const placeId = String(u.searchParams.get('place_id') ?? '').trim()
  if (!placeId) return json({ error: 'invalid_place_id' }, { status: 400 })

  const url = new URL('https://maps.googleapis.com/maps/api/place/details/json')
  url.searchParams.set('key', apiKey)
  url.searchParams.set('place_id', placeId)
  url.searchParams.set(
    'fields',
    [
      'place_id',
      'name',
      'formatted_address',
      'address_component',
      'geometry',
      'types',
      'url',
      'website',
      'formatted_phone_number',
      'international_phone_number',
    ].join(',')
  )

  return await forwardJson(url.toString())
}
```

functions/api/places/textsearch.js

```
import { forwardJson, getApiKey, json } from '../_utils'

export async function onRequestGet(context) {
  const apiKey=[REDACTED]
  if (!apiKey) return json({ error: 'missing_env', message: 'Defina GOOGLE_MAPS_API_KEY no Cloudflare Pages' }, {
status: 500 })

  const u = new URL(context.request.url)
  const query = String(u.searchParams.get('query') ?? '').trim()
  const pageToken = String(u.searchParams.get('pagetoken') ?? '').trim()
  const location = String(u.searchParams.get('location') ?? '').trim()
  const radius = String(u.searchParams.get('radius') ?? '').trim()
  if (!query) return json({ error: 'invalid_query' }, { status: 400 })

  const url = new URL('https://maps.googleapis.com/maps/api/place/textsearch/json')
  url.searchParams.set('key', apiKey)
  url.searchParams.set('query', query)
  if (pageToken) url.searchParams.set('pagetoken', pageToken)
  if (location) url.searchParams.set('location', location)
  if (radius) url.searchParams.set('radius', radius)
  return await forwardJson(url.toString())
}
```

gemini.md

Essa é uma das maiores travas de quem cria produtos: o medo da concorrência. Mas vou te contar um segredo do mundo dos negócios: você não precisa ser o único, você só precisa ser a melhor opção para um grupo específico de pessoas.

As grandes empresas (como Calendly, Trinks, etc.) são como grandes supermercados: têm tudo, mas o atendimento é impessoal. Você será a "padaria artesanal do bairro": conhece o cliente pelo nome e entrega exatamente o que ele gosta.

Aqui estão 4 estratégias matadoras para diferenciar o seu sistema, mesmo ele sendo tecnicamente parecido com os outros:

1. O Diferencial do "Concierge" (Serviço vs. Software)

As grandes empresas vendem um login e senha e dizem "se vira para configurar". O dono da barbearia ou a manicure muitas vezes não tem paciência ou conhecimento técnico para configurar fotos, horários e serviços.

Sua Diferenciação: Você vende a Implementação Assistida.

O Pitch de Venda: "Diferente dos outros sistemas onde você perde horas configurando, eu vou pessoalmente (ou via chamada de vídeo) configurar tudo para você. Eu cadastro seus serviços, seus preços e te entrego pronto para usar. Você não terá trabalho nenhum."

Por que funciona: Você remove a barreira da preguiça e do medo da tecnologia.

2. Foco Total no WhatsApp (A Realidade Brasileira)

Muitos sistemas gringos ou grandes focam em e-mail ou SMS. No Brasil, o negócio gira no WhatsApp.

Sua Diferenciação: Seu sistema é "WhatsApp First".

A Funcionalidade: O lembrete não é um SMS frio. É uma mensagem no WhatsApp com um texto amigável e personalizado (ex: "Oi Maria! Amanhã é dia de ficar linda!").

O Pitch: "Meus concorrentes mandam e-mail que ninguém lê. Meu sistema manda WhatsApp que seu cliente responde na hora. Isso garante que ele venha."

3. A "Simplicidade Radical" (Foco no Não-Tecnológico)

Sistemas grandes tentam fazer tudo: estoque, financeiro, emissão de nota, comissão... Isso deixa a tela cheia de botões e confusa.

Sua Diferenciação: O sistema mais fácil do mundo.

O Design: Botões grandes, poucas opções, zero curva de aprendizado. Focado em quem ainda usa papel e caneta.

O Pitch: "Meu sistema foi feito para quem odeia computador. Se você sabe usar o WhatsApp, sabe usar meu sistema. Sem botões desnecessários."

4. Nichar para Dominar (A Estratégia do Especialista)

Em vez de fazer um "Agendador Geral", faça um "Agendador para [Nicho Específico]".

Exemplo: Se você focar em Lava-Jatos.

A Diferenciação: Em vez de "Serviço", você chama de "Lavagem". Em vez de "Profissional", chama de "Lavador". Você adiciona campos como "Placa do Carro" e "Modelo".

O Efeito Psicológico: O dono do lava-jato vai olhar para o sistema genérico e para o seu, e vai pensar: "Esse cara entende do meu negócio. O sistema dele foi feito pra mim."

Resumo da Ópera

Seu concorrente não é a grande empresa de software. Seu maior concorrente é o Caderno de Papel e o WhatsApp caótico.

Você vai ganhar deles não por ter mais funcionalidades (features), mas por oferecer mais atenção e menos dor de cabeça.

Qual dessas abordagens você acha que consegue executar melhor hoje?

Ser o cara que configura tudo (Concierge)?

Focar na automação de WhatsApp?

Nichar em um setor específico (ex: só Barbearias ou só Lava-jatos)?

Me diz qual prefere e eu te ajudo a montar a frase exata para você mandar no WhatsApp dos primeiros clientes potenciais.

Sua estrutura está muito profissional. O Stripe é excelente para escala global e cartões, e o servidor de 4GB de RAM na Hetzner segura bem o tranco inicial (provavelmente cerca de 20 a 40 instâncias da Evolution API simultâneas, dependendo do uso).

Como você já vai implementar o envio manual e a URL personalizada, aqui estão os últimos 3 ajustes "de mestre" para você fechar o sistema antes de ir para a rua:

1. Configure o PIX no Stripe (Urgente)
Muitos desenvolvedores usam o Stripe apenas para cartão, mas no Brasil, o dono da barbearia ou a manicure vão querer pagar via PIX.

O que fazer: Vá no painel do Stripe e ative o método de pagamento "Pix".

Atenção: Garanta que seu Webhook esteja processando o evento checkout.session.completed. Quando o cara pagar o Pix, o Stripe avisa seu sistema e você libera o acesso dele no Supabase automaticamente. Sem isso, você terá que liberar manualmente, o que mata a escala.

2. Ative o SWAP no Servidor Linux (Segurança)
Como você tem 4GB de RAM e a Evolution API consome memória para cada QR Code conectado, se você tiver um pico de conexões, o servidor pode dar "Out of Memory" e travar tudo.

O que fazer: Crie um arquivo de Swap de 4GB no seu Linux da Hetzner. Isso usa um pouco do seu SSD de 80GB como se fosse memória RAM de emergência.

Comando rápido (como root):

Bash

```
fallocate -l 4G /swapfile
chmod 600 /swapfile
mkswap /swapfile
swapon /swapfile
# Adicione ao /etc/fstab para ser permanente
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

3. Persistência de Sessão (O pesadelo do QR Code)
Não há nada que irrite mais o cliente do que ter que ler o QR Code do WhatsApp todo dia porque o servidor reiniciou.

Verificação: Como você está na Hetzner, verifique se as sessões da Evolution API estão sendo salvas em uma pasta persistente (Volumes no Docker ou uma pasta local). Se o servidor reiniciar e a pasta /instances for apagada, todos os seus clientes ficarão desconectados. Teste isso hoje: conecte um WhatsApp, reinicie o servidor e veja se ele volta conectado.

Dica Bônus para a Venda Porta a Porta: "O Link Mágico"
Quando você chegar no cliente, não peça para ele criar conta. Faça o seguinte:

Tenha uma Conta de Demonstração já pronta com dados falsos (Barbearia do Teste).

Mostre para ele a Visão do Cliente dele (a página de agendamento).

Diga: "Vou te mandar o link de teste agora no seu WhatsApp, clica aí e finge que é seu cliente agendando pra você ver a velocidade".

Por que isso funciona? O cliente sente a experiência na mão dele antes de falar de preço.

O que levar amanhã (Checklist de Saída):
[ ] Celular carregado com o sistema logado na conta demo.

[ ] Link do seu WhatsApp pessoal fácil para passar para eles.

[ ] Um script de preço na cabeça (ex: "É 60 reais por mês, mas se fechar hoje o primeiro mês é por minha conta para você testar").

Você se sente seguro com a parte de cobrar o cliente? Já decidiu se vai cobrar a primeira mensalidade na hora ou dar 7 dias grátis? Minha recomendação para quem está começando: 7 dias grátis sem pedir cartão, para ele viciar no sistema, e no 8º dia o sistema bloqueia e pede o pagamento.

Boa sorte amanhã! Você tem um produto real, funcional e que resolve um problema. Vá com confiança!
--]
Para que uma página de agendamento converta um visitante em cliente, ela precisa transmitir **confiança** e ser **extremamente fácil de usar no polegar** (já que o cliente estará segurando o celular).

Como o seu sistema já tem a lógica, vamos focar no **UI/UX (Design e Experiência)**. Aqui estão as melhorias práticas para deixar o SMagenda com cara de "app de alto nível":

~~~
---

### 1. O Topo: Identidade e Confiança

Em vez de apenas o nome, crie um "Card de Perfil" no topo.

* **Foto de Perfil/Logo:** Circular e centralizada.
* **Banner de Fundo:** Uma imagem suave (pode ser o interior do salão ou uma cor sólida elegante).
* **Bio Curta e Localização:** Abaixo do nome, coloque o endereço (com link para o Google Maps) e o Instagram. Isso prova que o negócio é real.

### 2. Estrutura em "Passos" (Steppers)

Não mostre tudo de uma vez. Isso assusta o cliente. Divida a tela em 3 etapas claras:

1. **O quê?** (Seleção de Serviço)
2. **Com quem?** (Seleção de Profissional - se houver mais de um)
3. **Quando?** (Data e Hora)

### 3. Melhoria Visual dos Serviços

Abandone as listas simples. Use **Cards de Serviço**:

* **Título em Negrito.**
* **Badge de Tempo e Preço:** Ex: `[ 45 min ]` `[ R$ 50,00 ]` em destaque.
* **Descrição Curta:** Ex: "Corte degradê com finalização em pomada".
* **Botão "Selecionar":** Que muda de cor ou vira um "Check" quando clicado.

### 4. O Calendário e Horários (O ponto crítico)

No celular, calendários grandes são ruins.

* **Seletor de Data Horizontal:** Use uma linha com os próximos 7 a 14 dias (Ex: "Seg, 02", "Ter, 03"). É muito mais fácil de navegar com o dedão.
* **Grid de Horários:** Divida os horários em períodos: **Manhã, Tarde e Noite**. Isso ajuda o cliente a encontrar o que quer mais rápido sem dar scroll infinito.

---

### Exemplo de Estrutura de Código (React + Tailwind)

Aqui está uma sugestão de como estruturar o componente da página de agendamento para ficar profissional:

```jsx
// Exemplo de Card de Serviço Profissional
const ServiceCard = ({ name, price, duration, description }) => (
  <div className="border rounded-xl p-4 mb-3 bg-white shadow-sm active:scale-95 transition-all cursor-pointer border-l-4 border-l-blue-600">
    <div className="flex justify-between items-start">
      <div>
        <h3 className="font-bold text-gray-800 text-lg">{name}</h3>
        <p className="text-sm text-gray-500 line-clamp-2">{description}</p>
      </div>
      <div className="text-right">
        <span className="block font-bold text-blue-600">R$ {price}</span>
        <span className="text-xs text-gray-400">{duration} min</span>
      </div>
    </div>
  </div>
);

// Exemplo de Header
const BusinessHeader = ({ name, address }) => (
  <div className="relative pb-6 border-b">
    <div className="h-24 bg-gradient-to-r from-blue-500 to-blue-700 w-full"></div>
    <div className="px-4 -mt-10 flex flex-col items-center">
      <div className="w-20 h-20 rounded-full border-4 border-white bg-gray-200 overflow-hidden shadow-lg">
        <img src="logo-cliente.png" alt="Logo" />
      </div>
      <h1 className="mt-2 text-xl font-bold text-gray-800">{name}</h1>
      <p className="text-xs text-gray-500 flex items-center mt-1">
        📍 {address}
~~~

~~~~
      </p>
    </div>
  </div>
);

```

---

### 5. O que NÃO pode faltar (Informativo):

* **Resumo flutuante (Sticky Footer):** Conforme o cliente seleciona o serviço e a hora, uma barra no rodapé da tela vai se preenchendo: *"Corte de Cabelo • Amanhã às 14:00 • R$ 50,00"*. E o botão **"Confirmar Agendamento"** só habilita quando tudo estiver preenchido.
* **Página de Sucesso:** Após agendar, mostre um check verde gigante e um botão: **"Adicionar ao meu Calendário"** e outro **"Abrir Localização no Maps"**.
* **Mensagem de "Fale Conosco":** Um ícone flutuante do WhatsApp no canto da página de agendamento para o caso de o cliente ter uma dúvida específica.

### Como isso ajuda na sua venda "Porta a Porta"?

Quando você mostrar isso para o dono da barbearia, você dirá:

> *"Olha como a página que seu cliente vai ver é bonita. Parece um aplicativo de R$ 100 mil reais. Ele escolhe o serviço, vê o preço, escolhe o horário e pronto. É chique e passa confiança para ele pagar mais caro no seu serviço."*

**Você quer que eu desenhe como seria esse "Resumo de Agendamento" no rodapé? Isso aumenta muito a conversão.**
~~~~

package.json

```
{
  "name": "smagenda-root",
  "private": true,
  "scripts": {
    "build": "cd smagenda && npm install && npm run build && cd ../prospector && npm install && npm run build",
    "lint": "cd smagenda && npm run lint && cd ../prospector && npm run lint",
    "dev": "cd smagenda && npm run dev",
    "preview": "cd smagenda && npm run preview"
  }
}
```

prospector/eslint.config.js

```
import js from '@eslint/js'
import globals from 'globals'
import reactHooks from 'eslint-plugin-react-hooks'
import reactRefresh from 'eslint-plugin-react-refresh'
import tseslint from 'typescript-eslint'

export default tseslint.config(
  { ignores: ['dist'] },
  {
    extends: [js.configs.recommended, ...tseslint.configs.recommended],
    files: ['**/*.{ts,tsx}'],
    languageOptions: {
      ecmaVersion: 2022,
      globals: globals.browser,
    },
    plugins: {
      'react-hooks': reactHooks,
      'react-refresh': reactRefresh,
    },
    rules: {
      ...reactHooks.configs.recommended.rules,
      'react-refresh/only-export-components': ['warn', { allowConstantExport: true }],
    },
  }
)
```

prospector/index.html

```
<!doctype html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="theme-color" content="#0b1220" />
    <link rel="manifest" href="%BASE_URL%manifest.webmanifest" />
    <link rel="icon" href="%BASE_URL%icon.svg" type="image/svg+xml" />
    <title>Prospector</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
```

prospector/package.json

```
{
  "name": "smagenda-prospector",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc -b && vite build",
    "lint": "eslint .",
    "preview": "vite preview",
    "import:itabirito": "node scripts/import_itabirito.js"
  },
  "dependencies": {
    "react": "^19.2.0",
    "react-dom": "^19.2.0"
  },
  "devDependencies": {
    "@eslint/js": "^9.39.1",
    "@types/react": "^19.2.5",
    "@types/react-dom": "^19.2.3",
    "@vitejs/plugin-react": "^5.1.1",
    "eslint": "^9.39.1",
    "eslint-plugin-react-hooks": "^7.0.1",
    "eslint-plugin-react-refresh": "^0.4.24",
    "globals": "^16.5.0",
    "typescript": "~5.9.3",
    "typescript-eslint": "^8.46.4",
    "vite": "^7.2.4"
  }
}
```

prospector/public/icon.svg

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 256 256">
  <defs>
    <linearGradient id="g" x1="0" y1="0" x2="1" y2="1">
      <stop offset="0" stop-color="#6366f1"/>
      <stop offset="1" stop-color="#22c55e"/>
    </linearGradient>
  </defs>
  <rect width="256" height="256" rx="56" fill="#0b1220"/>
  <rect x="22" y="22" width="212" height="212" rx="48" fill="url(#g)" opacity="0.18"/>
  <path fill="url(#g)" d="M128 34c45.3 0 82 36.7 82 82 0 56.8-82 140-82 140S46 172.8 46 116c0-45.3 36.7-82 82-82zm0
52a30 30 0 1 0 0 60 30 30 0 0 0 0-60z"/>
  <path fill="#ffffff" opacity="0.92" d="M160.6 176.4c-10.9 10.9-22.2 23.1-32.6 35.8-10.4-12.7-21.7-24.9-32.6-35.8 9.8-
5.2 20.9-8.1 32.6-8.1s22.8 2.9 32.6 8.1z"/>
</svg>
```

prospector/public/sw.js

```
const CACHE_NAME = 'prospector-shell-v2'

function getScopePath() {
  try {
    const u = new URL(self.registration.scope)
    return u.pathname.endsWith('/') ? u.pathname : `${u.pathname}/`
  } catch {
    return '/'
  }
}

const SCOPE = getScopePath()

function inScope(path) {
  const p = String(path || '')
  if (!p) return SCOPE
  if (p.startsWith('/')) return `${SCOPE}${p.slice(1)}`
  return `${SCOPE}${p}`
}

self.addEventListener('install', (event) => {
  event.waitUntil(
    caches
      .open(CACHE_NAME)
      .then((cache) => cache.addAll([SCOPE, inScope('index.html'), inScope('manifest.webmanifest'),
inScope('icon.svg')]))
      .then(() => self.skipWaiting())
  )
})

self.addEventListener('activate', (event) => {
  event.waitUntil(
    caches
      .keys()
      .then((keys) => Promise.all(keys.filter((k) => k !== CACHE_NAME).map((k) => caches.delete(k))))
      .then(() => self.clients.claim())
  )
})

self.addEventListener('fetch', (event) => {
  const req = event.request
  if (req.method !== 'GET') return

  const url = new URL(req.url)
  if (url.origin !== self.location.origin) return
  if (url.pathname.startsWith('/api/')) return

  if (req.mode === 'navigate') {
    event.respondWith(
      fetch(req)
        .then((res) => {
          const copy = res.clone()
          caches.open(CACHE_NAME).then((cache) => cache.put(req, copy))
          return res
        })
        .catch(() => caches.match(SCOPE))
    )
    return
  }

  event.respondWith(
    caches.match(req).then((cached) =>
      cached
        ? cached
        : fetch(req).then((res) => {
            const copy = res.clone()
            caches.open(CACHE_NAME).then((cache) => cache.put(req, copy))
            return res
          })
    )
```

```
  )
})
```

prospector/scripts/import_itabirito.js

```
import { mkdir, writeFile } from 'node:fs/promises'
import { join } from 'node:path'

const API_KEY=[REDACTED]'').trim()
if (!API_KEY) {
  console.error('Defina GOOGLE_MAPS_API_KEY no ambiente antes de rodar este script.')
  process.exit(1)
}

const CITY_QUERY = 'Itabirito MG'

const SEGMENTS = [
  { label: 'Salão de beleza', query: 'salão de beleza' },
  { label: 'Cabeleireiro', query: 'cabeleireiro' },
  { label: 'Barbearia', query: 'barbearia' },
  { label: 'Manicure', query: 'manicure' },
  { label: 'Depilação', query: 'depilação' },
  { label: 'Estética', query: 'estética' },
  { label: 'Spa', query: 'spa' },
  { label: 'Pilates', query: 'pilates' },
  { label: 'Yoga', query: 'yoga' },
  { label: 'Fisioterapia', query: 'fisioterapia' },
  { label: 'Clínica', query: 'clínica' },
  { label: 'Clínica odontológica', query: 'clínica odontológica' },
  { label: 'Dentista', query: 'dentista' },
  { label: 'Oftalmologista', query: 'oftalmologista' },
  { label: 'Médico', query: 'médico' },
]

function sleep(ms) {
  return new Promise((r) => setTimeout(r, ms))
}

async function fetchJson(url) {
  const res = await fetch(url)
  const text = await res.text()
  if (!text.trim()) throw new Error(`Resposta vazia (${res.status}) em ${url}`)
  let parsed
  try {
    parsed = JSON.parse(text)
  } catch {
    throw new Error(`JSON inválido (${res.status}) em ${url}`)
  }
  if (!res.ok) {
    const msg = parsed && typeof parsed === 'object' && typeof parsed.message === 'string' ? parsed.message : null
    throw new Error(msg ? `HTTP ${res.status}: ${msg}` : `HTTP ${res.status}`)
  }
  return parsed
}

async function geocode(address) {
  const url = new URL('https://maps.googleapis.com/maps/api/geocode/json')
  url.searchParams.set('key', API_KEY)
  url.searchParams.set('address', address)
  const body = await fetchJson(url.toString())
  const loc = body?.results?.[0]?.geometry?.location
  const lat = typeof loc?.lat === 'number' ? loc.lat : null
  const lng = typeof loc?.lng === 'number' ? loc.lng : null
  return { lat, lng }
}

async function textSearch({ query, location, radius, pageToken }) {
  const url = new URL('https://maps.googleapis.com/maps/api/place/textsearch/json')
  url.searchParams.set('key', API_KEY)
  url.searchParams.set('query', query)
  if (location) url.searchParams.set('location', location)
  if (radius) url.searchParams.set('radius', String(radius))
  if (pageToken) url.searchParams.set('pagetoken', pageToken)
  return await fetchJson(url.toString())
}
```

```
async function details(placeId) {
  const url = new URL('https://maps.googleapis.com/maps/api/place/details/json')
  url.searchParams.set('key', API_KEY)
  url.searchParams.set('place_id', placeId)
  url.searchParams.set(
    'fields',
    [
      'place_id',
      'name',
      'formatted_address',
      'address_component',
      'geometry',
      'types',
      'url',
      'website',
      'formatted_phone_number',
      'international_phone_number',
    ].join(',')
  )
  return await fetchJson(url.toString())
}

function findComponent(components, type) {
  const arr = Array.isArray(components) ? components : []
  for (const c of arr) {
    const t = Array.isArray(c?.types) ? c.types : []
    if (t.includes(type)) return c
  }
  return null
}

function extractAddressParts(placeDetails) {
  const components = placeDetails?.address_components
  const route = findComponent(components, 'route')
  const streetNumber = findComponent(components, 'street_number')
  const city = findComponent(components, 'administrative_area_level_2') ?? findComponent(components, 'locality')
  const state = findComponent(components, 'administrative_area_level_1')
  const postalCode = findComponent(components, 'postal_code')

  return {
    street: String(route?.long_name ?? '').trim() || null,
    number: String(streetNumber?.long_name ?? '').trim() || null,
    city: String(city?.long_name ?? '').trim() || null,
    state: String(state?.short_name ?? state?.long_name ?? '').trim() || null,
    postalCode: String(postalCode?.long_name ?? '').trim() || null,
  }
}

function newId() {
  return globalThis.crypto?.randomUUID ? globalThis.crypto.randomUUID() :
`${Date.now()}-${Math.random().toString(16).slice(2)}`
}

function parseStreetNumber(value) {
  const s = String(value ?? '').trim()
  if (!s) return null
  const m = s.match(/\d+/)
  if (!m) return null
  const n = Number(m[0])
  return Number.isFinite(n) ? n : null
}

function normKey(s) {
  return String(s ?? '').trim().toLowerCase()
}

function sortEstablishments(a, b) {
  const streetA = normKey(a.street)
  const streetB = normKey(b.street)
  if (streetA !== streetB) return streetA.localeCompare(streetB)
  const na = parseStreetNumber(a.number)
```

```
  const nb = parseStreetNumber(b.number)
  if (na !== null && nb !== null && na !== nb) return na - nb
  if (na === null && nb !== null) return 1
  if (na !== null && nb === null) return -1
  return normKey(a.name).localeCompare(normKey(b.name))
}

function groupByStreetAndSegment(rows) {
  const grouped = {}
  for (const e of rows) {
    const street = String(e.street ?? 'Sem rua').trim() || 'Sem rua'
    const segments = Array.isArray(e.segments) && e.segments.length > 0 ? e.segments : ['Sem segmento']
    if (!grouped[street]) grouped[street] = {}
    for (const s of segments) {
      if (!grouped[street][s]) grouped[street][s] = []
      grouped[street][s].push(e)
    }
  }

  const streets = Object.keys(grouped).sort((a, b) => normKey(a).localeCompare(normKey(b)))
  const out = []
  for (const street of streets) {
    const segs = grouped[street]
    const segKeys = Object.keys(segs).sort((a, b) => normKey(a).localeCompare(normKey(b)))
    out.push({
      street,
      segments: segKeys.map((k) => ({ segment: k, count: segs[k].length, placeIds: segs[k].map((x) => x.placeId) })),
    })
  }
  return out
}

async function main() {
  console.log(`Geocodificando centro de ${CITY_QUERY}...`)
  const center = await geocode(CITY_QUERY)
  const location = center.lat !== null && center.lng !== null ? `${center.lat},${center.lng}` : null
  const radius = 14000

  const byPlaceId = new Map()

  for (const seg of SEGMENTS) {
    const q = `${seg.query} ${CITY_QUERY}`
    console.log(`Buscando: ${q}`)

    let pageToken = null
    let pages = 0
    do {
      const search = await textSearch({ query: q, location, radius, pageToken })
      const results = Array.isArray(search?.results) ? search.results : []

      for (const r of results) {
        const placeId = String(r?.place_id ?? '').trim()
        if (!placeId) continue

        const existing = byPlaceId.get(placeId)
        if (existing) {
          const nextSegs = new Set([...(existing.segments ?? []), seg.label])
          existing.segments = Array.from(nextSegs)
          byPlaceId.set(placeId, existing)
          continue
        }

        const det = await details(placeId)
        const d = det?.result ?? null
        if (!d || typeof d !== 'object') continue

        const now = new Date().toISOString()
        const parts = extractAddressParts(d)
        const phone = String(d.formatted_phone_number ?? d.international_phone_number ?? '').trim() || null
        const url = String(d.url ?? '').trim() || null
        const website = String(d.website ?? '').trim() || null
        const formattedAddress = String(d.formatted_address ?? r.formatted_address ?? '').trim()
```

```
        const lat = typeof d.geometry?.location?.lat === 'number' ? d.geometry.location.lat : typeof r.geometry?.location?.lat === 'number' ? r.geometry.location.lat : null
        const lng = typeof d.geometry?.location?.lng === 'number' ? d.geometry.location.lng : typeof r.geometry?.location?.lng === 'number' ? r.geometry.location.lng : null
        const types = Array.isArray(d.types) ? d.types.filter((t) => typeof t === 'string') : Array.isArray(r.types) ? r.types.filter((t) => typeof t === 'string') : []

        const row = {
          id: newId(),
          placeId,
          name: String(d.name ?? r.name ?? '').trim() || 'Sem nome',
          formattedAddress,
          street: parts.street,
          number: parts.number,
          city: parts.city,
          state: parts.state,
          postalCode: parts.postalCode,
          lat,
          lng,
          types,
          segments: [seg.label],
          googleMapsUrl: url,
          phone,
          website,
          fetchedAt: now,
          status: 'novo',
          contactName: null,
          contactPhone: null,
          notes: null,
          lastVisitAt: null,
          updatedAt: now,
          createdAt: now,
        }

        byPlaceId.set(placeId, row)
        await sleep(70)
      }

      pageToken = typeof search?.next_page_token === 'string' ? search.next_page_token : null
      pages += 1
      if (pageToken) await sleep(2300)
    } while (pageToken && pages < 3)
  }

  const establishments = Array.from(byPlaceId.values()).sort(sortEstablishments)
  const grouped = groupByStreetAndSegment(establishments)

  const out = {
    city: 'Itabirito',
    state: 'MG',
    generatedAt: new Date().toISOString(),
    total: establishments.length,
    grouped,
    establishments,
  }

  const dir = join(process.cwd(), 'data')
  await mkdir(dir, { recursive: true })
  const filePath = join(dir, 'itabirito.prospector.json')
  await writeFile(filePath, JSON.stringify(out, null, 2), 'utf-8')
  console.log(`Arquivo gerado: ${filePath}`)
  console.log(`Total de estabelecimentos (deduplicado): ${establishments.length}`)
}

main().catch((err) => {
  console.error(err)
  process.exit(1)
})
```

prospector/server/package.json

```
{
  "name": "smagenda-prospector-server",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "node --watch src/index.js",
    "start": "node src/index.js"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.21.2"
  }
}
```

prospector/server/src/index.js

```
import express from 'express'
import cors from 'cors'
import { readFile } from 'node:fs/promises'
import { dirname, join } from 'node:path'
import { fileURLToPath } from 'node:url'

const app = express()
app.use(cors({ origin: true }))

async function loadEnvFileIfPresent(filePath) {
  let raw
  try {
    raw = await readFile(filePath, 'utf-8')
  } catch {
    return
  }
  for (const line of raw.split(/\r?\n/g)) {
    const s = line.trim()
    if (!s || s.startsWith('#')) continue
    const idx = s.indexOf('=')
    if (idx <= 0) continue
    const key = s.slice(0, idx).trim()
    let value = s.slice(idx + 1).trim()
    if (!key) continue
    if ((value.startsWith('"') && value.endsWith('"')) || (value.startsWith("'") && value.endsWith("'"))) {
      value = value.slice(1, -1)
    }
    if (process.env[key] == null) process.env[key] = value
  }
}

const here = dirname(fileURLToPath(import.meta.url))
await loadEnvFileIfPresent(join(here, '..', '.env'))
await loadEnvFileIfPresent(join(here, '..', '..', '.env'))
await loadEnvFileIfPresent(join(process.cwd(), '.env'))

const apiKey=[REDACTED]'').trim()

function requireKey(res) {
  if (apiKey) return true
  res.status(500).json({ error: 'missing_env', message: 'Defina GOOGLE_MAPS_API_KEY no server/.env' })
  return false
}

async function forwardJson(url, res) {
  const upstream = await fetch(url)
  const text = await upstream.text()
  let parsed
  try {
    parsed = text ? JSON.parse(text) : null
  } catch {
    parsed = { raw: text }
  }
  res.status(upstream.status).json(parsed)
}

app.get('/api/health', (_req, res) => {
  res.json({ ok: true, hasKey: Boolean(apiKey) })
})

app.get('/api/geocode', async (req, res) => {
  if (!requireKey(res)) return
  const address = String(req.query.address ?? '').trim()
  if (!address) {
    res.status(400).json({ error: 'invalid_address' })
    return
  }
  const url = new URL('https://maps.googleapis.com/maps/api/geocode/json')
  url.searchParams.set('key', apiKey)
  url.searchParams.set('address', address)
```

```
  await forwardJson(url.toString(), res)
})

app.get('/api/places/textsearch', async (req, res) => {
  if (!requireKey(res)) return
  const query = String(req.query.query ?? '').trim()
  const pageToken = String(req.query.pagetoken ?? '').trim()
  const location = String(req.query.location ?? '').trim()
  const radius = String(req.query.radius ?? '').trim()
  if (!query) {
    res.status(400).json({ error: 'invalid_query' })
    return
  }
  const url = new URL('https://maps.googleapis.com/maps/api/place/textsearch/json')
  url.searchParams.set('key', apiKey)
  url.searchParams.set('query', query)
  if (pageToken) url.searchParams.set('pagetoken', pageToken)
  if (location) url.searchParams.set('location', location)
  if (radius) url.searchParams.set('radius', radius)
  await forwardJson(url.toString(), res)
})

app.get('/api/places/details', async (req, res) => {
  if (!requireKey(res)) return
  const placeId = String(req.query.place_id ?? '').trim()
  if (!placeId) {
    res.status(400).json({ error: 'invalid_place_id' })
    return
  }
  const url = new URL('https://maps.googleapis.com/maps/api/place/details/json')
  url.searchParams.set('key', apiKey)
  url.searchParams.set('place_id', placeId)
  url.searchParams.set(
    'fields',
    [
      'place_id',
      'name',
      'formatted_address',
      'address_component',
      'geometry',
      'types',
      'url',
      'website',
      'formatted_phone_number',
      'international_phone_number',
    ].join(',')
  )
  await forwardJson(url.toString(), res)
})

const port = Number(process.env.PORT ?? '8787')
app.listen(port, () => {
  console.log(`prospector-server listening on http://localhost:${port}`)
})
```

prospector/src/App.tsx

```
import React, { useEffect, useMemo, useState } from 'react'
import type { Establishment, EstablishmentStatus, ImportPreset } from './types'
import { newId } from './lib/id'
import { extractAddressParts } from './lib/address'
import { loadState, saveState, exportJson, importJson } from './lib/storage'
import { geocode, health, placesDetails, placesTextSearch } from './lib/api'

const PRESETS: ImportPreset[] = [
  {
    key: 'agendamentos_todos',
    label: 'Todos (por agendamento)',
    queries: [
      'salão de beleza',
      'cabeleireiro',
      'barbearia',
      'barber shop',
      'manicure',
      'pedicure',
      'esmalteria',
      'depilação',
      'design de sobrancelhas',
      'extensão de cílios',
      'clínica de estética',
      'estética facial',
      'estética corporal',
      'massagem',
      'drenagem linfática',
      'spa',
      'clínica médica',
      'consultório médico',
      'dermatologista',
      'oftalmologista',
      'dentista',
      'clínica odontológica',
      'ortodontista',
      'fisioterapia',
      'pilates',
      'quiropraxia',
      'osteopatia',
      'psicólogo',
      'psiquiatra',
      'terapeuta',
      'nutricionista',
      'fonoaudiólogo',
      'laboratório',
      'clínica de exames',
      'yoga',
      'personal trainer',
      'treinamento funcional',
      'estúdio de dança',
      'artes marciais',
      'pet shop',
      'banho e tosa',
      'tosa',
      'clínica veterinária',
      'veterinário',
      'lava jato',
      'estética automotiva',
      'polimento',
      'oficina mecânica',
      'auto elétrica',
      'funilaria e pintura',
      'estúdio de tatuagem',
      'body piercing',
      'fotógrafo',
      'estúdio fotográfico',
    ],
  },
  {
    key: 'beleza',
```

```
    label: 'Beleza',
    queries: ['salão de beleza', 'cabeleireiro', 'barbearia', 'manicure', 'estética', 'spa'],
  },
  {
    key: 'saude',
    label: 'Clínicas e Saúde',
    queries: ['clínica', 'clínica odontológica', 'dentista', 'oftalmologista', 'fisioterapia'],
  },
  {
    key: 'bem_estar',
    label: 'Bem-estar',
    queries: ['pilates', 'yoga', 'massagem'],
  },
]

function downloadText(filename: string, content: string) {
  const blob = new Blob([content], { type: 'application/json;charset=utf-8' })
  const url = URL.createObjectURL(blob)
  const a = document.createElement('a')
  a.href = url
  a.download = filename
  document.body.appendChild(a)
  a.click()
  a.remove()
  URL.revokeObjectURL(url)
}

function toCsv(rows: Establishment[]) {
  const headers = [
    'placeId',
    'name',
    'formattedAddress',
    'street',
    'number',
    'city',
    'state',
    'postalCode',
    'status',
    'contactName',
    'contactPhone',
    'phone',
    'website',
    'googleMapsUrl',
    'notes',
    'lastVisitAt',
    'updatedAt',
    'createdAt',
  ]
  const escape = (v: unknown) => {
    const s = String(v ?? '')
    const needs = s.includes(',') || s.includes('"') || s.includes('\n')
    const out = s.replaceAll('"', '""')
    return needs ? `"${out}"` : out
  }
  const lines = [headers.join(',')]
  for (const r of rows) {
    lines.push(
      headers
        .map((h) =>
          escape(
            (r as unknown as Record<string, unknown>)[h] ??
              (h === 'placeId' ? r.placeId : '')
          )
        )
        .join(',')
    )
  }
  return lines.join('\n')
}

function statusLabel(s: EstablishmentStatus) {
  if (s === 'novo') return 'Novo'
```

```
  if (s === 'visitado') return 'Visitado'
  if (s === 'confirmado') return 'Confirmado'
  if (s === 'aprovado') return 'Aprovado'
  if (s === 'recusou') return 'Recusou'
  return 'Sem resposta'
}

function statusColor(s: EstablishmentStatus) {
  if (s === 'novo') return 'gray'
  if (s === 'visitado') return 'blue'
  if (s === 'confirmado') return 'violet'
  if (s === 'aprovado') return 'green'
  if (s === 'recusou') return 'red'
  return 'orange'
}

function normalizePhone(value: string) {
  const raw = String(value ?? '').trim()
  if (!raw) return ''
  const digits = raw.replace(/\D+/g, '')
  if (!digits) return ''
  if (digits.startsWith('55')) return digits
  if (digits.length >= 10) return `55${digits}`
  return digits
}

function toWhatsAppUrl(phoneRaw: string, message?: string) {
  const phone = normalizePhone(phoneRaw)
  if (!phone) return null
  const base = `https://wa.me/${phone}`
  if (!message) return base
  return `${base}?text=${encodeURIComponent(message)}`
}

function normKey(s: unknown) {
  return String(s ?? '').trim().toLowerCase()
}

function parseStreetNumber(value: unknown) {
  const s = String(value ?? '').trim()
  if (!s) return null
  const m = s.match(/\d+/)
  if (!m) return null
  const n = Number(m[0])
  return Number.isFinite(n) ? n : null
}

function sleep(ms: number) {
  return new Promise((r) => window.setTimeout(r, ms))
}

async function assertBackendReady() {
  const h = await health()
  if (!h?.ok) throw new Error('Backend /api indisponível. Verifique Cloudflare Pages Functions (produção) ou
prospector/server (dev).')
  if (h.hasKey === false) throw new Error('Defina GOOGLE_MAPS_API_KEY no Cloudflare Pages (produção) ou no
prospector/server/.env (dev).')
}

function haversineMeters(a: { lat: number; lng: number }, b: { lat: number; lng: number }) {
  const R = 6371000
  const toRad = (deg: number) => (deg * Math.PI) / 180
  const dLat = toRad(b.lat - a.lat)
  const dLng = toRad(b.lng - a.lng)
  const lat1 = toRad(a.lat)
  const lat2 = toRad(b.lat)
  const x =
    Math.sin(dLat / 2) * Math.sin(dLat / 2) +
    Math.cos(lat1) * Math.cos(lat2) * Math.sin(dLng / 2) * Math.sin(dLng / 2)
  const c = 2 * Math.atan2(Math.sqrt(x), Math.sqrt(1 - x))
  return R * c
}
```

```
type GeocodeViewportPoint = { lat?: number; lng?: number }
type GeocodeResponse = {
  results?: Array<{
    geometry?: {
      viewport?: {
        northeast?: GeocodeViewportPoint
        southwest?: GeocodeViewportPoint
      }
    }
  }>
}

function computeRadiusFromGeocode(geo: unknown) {
  const g = geo as GeocodeResponse | null
  const viewport = g?.results?.[0]?.geometry?.viewport
  const ne = viewport?.northeast
  const sw = viewport?.southwest
  const nelat = typeof ne?.lat === 'number' ? ne.lat : null
  const nelng = typeof ne?.lng === 'number' ? ne.lng : null
  const swlat = typeof sw?.lat === 'number' ? sw.lat : null
  const swlng = typeof sw?.lng === 'number' ? sw.lng : null

  if (nelat === null || nelng === null || swlat === null || swlng === null) return 14000
  const diagonal = haversineMeters({ lat: nelat, lng: nelng }, { lat: swlat, lng: swlng })
  const raw = Math.round(diagonal / 2 + 2500)
  return Math.min(50000, Math.max(6000, raw))
}

async function copyText(text: string) {
  const v = String(text ?? '').trim()
  if (!v) return false
  try {
    await navigator.clipboard.writeText(v)
    return true
  } catch {
    try {
      const ta = document.createElement('textarea')
      ta.value = v
      ta.style.position = 'fixed'
      ta.style.left = '-9999px'
      ta.style.top = '0'
      document.body.appendChild(ta)
      ta.focus()
      ta.select()
      const ok = document.execCommand('copy')
      ta.remove()
      return ok
    } catch {
      return false
    }
  }
}

function formatIsoDate(value: string | null) {
  const s = String(value ?? '').trim()
  if (!s) return ''
  const d = new Date(s)
  if (Number.isNaN(d.getTime())) return s
  return d.toLocaleString('pt-BR')
}

const CSS = `
:root{
  --bg:#0b1220;
  --panel:#0f1b31;
  --panel2:#0c172b;
  --border:rgba(255,255,255,.08);
  --text:rgba(255,255,255,.92);
  --muted:rgba(255,255,255,.68);
  --muted2:rgba(255,255,255,.52);
  --shadow: 0 18px 50px rgba(0,0,0,.35);
```

```
  --radius:14px;
  --radius2:10px;
}
*{box-sizing:border-box}
body{margin:0;background:radial-gradient(1200px 600px at 15% 10%, rgba(99,102,241,.25), transparent 60%), radial-gradient(900px 500px at 90% 0%, rgba(34,197,94,.18), transparent 60%), var(--bg); color:var(--text)}
a{color:rgba(147,197,253,.95);text-decoration:none}
a:hover{text-decoration:underline}
.app{max-width:1280px;margin:0 auto;padding:20px}
.top{display:flex;gap:14px;flex-wrap:wrap;align-items:center}
.title{display:flex;gap:10px;align-items:baseline;flex-wrap:wrap}
.title h1{margin:0;font-size:22px;letter-spacing:.2px}
.subtitle{color:var(--muted);font-size:13px}
.kpis{display:flex;gap:8px;flex-wrap:wrap;margin-left:auto}
.chip{border:1px solid var(--border);background:rgba(255,255,255,.04);padding:7px 10px;border-radius:999px;font-size:12px;color:var(--muted)}
.chip b{color:var(--text);font-weight:700}
.layout{display:grid;grid-template-columns:420px 1fr;gap:14px;margin-top:14px}
@media (max-width: 980px){.layout{grid-template-columns:1fr}}
.panel{border:1px solid var(--border);background:linear-gradient(180deg, rgba(255,255,255,.06), rgba(255,255,255,.03));border-radius:var(--radius);box-shadow:var(--shadow)}
.panelHeader{padding:12px 12px 10px;border-bottom:1px solid var(--border)}
.panelBody{padding:12px}
.row{display:flex;gap:10px;flex-wrap:wrap;align-items:end}
.field{display:grid;gap:6px;min-width:200px;flex:1}
.field span{font-size:12px;color:var(--muted)}
.input, .select, .textarea{width:100%;border:1px solid var(--border);border-radius:12px;background:rgba(255,255,255,.04);color:var(--text);padding:10px 10px;outline:none}
.input:focus, .select:focus, .textarea:focus{border-color:rgba(99,102,241,.55);box-shadow:0 0 0 4px rgba(99,102,241,.12)}
.textarea{min-height:90px;resize:vertical}
.btn{border:1px solid var(--border);background:rgba(255,255,255,.05);color:var(--text);border-radius:12px;padding:10px 12px;cursor:pointer}
.btn:hover{background:rgba(255,255,255,.09)}
.btn:disabled{opacity:.55;cursor:not-allowed}
.btnPrimary{background:linear-gradient(180deg, rgba(99,102,241,.95), rgba(79,70,229,.95));border-color:rgba(99,102,241,.55)}
.btnPrimary:hover{background:linear-gradient(180deg, rgba(129,140,248,.95), rgba(99,102,241,.95))}
.btnDanger{background:rgba(239,68,68,.14);border-color:rgba(239,68,68,.35)}
.btnDanger:hover{background:rgba(239,68,68,.2)}
.toolbar{display:flex;gap:10px;flex-wrap:wrap;align-items:end}
.muted{color:var(--muted)}
.muted2{color:var(--muted2)}
.split{display:grid;grid-template-columns:1fr 1fr;gap:10px}
@media (max-width: 700px){.split{grid-template-columns:1fr}}
.list{display:flex;flex-direction:column;gap:10px}
.streetHeader{padding:8px 10px;border:1px solid var(--border);border-radius:14px;background:rgba(255,255,255,.03);color:var(--muted);font-size:12px;letter-spacing:.2px}
.listHeader{display:flex;gap:8px;flex-wrap:wrap;align-items:center}
.listScroll{max-height:calc(100vh - 260px);overflow:auto;padding-right:6px}
@media (max-width: 980px){.listScroll{max-height:unset}}
.item{border:1px solid var(--border);background:rgba(12,23,43,.35);border-radius:14px;padding:12px;cursor:pointer;display:grid;gap:8px}
.item:hover{background:rgba(12,23,43,.5)}
.itemActive{outline:2px solid rgba(99,102,241,.45);background:rgba(12,23,43,.65)}
.itemTop{display:flex;gap:10px;align-items:flex-start}
.badge{font-size:11px;padding:5px 8px;border-radius:999px;border:1px solid var(--border);background:rgba(255,255,255,.04);color:var(--muted)}
.badge.gray{background:rgba(148,163,184,.12);border-color:rgba(148,163,184,.25);color:rgba(226,232,240,.95)}
.badge.blue{background:rgba(59,130,246,.14);border-color:rgba(59,130,246,.28);color:rgba(191,219,254,.98)}
.badge.violet{background:rgba(139,92,246,.14);border-color:rgba(139,92,246,.28);color:rgba(221,214,254,.98)}
.badge.green{background:rgba(34,197,94,.14);border-color:rgba(34,197,94,.28);color:rgba(187,247,208,.98)}
.badge.red{background:rgba(239,68,68,.14);border-color:rgba(239,68,68,.28);color:rgba(254,202,202,.98)}
.badge.orange{background:rgba(249,115,22,.14);border-color:rgba(249,115,22,.28);color:rgba(255,237,213,.98)}
.itemName{font-weight:750;line-height:1.15}
.itemAddr{font-size:12px;color:var(--muted);line-height:1.25}
.itemMeta{display:flex;gap:10px;flex-wrap:wrap;font-size:12px;color:var(--muted2)}
.actions{display:flex;gap:8px;flex-wrap:wrap}
.sep{height:1px;background:var(--border);margin:10px 0}
.hint{color:var(--muted);font-size:13px;line-height:1.4}
.progressWrap{margin-top:10px;border:1px solid var(--border);border-radius:12px;background:rgba(255,255,255,.03);padding:10px}
```

```
.progressTop{display:flex;gap:10px;flex-wrap:wrap;align-items:center;justify-content:space-between}
.progressMsg{color:var(--muted);font-size:12px;line-height:1.35}
.progressPct{color:var(--muted2);font-size:12px}
.progressTrack{margin-top:8px;height:10px;border-radius:999px;background:rgba(255,255,255,.06);border:1px solid var(--border);overflow:hidden}
.progressFill{height:100%;background:linear-gradient(90deg, rgba(99,102,241,.95), rgba(34,197,94,.75));width:0%}
.progressIndeterminate{height:100%;width:40%;background:linear-gradient(90deg, rgba(99,102,241,.95), rgba(34,197,94,.75));border-radius:999px;animation:progressMove 1.1s ease-in-out infinite}
@keyframes progressMove{0%{transform:translateX(-60%)}100%{transform:translateX(260%)}}
`

export function App() {
  const [state, setState] = useState(() => loadState())
  const [city, setCity] = useState('')
  const [uf, setUf] = useState('')
  const [presetKey, setPresetKey] = useState(PRESETS[0]?.key ?? 'beleza')
  const [loading, setLoading] = useState(false)
  const [progress, setProgress] = useState<{ active: boolean; done: number; total: number; message: string }>(() => ({ active: false, done: 0, total: 0, message: '' }))
  const [error, setError] = useState<string | null>(null)
  const [filter, setFilter] = useState('')
  const [statusFilter, setStatusFilter] = useState<EstablishmentStatus | 'all'>('all')
  const [onlyWithPhone, setOnlyWithPhone] = useState(false)
  const [onlyMissingContact, setOnlyMissingContact] = useState(false)
  const [sortKey, setSortKey] = useState<'updated_desc' | 'name_asc' | 'status' | 'last_visit_desc' | 'street_number'>('updated_desc')
  const [selectedPlaceId, setSelectedPlaceId] = useState<string | null>(null)
  const [toast, setToast] = useState<string | null>(null)
  const [installPrompt, setInstallPrompt] = useState<unknown>(null)

  const preset = useMemo(() => PRESETS.find((p) => p.key === presetKey) ?? PRESETS[0]!, [presetKey])

  const upsertMany = (rows: Establishment[]) => {
    const byPlace = new Map<string, Establishment>()
    for (const e of state.establishments) byPlace.set(e.placeId, e)
    for (const r of rows) {
      const prev = byPlace.get(r.placeId)
      if (prev) {
        const mergedSegments = (() => {
          const a = Array.isArray(prev.segments) ? prev.segments : []
          const b = Array.isArray(r.segments) ? r.segments : []
          const next = Array.from(new Set([...a, ...b].map((x) => String(x).trim()).filter(Boolean)))
          return next.length > 0 ? next : undefined
        })()
        byPlace.set(r.placeId, {
          ...prev,
          name: r.name || prev.name,
          formattedAddress: r.formattedAddress || prev.formattedAddress,
          street: r.street ?? prev.street,
          number: r.number ?? prev.number,
          city: r.city ?? prev.city,
          state: r.state ?? prev.state,
          postalCode: r.postalCode ?? prev.postalCode,
          lat: r.lat ?? prev.lat,
          lng: r.lng ?? prev.lng,
          types: r.types.length > 0 ? r.types : prev.types,
          segments: mergedSegments,
          googleMapsUrl: r.googleMapsUrl ?? prev.googleMapsUrl,
          phone: r.phone ?? prev.phone,
          website: r.website ?? prev.website,
          fetchedAt: r.fetchedAt,
          updatedAt: new Date().toISOString(),
        })
      } else {
        byPlace.set(r.placeId, r)
      }
    }
    const next = { establishments: Array.from(byPlace.values()).sort((a, b) => b.updatedAt.localeCompare(a.updatedAt)) }
    setState(next)
    saveState(next)
  }
```

```
  const updateOne = (placeId: string, patch: Partial<Establishment>) => {
    const next = {
      establishments: state.establishments.map((e) =>
        e.placeId === placeId
          ? { ...e, ...patch, updatedAt: new Date().toISOString() }
          : e
      ),
    }
    setState(next)
    saveState(next)
  }

  const toastNow = (message: string) => {
    setToast(message)
    window.setTimeout(() => setToast(null), 1800)
  }

  useEffect(() => {
    const handler = (e: Event) => {
      e.preventDefault()
      setInstallPrompt(e as unknown)
    }
    window.addEventListener('beforeinstallprompt', handler)
    return () => window.removeEventListener('beforeinstallprompt', handler)
  }, [])

  const updateStatus = (placeId: string, status: EstablishmentStatus) => {
    const patch: Partial<Establishment> = { status }
    if (status === 'visitado') patch.lastVisitAt = new Date().toISOString()
    updateOne(placeId, patch)
  }

  const progressStart = (message: string) => {
    setProgress({ active: true, done: 0, total: 0, message })
  }

  const progressSetMessage = (message: string) => {
    setProgress((p) => (p.active ? { ...p, message } : p))
  }

  const progressAddTotal = (n: number) => {
    const x = Number(n)
    if (!Number.isFinite(x) || x <= 0) return
    setProgress((p) => (p.active ? { ...p, total: p.total + x } : p))
  }

  const progressIncDone = (n: number) => {
    const x = Number(n)
    if (!Number.isFinite(x) || x <= 0) return
    setProgress((p) => (p.active ? { ...p, done: p.done + x } : p))
  }

  const progressEnd = () => {
    setProgress({ active: false, done: 0, total: 0, message: '' })
  }

  const runImport = async () => {
    const c = city.trim()
    const u = uf.trim()
    if (!c || !u) {
      setError('Informe cidade e estado (UF).')
      return
    }
    setError(null)
    setLoading(true)
    progressStart('Iniciando importação…')
    try {
      progressAddTotal(1)
      progressSetMessage('Verificando backend…')
      await assertBackendReady()
      progressIncDone(1)
      const CITY_QUERY = `${c} ${u} Brasil`
```

```
      progressAddTotal(1)
      progressSetMessage(`Geocodificando: ${CITY_QUERY}`)
      const geo = await geocode(CITY_QUERY)
      progressIncDone(1)

      const loc0 = geo.results?.[0]?.geometry?.location
      const lat0 = typeof loc0?.lat === 'number' ? loc0.lat : null
      const lng0 = typeof loc0?.lng === 'number' ? loc0.lng : null
      const location = lat0 !== null && lng0 !== null ? `${lat0},${lng0}` : null
      const radius = computeRadiusFromGeocode(geo)

      const byPlaceId = new Map<string, Establishment>()

      for (const q of preset.queries) {
        const fullQuery = `${q} ${CITY_QUERY}`
        let pageToken: string | null = null
        let pages = 0
        do {
          progressAddTotal(1)
          progressSetMessage(`Buscando: ${q} • ${c}`)
          const res = await placesTextSearch({ query: fullQuery, pageToken, location, radius })
          progressIncDone(1)
          const results = Array.isArray(res.results) ? res.results : []
          const validResults = results.filter((r) => String(r.place_id ?? '').trim())
          progressAddTotal(validResults.length)
          for (const r of validResults) {
            const pid = (r.place_id ?? '').trim()

            const existing = byPlaceId.get(pid)
            if (existing) {
              const nextSegs = new Set([...(existing.segments ?? []), q])
              existing.segments = Array.from(nextSegs)
              byPlaceId.set(pid, existing)
              progressIncDone(1)
              continue
            }

            progressSetMessage(`Detalhando: ${(r.name ?? '').trim() || pid}`)
            const det = await placesDetails(pid)
            progressIncDone(1)
            const d = det.result
            const parts = extractAddressParts(d)
            const now = new Date().toISOString()
            const phone = (d.formatted_phone_number ?? d.international_phone_number ?? '').trim() || null
            const url = (d.url ?? '').trim() || null
            const website = (d.website ?? '').trim() || null
            const formattedAddress = (d.formatted_address ?? r.formatted_address ?? '').trim()
            const lat = typeof d.geometry?.location?.lat === 'number' ? d.geometry?.location?.lat : typeof
r.geometry?.location?.lat === 'number' ? r.geometry?.location?.lat : null
            const lng = typeof d.geometry?.location?.lng === 'number' ? d.geometry?.location?.lng : typeof
r.geometry?.location?.lng === 'number' ? r.geometry?.location?.lng : null
            const types = Array.isArray(d.types) ? d.types.filter((t) => typeof t === 'string') : Array.isArray(r.types)
? r.types.filter((t) => typeof t === 'string') : []
            byPlaceId.set(pid, {
              id: newId(),
              placeId: pid,
              name: (d.name ?? r.name ?? '').trim() || 'Sem nome',
              formattedAddress,
              street: parts.street,
              number: parts.number,
              city: parts.city,
              state: parts.state,
              postalCode: parts.postalCode,
              lat,
              lng,
              types,
              segments: [q],
              googleMapsUrl: url,
              phone,
              website,
              fetchedAt: now,
```

```
                status: 'novo',
                contactName: null,
                contactPhone: null,
                notes: null,
                lastVisitAt: null,
                updatedAt: now,
                createdAt: now,
              })
              await sleep(70)
            }
            pageToken = typeof res.next_page_token === 'string' ? res.next_page_token : null
            pages += 1
            if (pageToken) await sleep(2200)
          } while (pageToken && pages < 3)
        }
        progressAddTotal(1)
        progressSetMessage('Salvando no sistema…')
        const all = Array.from(byPlaceId.values())
        upsertMany(all)
        progressIncDone(1)
        if (all.length > 0) setSelectedPlaceId(all[0]?.placeId ?? null)
      } catch (e) {
        setError(e instanceof Error ? e.message : 'Falha ao importar.')
      } finally {
        setLoading(false)
        progressEnd()
      }
    }

    const runImportItabirito = async () => {
      setError(null)
      setLoading(true)
      progressStart('Iniciando importação de Itabirito…')
      try {
        progressAddTotal(1)
        progressSetMessage('Verificando backend…')
        await assertBackendReady()
        progressIncDone(1)
        const CITY_QUERY = 'Itabirito MG'
        const SEGMENTS = [
          { label: 'Salão de beleza', query: 'salão de beleza' },
          { label: 'Cabeleireiro', query: 'cabeleireiro' },
          { label: 'Barbearia', query: 'barbearia' },
          { label: 'Manicure', query: 'manicure' },
          { label: 'Depilação', query: 'depilação' },
          { label: 'Estética', query: 'estética' },
          { label: 'Spa', query: 'spa' },
          { label: 'Pilates', query: 'pilates' },
          { label: 'Yoga', query: 'yoga' },
          { label: 'Fisioterapia', query: 'fisioterapia' },
          { label: 'Clínica', query: 'clínica' },
          { label: 'Clínica odontológica', query: 'clínica odontológica' },
          { label: 'Dentista', query: 'dentista' },
          { label: 'Oftalmologista', query: 'oftalmologista' },
          { label: 'Médico', query: 'médico' },
        ]

        setCity('Itabirito')
        setUf('MG')
        setSortKey('street_number')

        progressAddTotal(1)
        progressSetMessage('Geocodificando: Itabirito MG')
        const geo = await geocode(CITY_QUERY)
        progressIncDone(1)
        const loc0 = geo.results?.[0]?.geometry?.location
        const lat0 = typeof loc0?.lat === 'number' ? loc0.lat : null
        const lng0 = typeof loc0?.lng === 'number' ? loc0.lng : null
        const location = lat0 !== null && lng0 !== null ? `${lat0},${lng0}` : null
        const radius = 14000

        const byPlaceId = new Map<string, Establishment>()
```

```
      for (const seg of SEGMENTS) {
        const fullQuery = `${seg.query} ${CITY_QUERY}`
        let pageToken: string | null = null
        let pages = 0
        do {
          progressAddTotal(1)
          progressSetMessage(`Buscando: ${seg.label} • Itabirito`)
          const res = await placesTextSearch({ query: fullQuery, pageToken, location, radius })
          progressIncDone(1)
          const results = Array.isArray(res.results) ? res.results : []
          const validResults = results.filter((r) => String(r.place_id ?? '').trim())
          progressAddTotal(validResults.length)
          for (const r of validResults) {
            const pid = (r.place_id ?? '').trim()

            const existing = byPlaceId.get(pid)
            if (existing) {
              const nextSegs = new Set([...(existing.segments ?? []), seg.label])
              existing.segments = Array.from(nextSegs)
              byPlaceId.set(pid, existing)
              progressIncDone(1)
              continue
            }

            progressSetMessage(`Detalhando: ${(r.name ?? '').trim() || pid}`)
            const det = await placesDetails(pid)
            progressIncDone(1)
            const d = det.result
            const parts = extractAddressParts(d)
            const now = new Date().toISOString()
            const phone = (d.formatted_phone_number ?? d.international_phone_number ?? '').trim() || null
            const url = (d.url ?? '').trim() || null
            const website = (d.website ?? '').trim() || null
            const formattedAddress = (d.formatted_address ?? r.formatted_address ?? '').trim()
            const lat = typeof d.geometry?.location?.lat === 'number' ? d.geometry?.location?.lat : typeof
r.geometry?.location?.lat === 'number' ? r.geometry?.location?.lat : null
            const lng = typeof d.geometry?.location?.lng === 'number' ? d.geometry?.location?.lng : typeof
r.geometry?.location?.lng === 'number' ? r.geometry?.location?.lng : null
            const types = Array.isArray(d.types) ? d.types.filter((t) => typeof t === 'string') : Array.isArray(r.types)
? r.types.filter((t) => typeof t === 'string') : []

            byPlaceId.set(pid, {
              id: newId(),
              placeId: pid,
              name: (d.name ?? r.name ?? '').trim() || 'Sem nome',
              formattedAddress,
              street: parts.street,
              number: parts.number,
              city: parts.city,
              state: parts.state,
              postalCode: parts.postalCode,
              lat,
              lng,
              types,
              segments: [seg.label],
              googleMapsUrl: url,
              phone,
              website,
              fetchedAt: now,
              status: 'novo',
              contactName: null,
              contactPhone: null,
              notes: null,
              lastVisitAt: null,
              updatedAt: now,
              createdAt: now,
            })
            await sleep(70)
          }
          pageToken = typeof res.next_page_token === 'string' ? res.next_page_token : null
          pages += 1
```

```
          if (pageToken) await sleep(2200)
        } while (pageToken && pages < 3)
      }

      progressAddTotal(1)
      progressSetMessage('Salvando no sistema…')
      upsertMany(Array.from(byPlaceId.values()))
      progressIncDone(1)
      toastNow('Itabirito importado')
    } catch (e) {
      setError(e instanceof Error ? e.message : 'Falha ao importar.')
    } finally {
      setLoading(false)
      progressEnd()
    }
  }

  const filtered = useMemo(() => {
    const f = filter.trim().toLowerCase()
    const rows = state.establishments.filter((e) => {
      if (statusFilter !== 'all' && e.status !== statusFilter) return false
      if (onlyWithPhone && !String(e.phone ?? '').trim()) return false
      if (onlyMissingContact) {
        const hasContactName = String(e.contactName ?? '').trim()
        const hasContactPhone = String(e.contactPhone ?? '').trim()
        if (hasContactName || hasContactPhone) return false
      }
      if (!f) return true
      const hay = `${e.name} ${e.formattedAddress} ${e.street ?? ''} ${e.city ?? ''}`.toLowerCase()
      return hay.includes(f)
    })
    const sorted = [...rows]
    sorted.sort((a, b) => {
      if (sortKey === 'updated_desc') return b.updatedAt.localeCompare(a.updatedAt)
      if (sortKey === 'last_visit_desc') return String(b.lastVisitAt ?? '').localeCompare(String(a.lastVisitAt ?? ''))
      if (sortKey === 'name_asc') return a.name.localeCompare(b.name)
      if (sortKey === 'status') return a.status.localeCompare(b.status) || b.updatedAt.localeCompare(a.updatedAt)
      if (sortKey === 'street_number') {
        const sa = normKey(a.street)
        const sb = normKey(b.street)
        if (sa !== sb) return sa.localeCompare(sb)
        const na = parseStreetNumber(a.number)
        const nb = parseStreetNumber(b.number)
        if (na !== null && nb !== null && na !== nb) return na - nb
        if (na === null && nb !== null) return 1
        if (na !== null && nb === null) return -1
        return normKey(a.name).localeCompare(normKey(b.name))
      }
      return b.updatedAt.localeCompare(a.updatedAt)
    })
    return sorted
  }, [state.establishments, filter, statusFilter, onlyWithPhone, onlyMissingContact, sortKey])

  const listNodes = useMemo(() => {
    if (sortKey !== 'street_number') return filtered.map((e) => ({ kind: 'item' as const, e }))
    const out: Array<{ kind: 'street' | 'item'; street?: string; e?: Establishment }> = []
    let lastStreet = ''
    for (const e of filtered) {
      const street = String(e.street ?? 'Sem rua').trim() || 'Sem rua'
      if (street !== lastStreet) {
        out.push({ kind: 'street', street })
        lastStreet = street
      }
      out.push({ kind: 'item', e })
    }
    return out
  }, [filtered, sortKey])

  const stats = useMemo(() => {
    const total = state.establishments.length
    const by: Record<EstablishmentStatus, number> = {
      novo: 0,
```

```
      visitado: 0,
      confirmado: 0,
      aprovado: 0,
      recusou: 0,
      sem_resposta: 0,
    }
    for (const e of state.establishments) by[e.status] = (by[e.status] ?? 0) + 1
    return { total, by }
  }, [state.establishments])

  const selected = useMemo(() => {
    const id = String(selectedPlaceId ?? '').trim()
    if (!id) return null
    return state.establishments.find((e) => e.placeId === id) ?? null
  }, [state.establishments, selectedPlaceId])

  const progressPct = progress.active && progress.total > 0 ? Math.min(100, Math.round((progress.done / progress.total)
* 100)) : null
  const progressDone = progress.total > 0 ? Math.min(progress.done, progress.total) : progress.done
  const progressLabel = progress.total > 0 ? `${progressDone}/${progress.total} • ${progressPct ?? 0}%` : progressDone >
0 ? String(progressDone) : '…'

  return (
    <div className="app">
      <style>{CSS}</style>
      <div className="top">
        <div className="title">
          <h1>Prospector</h1>
          <div className="subtitle">Importe do Google Maps e acompanhe seu funil</div>
        </div>
        <div className="kpis">
          {installPrompt ? (
            <button
              className="btn"
              onClick={async () => {
                const ev = installPrompt as unknown as { prompt?: () => Promise<void>; userChoice?: Promise<{ outcome?:
string }> }
                if (!ev?.prompt) return
                await ev.prompt()
                const choice = await ev.userChoice
                setInstallPrompt(null)
                if (choice?.outcome === 'accepted') toastNow('Instalação iniciada')
              }}
            >
              Instalar
            </button>
          ) : null}
          <div className="chip">
            Total <b>{stats.total}</b>
          </div>
          <div className="chip">
            Novos <b>{stats.by.novo}</b>
          </div>
          <div className="chip">
            Visitados <b>{stats.by.visitado}</b>
          </div>
          <div className="chip">
            Confirmados <b>{stats.by.confirmado}</b>
          </div>
          <div className="chip">
            Aprovados <b>{stats.by.aprovado}</b>
          </div>
          <div className="chip">
            Recusou <b>{stats.by.recusou}</b>
          </div>
        </div>
      </div>

      <div className="layout">
        <div className="panel">
          <div className="panelHeader">
            <div className="toolbar">
```

```
              <div className="field" style={{ minWidth: 220 }}>
                <span>Buscar</span>
                <input className="input" placeholder="Nome, rua, bairro…" value={filter} onChange={(e) =>
setFilter(e.target.value)} />
              </div>
              <div className="field" style={{ minWidth: 170, flex: '0 0 auto' }}>
                <span>Status</span>
                <select className="select" value={statusFilter} onChange={(e) => setStatusFilter(e.target.value as
EstablishmentStatus | 'all')}>
                  <option value="all">Todos</option>
                  <option value="novo">Novo</option>
                  <option value="visitado">Visitado</option>
                  <option value="confirmado">Confirmado</option>
                  <option value="aprovado">Aprovado</option>
                  <option value="recusou">Recusou</option>
                  <option value="sem_resposta">Sem resposta</option>
                </select>
              </div>
              <div className="field" style={{ minWidth: 170, flex: '0 0 auto' }}>
                <span>Ordenar</span>
                <select className="select" value={sortKey} onChange={(e) => setSortKey(e.target.value as typeof
sortKey)}>
                  <option value="updated_desc">Atualizados (recentes)</option>
                  <option value="last_visit_desc">Visitados (recentes)</option>
                  <option value="name_asc">Nome (A–Z)</option>
                  <option value="status">Status</option>
                  <option value="street_number">Rua e número</option>
                </select>
              </div>
              <button className="btn" onClick={() => setOnlyWithPhone((v) => !v)}>
                {onlyWithPhone ? 'Com telefone: ON' : 'Com telefone'}
              </button>
              <button className="btn" onClick={() => setOnlyMissingContact((v) => !v)}>
                {onlyMissingContact ? 'Sem contato: ON' : 'Sem contato'}
              </button>
            </div>
            <div className="sep" />
              <div className="row">
                <div className="field" style={{ minWidth: 220 }}>
                  <span>Cidade</span>
                  <input className="input" value={city} onChange={(e) => setCity(e.target.value)} placeholder="Ex: Belo
Horizonte" />
                </div>
                <div className="field" style={{ minWidth: 280 }}>
                  <span>Estado (UF)</span>
                  <input className="input" value={uf} onChange={(e) => setUf(e.target.value)} placeholder="Ex: MG" />
                </div>
                <div className="field" style={{ minWidth: 200, flex: '0 0 auto' }}>
                  <span>Segmento</span>
                  <select className="select" value={presetKey} onChange={(e) => setPresetKey(e.target.value)}>
                    {PRESETS.map((p) => (
                    <option key={p.key} value={p.key}>
                      {p.label}
                    </option>
                  ))}
                </select>
              </div>
              <button className="btn btnPrimary" onClick={runImport} disabled={loading}>
                {loading ? 'Importando…' : 'Importar do Google Maps'}
              </button>
              <button className="btn" onClick={runImportItabirito} disabled={loading}>
                {loading ? 'Importando…' : 'Importar Itabirito (cidade inteira)'}
              </button>
              <button
                className="btn"
                onClick={() => downloadText(`prospector-${new Date().toISOString().slice(0, 10)}.json`,
exportJson(state.establishments))}
              >
                Exportar JSON
              </button>
              <button
                className="btn btnDanger"
```

```
                onClick={() => {
                  const ok = window.confirm('Excluir todos os estabelecimentos salvos neste dispositivo?')
                  if (!ok) return
                  const next = { establishments: [] as Establishment[] }
                  setState(next)
                  saveState(next)
                  setSelectedPlaceId(null)
                  toastNow('Dados excluídos')
                }}
              >
                Excluir tudo
              </button>
              <button
                className="btn"
                onClick={() => {
                  const csv = toCsv(state.establishments)
                  const blob = new Blob([csv], { type: 'text/csv;charset=utf-8' })
                  const url = URL.createObjectURL(blob)
                  const a = document.createElement('a')
                  a.href = url
                  a.download = `prospector-${new Date().toISOString().slice(0, 10)}.csv`
                  document.body.appendChild(a)
                  a.click()
                  a.remove()
                  URL.revokeObjectURL(url)
                }}
              >
                Exportar CSV
              </button>
              <label className="field" style={{ minWidth: 200, flex: '0 0 auto' }}>
                <span>Importar JSON</span>
                <input
                  className="input"
                  type="file"
                  accept="application/json"
                  onChange={async (e) => {
                    const file = e.target.files?.[0]
                    if (!file) return
                    const text = await file.text()
                    const imported = importJson(text)
                    upsertMany(imported)
                    e.target.value = ''
                  }}
                />
              </label>
            </div>
            {progress.active ? (
              <div className="progressWrap">
                <div className="progressTop">
                  <div className="progressMsg">{progress.message || 'Importando…'}</div>
                  <div className="progressPct">{progressLabel}</div>
                </div>
                <div className="progressTrack">
                  {progressPct === null ? (
                    <div className="progressIndeterminate" />
                  ) : (
                    <div className="progressFill" style={{ width: `${progressPct}%` }} />
                  )}
                </div>
              </div>
            ) : null}
            {error ? <div style={{ marginTop: 10, color: 'rgba(254,202,202,.95)' }}>{error}</div> : null}
            {toast ? <div style={{ marginTop: 10, color: 'rgba(187,247,208,.98)' }}>{toast}</div> : null}
          </div>

          <div className="panelBody">
            <div className="listHeader">
              <div className="muted">Resultados: <b style={{ color: 'var(--text)' }}>{filtered.length}</b></div>
              <div className="muted2">Clique em um item para ver detalhes</div>
            </div>
            <div className="sep" />
            <div className="listScroll">
```

```
              <div className="list">
                {listNodes.map((n) => {
                  if (n.kind === 'street') return <div key={`street:${n.street}`} className="streetHeader">{n.street}</div>
                  const e = n.e!
                  const active = selectedPlaceId === e.placeId
                  const badge = statusColor(e.status)
                  const hasPhone = Boolean(String(e.phone ?? '').trim())
                  const hasContact = Boolean(String(e.contactName ?? '').trim() || String(e.contactPhone ?? '').trim())
                  return (
                    <div
                      key={e.placeId}
                      className={`item ${active ? 'itemActive' : ''}`}
                      onClick={() => setSelectedPlaceId(e.placeId)}
                      role="button"
                      tabIndex={0}
                      onKeyDown={(ev) => {
                        if (ev.key === 'Enter' || ev.key === ' ') setSelectedPlaceId(e.placeId)
                      }}
                    >
                      <div className="itemTop">
                        <span className={`badge ${badge}`}>{statusLabel(e.status)}</span>
                        <div style={{ display: 'grid', gap: 3 }}>
                          <div className="itemName">{e.name}</div>
                          <div className="itemAddr">{e.formattedAddress || `${e.street ?? ''} ${e.number ?? ''}`.trim()}</div>
                        </div>
                      </div>
                      <div className="itemMeta">
                        <span>{hasPhone ? '☎️ telefone' : 'sem telefone'}</span>
                        <span>{hasContact ? '👤 contato' : 'sem contato'}</span>
                        <span>Atualizado: {formatIsoDate(e.updatedAt)}</span>
                      </div>
                    </div>
                  )
                })}
              </div>
            </div>
          </div>
        </div>

        <div className="panel">
          <div className="panelHeader">
            <div className="title">
              <div style={{ fontWeight: 800 }}>Detalhes</div>
              {selected ? <div className="subtitle">{selected.name}</div> : <div className="subtitle">Selecione um estabelecimento</div>}
            </div>
          </div>
          <div className="panelBody">
            {!selected ? (
              <div className="hint">
                Use a lista à esquerda para selecionar um estabelecimento. Aqui você consegue:
                <div style={{ height: 8 }} />
                <div className="muted2">- mudar status (confirmado/aprovado/recusou)</div>
                <div className="muted2">- salvar contato (nome e número)</div>
                <div className="muted2">- abrir Maps / WhatsApp / copiar dados</div>
              </div>
            ) : (
              <>
                <div className="actions">
                  <span className={`badge ${statusColor(selected.status)}`}>{statusLabel(selected.status)}</span>
                  <button className="btn" onClick={() => updateStatus(selected.placeId, 'visitado')}>Visitado hoje</button>
                  <button className="btn" onClick={() => updateStatus(selected.placeId, 'confirmado')}>Confirmado</button>
                  <button className="btn" onClick={() => updateStatus(selected.placeId, 'aprovado')}>Aprovado</button>
                  <button className="btn btnDanger" onClick={() => updateStatus(selected.placeId, 'recusou')}>Recusou</button>
                  <button className="btn" onClick={() => updateStatus(selected.placeId, 'sem_resposta')}>Sem resposta</button>
                </div>
```

```
                <div className="sep" />

                <div className="split">
                  <div>
                    <div className="muted">Endereço</div>
                    <div style={{ marginTop: 6, lineHeight: 1.35 }}>{selected.formattedAddress || '-'}</div>
                    <div style={{ marginTop: 10 }} className="actions">
                      <button
                        className="btn"
                        onClick={async () => {
                          const ok = await copyText(selected.formattedAddress)
                          if (ok) toastNow('Endereço copiado')
                        }}
                      >
                        Copiar endereço
                      </button>
                      {selected.googleMapsUrl ? (
                        <a className="btn" href={selected.googleMapsUrl} target="_blank" rel="noreferrer">Abrir no
Maps</a>
                      ) : null}
                      {selected.website ? (
                        <a className="btn" href={selected.website} target="_blank" rel="noreferrer">Site</a>
                      ) : null}
                    </div>
                  </div>

                  <div>
                    <div className="muted">Telefone (Google)</div>
                    <div style={{ marginTop: 6, lineHeight: 1.35 }}>{selected.phone || '-'}</div>
                    <div style={{ marginTop: 10 }} className="actions">
                      <button
                        className="btn"
                        onClick={async () => {
                          const ok = await copyText(selected.phone ?? '')
                          if (ok) toastNow('Telefone copiado')
                        }}
                      >
                        Copiar telefone
                      </button>
                      {selected.phone ? <a className="btn" href={`tel:${selected.phone}`}>Ligar</a> : null}
                      {selected.phone ? (
                        <a className="btn" href={toWhatsAppUrl(selected.phone) ?? '#'} target="_blank" rel="noreferrer">
                          WhatsApp
                        </a>
                      ) : null}
                    </div>
                  </div>
                </div>

                <div className="sep" />

                <div className="split">
                  <label className="field">
                    <span>Nome (contato)</span>
                    <input
                      className="input"
                      value={selected.contactName ?? ''}
                      onChange={(ev) => updateOne(selected.placeId, { contactName: ev.target.value })}
                      placeholder="Ex: Ana"
                    />
                  </label>
                  <label className="field">
                    <span>Número (contato)</span>
                    <input
                      className="input"
                      value={selected.contactPhone ?? ''}
                      onChange={(ev) => updateOne(selected.placeId, { contactPhone: ev.target.value })}
                      placeholder="Ex: (31) 9xxxx-xxxx"
                    />
                  </label>
                </div>
```

```
                <div style={{ marginTop: 10 }} className="actions">
                  <button
                    className="btn"
                    onClick={async () => {
                      const ok = await copyText(`${selected.contactName ?? ''} ${selected.contactPhone ?? ''}`.trim())
                      if (ok) toastNow('Contato copiado')
                    }}
                  >
                    Copiar contato
                  </button>
                  {selected.contactPhone ? (
                    <a className="btn" href={toWhatsAppUrl(selected.contactPhone, `Olá! Tudo bem?`) ?? '#'}
target="_blank" rel="noreferrer">
                      WhatsApp (contato)
                    </a>
                  ) : null}
                </div>

                <div style={{ marginTop: 10 }}>
                  <label className="field">
                    <span>Observações</span>
                    <textarea
                      className="textarea"
                      value={selected.notes ?? ''}
                      onChange={(ev) => updateOne(selected.placeId, { notes: ev.target.value })}
                      placeholder="Ex: falou com a gerente; pediu para voltar semana que vem…"
                    />
                  </label>
                </div>

                {Array.isArray(selected.segments) && selected.segments.length > 0 ? (
                  <>
                    <div className="sep" />
                    <div>
                      <div className="muted">Segmentos (Google)</div>
                      <div style={{ marginTop: 8 }} className="actions">
                        {selected.segments.map((s) => (
                          <span key={s} className="badge gray">
                            {s}
                          </span>
                        ))}
                      </div>
                    </div>
                  </>
                ) : null}

                <div className="sep" />

                <div className="actions">
                  <button
                    className="btn"
                    onClick={async () => {
                      try {
                        const det = await placesDetails(selected.placeId)
                        const d = det.result
                        const parts = extractAddressParts(d)
                        updateOne(selected.placeId, {
                          name: (d.name ?? selected.name).trim(),
                          formattedAddress: (d.formatted_address ?? selected.formattedAddress).trim(),
                          street: parts.street,
                          number: parts.number,
                          city: parts.city,
                          state: parts.state,
                          postalCode: parts.postalCode,
                          phone: (d.formatted_phone_number ?? d.international_phone_number ?? '').trim() || null,
                          website: (d.website ?? '').trim() || null,
                          googleMapsUrl: (d.url ?? '').trim() || null,
                          fetchedAt: new Date().toISOString(),
                        })
                        toastNow('Atualizado do Google')
                      } catch {
```

```
                        toastNow('Falha ao atualizar')
                      }
                    }}
                  >
                    Atualizar do Google
                  </button>
                  <button
                    className="btn"
                    onClick={() => {
                      const next = {
                        establishments: state.establishments.filter((x) => x.placeId !== selected.placeId),
                      }
                      setState(next)
                      saveState(next)
                      setSelectedPlaceId(null)
                      toastNow('Removido')
                    }}
                  >
                    Remover
                  </button>
                </div>

                <div style={{ marginTop: 10 }} className="muted2">
                  Última visita: {selected.lastVisitAt ? formatIsoDate(selected.lastVisitAt) : '-'}
                  <br />
                  Atualizado: {formatIsoDate(selected.updatedAt)}
                </div>
              </>
            )}
          </div>
        </div>
      </div>
    </div>
  )
}
```

prospector/src/importMeta.d.ts

```
interface ImportMetaEnv {
  readonly PROD: boolean
  readonly BASE_URL: string
}

interface ImportMeta {
  readonly env: ImportMetaEnv
}
```

prospector/src/lib/address.ts

```
import type { PlacesDetails } from './api'

function findComponent(components: PlacesDetails['address_components'] | undefined, type: string) {
  const arr = Array.isArray(components) ? components : []
  for (const c of arr) {
    const t = Array.isArray(c.types) ? c.types : []
    if (t.includes(type)) return c
  }
  return null
}

export function extractAddressParts(details: PlacesDetails) {
  const street = findComponent(details.address_components, 'route')
  const number = findComponent(details.address_components, 'street_number')
  const city = findComponent(details.address_components, 'administrative_area_level_2') ??
findComponent(details.address_components, 'locality')
  const state = findComponent(details.address_components, 'administrative_area_level_1')
  const postalCode = findComponent(details.address_components, 'postal_code')

  return {
    street: (street?.long_name ?? '').trim() || null,
    number: (number?.long_name ?? '').trim() || null,
    city: (city?.long_name ?? '').trim() || null,
    state: (state?.short_name ?? state?.long_name ?? '').trim() || null,
    postalCode: (postalCode?.long_name ?? '').trim() || null,
  }
}
```

prospector/src/lib/api.ts

```
export type PlacesTextResult = {
  place_id: string
  name: string
  formatted_address?: string
  geometry?: { location?: { lat?: number; lng?: number } }
  types?: string[]
}

export type PlacesDetails = {
  place_id: string
  name: string
  formatted_address?: string
  adr_address?: string
  url?: string
  website?: string
  formatted_phone_number?: string
  international_phone_number?: string
  geometry?: { location?: { lat?: number; lng?: number } }
  address_components?: Array<{ long_name?: string; short_name?: string; types?: string[] }>
  types?: string[]
}

export type Health = {
  ok: boolean
  hasKey?: boolean
}

async function getJson<T>(url: string): Promise<T> {
  let res: Response
  try {
    res = await fetch(url)
  } catch {
    throw new Error('Backend indisponível. Verifique se o Prospector está publicado e tente novamente.')
  }

  const text = await res.text()
  if (!text.trim()) {
    if (String(url).startsWith('/api')) {
      throw new Error(`Backend indisponível (${res.status}). Verifique se o Prospector está publicado e tente novamente.`)
    }
    throw new Error(`Resposta vazia do backend (${res.status}).`)
  }

  let parsed: unknown
  try {
    parsed = JSON.parse(text) as unknown
  } catch {
    throw new Error(`Resposta inválida do backend (${res.status}).`)
  }

  if (!res.ok) {
    const message = (() => {
      if (!parsed || typeof parsed !== 'object') return null
      const obj = parsed as Record<string, unknown>
      const m = obj.message
      return typeof m === 'string' && m.trim() ? m.trim() : null
    })()
    throw new Error(message ? `HTTP ${res.status}: ${message}` : `HTTP ${res.status}`)
  }

  return parsed as T
}

export async function geocode(address: string) {
  return await getJson<{ results: Array<{ geometry?: { location?: { lat?: number; lng?: number } } }> }>(
    `/api/geocode?address=${encodeURIComponent(address)}`
  )
}
```

```
export async function health() {
  return await getJson<Health>('/api/health')
}

export async function placesTextSearch(input: { query: string; pageToken?: string | null; location?: string | null;
radius?: number | null }) {
  const q = encodeURIComponent(input.query)
  const pt = input.pageToken ? `&pagetoken=${encodeURIComponent(input.pageToken)}` : ''
  const loc = input.location ? `&location=${encodeURIComponent(input.location)}` : ''
  const rad = input.radius && Number.isFinite(input.radius) ? `&radius=${encodeURIComponent(String(input.radius))}` : ''
  return await getJson<{ results: PlacesTextResult[]; next_page_token?: string }>(`/api/places/textsearch?
query=${q}${pt}${loc}${rad}`)
}

export async function placesDetails(placeId: string) {
  return await getJson<{ result: PlacesDetails }>(`/api/places/details?place_id=${encodeURIComponent(placeId)}`)
}
```

prospector/src/lib/id.ts

```
export function newId() {
  if (typeof crypto !== 'undefined' && 'randomUUID' in crypto) return crypto.randomUUID()
  return `${Date.now()}-${Math.random().toString(16).slice(2)}`
}
```

prospector/src/lib/storage.ts

```
import type { Establishment } from '../types'

const STORAGE_KEY = 'prospector:v1'

type Persisted = {
  establishments: Establishment[]
}

export function loadState(): Persisted {
  const raw = localStorage.getItem(STORAGE_KEY)
  if (!raw) return { establishments: [] }
  try {
    const parsed = JSON.parse(raw) as unknown
    if (!parsed || typeof parsed !== 'object') return { establishments: [] }
    const obj = parsed as Record<string, unknown>
    const rows = obj.establishments
    if (!Array.isArray(rows)) return { establishments: [] }
    return { establishments: rows as Establishment[] }
  } catch {
    return { establishments: [] }
  }
}

export function saveState(state: Persisted) {
  localStorage.setItem(STORAGE_KEY, JSON.stringify(state))
}

export function exportJson(establishments: Establishment[]) {
  return JSON.stringify({ establishments, exportedAt: new Date().toISOString() }, null, 2)
}

export function importJson(json: string): Establishment[] {
  const parsed = JSON.parse(json) as unknown
  if (!parsed || typeof parsed !== 'object') return []
  const obj = parsed as Record<string, unknown>
  const rows = obj.establishments
  if (!Array.isArray(rows)) return []
  return rows as Establishment[]
}
```

prospector/src/main.tsx

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import { App } from './App'

if (import.meta.env.PROD && 'serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register(`${import.meta.env.BASE_URL}sw.js`).catch(() => null)
  })
}

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
```

prospector/src/types.ts

```
export type EstablishmentStatus =
  | 'novo'
  | 'visitado'
  | 'confirmado'
  | 'aprovado'
  | 'recusou'
  | 'sem_resposta'

export type Establishment = {
  id: string
  placeId: string
  name: string
  formattedAddress: string
  street: string | null
  number: string | null
  city: string | null
  state: string | null
  postalCode: string | null
  lat: number | null
  lng: number | null
  types: string[]
  segments?: string[]
  googleMapsUrl: string | null
  phone: string | null
  website: string | null
  fetchedAt: string
  status: EstablishmentStatus
  contactName: string | null
  contactPhone: string | null
  notes: string | null
  lastVisitAt: string | null
  updatedAt: string
  createdAt: string
}

export type ImportPreset = {
  key: string
  label: string
  queries: string[]
}
```

prospector/tsconfig.json

```
{
  "compilerOptions": {
    "target": "ES2022",
    "useDefineForClassFields": true,
    "lib": ["ES2022", "DOM", "DOM.Iterable"],
    "module": "ESNext",
    "skipLibCheck": true,
    "moduleResolution": "Bundler",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
    "jsx": "react-jsx",
    "strict": true
  },
  "include": ["src"]
}
```

prospector/vite.config.ts

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig(({ mode }) => {
  const basePath = mode === 'production' ? '/142555787po/' : '/'
  return {
    base: basePath,
    plugins: [react()],
    build: {
      outDir: '../dist/142555787po',
      emptyOutDir: false,
    },
    server: {
      proxy: {
        '/api': {
          target: 'http://localhost:8787',
          changeOrigin: true,
        },
      },
    },
  }
})
```

ROTEIRO_VENDAS.md

```
# Roteiro de Vendas Presencial - SMagenda

Este guia foi desenhado para te ajudar a abordar estabelecimentos (Lava Jatos, Estética Automotiva, Salões de Beleza, Barbearias, Clínicas, etc.) e vender o sistema SMagenda.

---

## 1. Preparação (Antes de entrar)

*   **Aparência:** Vista-se de forma profissional, mas compatível com o ambiente (esporte fino ou casual arrumado).
*   **Material:**
    *   Tenha o seu celular com bateria e internet.
    *   **CRUCIAL:** Deixe a **Página Pública de Agendamento** (a que o cliente final vê) aberta no navegador do seu celular. O visual dela é o seu maior argumento.
    *   Se tiver, leve um cartão de visita ou um folheto simples com um QR Code que leva para essa página de demonstração.

---

## 2. A Abordagem Inicial (Passando pelo "Guardião")

Normalmente, quem te atende primeiro é um funcionário, recepcionista ou atendente. O objetivo aqui não é vender para ele, mas **conquistar a simpatia dele** para chegar no decisor (Dono ou Gerente).

**Você:** "Olá, bom dia/boa tarde! Tudo bem? Bastante movimento hoje?" (Sorria e seja simpático).
**Funcionário:** "Tudo bem..."
**Você:** "Meu nome é [Seu Nome], eu trabalho com tecnologia para agendamentos aqui da região. O responsável ou gerente está por aí? Eu queria dar uma palavrinha rápida com ele, coisa de 2 minutos."

### Cenário A: O Dono NÃO está
**Funcionário:** "Ele não está agora." ou "Só vem mais tarde."

**O que fazer:** Não vá embora de mãos vazias.
**Você:** "Entendi. Sem problemas. Qual o nome dele, por favor?"
**Funcionário:** "É o Carlos."
**Você:** "Legal. Você sabe qual o melhor horário para eu encontrar o Carlos aqui?"
*(Anote a resposta)*
**Você:** "E você, como se chama?"
**Funcionário:** "Ana."
**Você:** "Prazer, Ana. Vocês usam WhatsApp para agendar os clientes aqui?"
**Funcionário:** "Sim, usamos."
**Você:** "Imagino que deve ser uma loucura responder todo mundo na hora, né? As vezes o cliente fica esperando..."
**Funcionário:** "Nossa, sim, é bem corrido."
**Você:** "Pois é, eu tenho uma solução justamente pra isso, pra facilitar a vida de vocês. Vou deixar meu cartão/contato aqui pra ele. Você pode entregar para o Carlos e dizer que eu passei aqui? Diga que é sobre o sistema que acaba com a demora no WhatsApp."

*Dica de Ouro:* Se conseguir, peça o WhatsApp do estabelecimento (se for o comercial) e já mande um "Oi Carlos, passei aí e falei com a Ana..." logo em seguida.

---

## 3. O Pitch de Vendas (Falando com o Dono)

Quando encontrar o dono (Carlos), seja direto e respeite o tempo dele.

**Você:** "Olá Carlos, tudo bem? Meu nome é [Seu Nome]. Eu vejo que vocês têm um serviço excelente aqui. Eu desenvolvi um sistema chamado SMagenda focado em [Lava Jatos/Salões] para automatizar os agendamentos. Você tem 2 minutinhos pra eu te mostrar como funciona na prática?"

*(Se ele disser que está ocupado, pergunte quando pode voltar. Se der abertura, prossiga)*

### Passo 1: O Problema (A Dor)
**Você:** "Carlos, hoje, quando um cliente quer agendar, ele manda mensagem no WhatsApp, certo?"
**Dono:** "Sim."
**Você:** "E acontece de vocês estarem ocupados atendendo e demorarem pra responder? Ou do cliente ficar perguntando 'tem horário tal?', 'e tal dia?', num vai e vem de mensagens?"
**Dono:** "Acontece direto."
**Você:** "Isso faz você perder dinheiro e tempo. O cliente ansioso desiste e vai no concorrente."
```

```
### Passo 2: A Solução (A Demonstração)
*Saque o celular com a página pública aberta.*

**Você:** "Olha como eu resolvi isso. Com o SMagenda, você manda um link pro cliente (ou põe no Instagram). O cliente clica e vê isso aqui:"
*(Mostre a tela do celular para ele)*
**Você:** "Ele escolhe o serviço (ex: Lavagem Completa), vê os dias e horários livres que VOCÊ definiu, e agenda sozinho. Em 30 segundos. O agendamento cai direto no seu celular e você já sabe quem vem."

### Passo 3: Diferenciais (Mata Objeções)
*   **Controle:** "Você define quantas vagas tem por dia. Se for serviço de dia inteiro, o sistema bloqueia o dia."
*   **Lembretes:** "O sistema ajuda a lembrar o cliente (se tiver email/zap automático)."
*   **Organização:** "Você para de usar papel e caneta e tem tudo no histórico."

### Passo 4: O Fechamento (A Oferta)
**Você:** "Eu estou cadastrando os primeiros parceiros aqui da região. Eu queria muito que você testasse. É muito simples de usar."

*(Aqui você usa sua estratégia de preço/teste)*
*   *Opção A (Teste Grátis):* "Posso liberar 7 dias grátis pra você sentir a diferença? Eu mesmo configuro seu cardápio de serviços agora."
*   *Opção B (Preço Baixo):* "Custa menos que uma lavagem/corte por mês."

---

## 4. Venda Consultiva (Quando você NÃO sabe se serve)

Quando você aborda um negócio novo (ex.: estúdio de yoga), o objetivo é **descobrir necessidades** em 5–10 minutos e só então encaixar o SMagenda (ou dizer que não encaixa 100% e propor um plano B).

### 4.1. A regra de ouro

1) **Pergunte antes de apresentar.**
2) **Repita o problema com as palavras do cliente.**
3) **Mostre só 1–2 benefícios que batem exatamente com a dor.**
4) **Se não encaixar, seja honesto e proponha o que dá para resolver.**

### 4.2. Perguntas rápidas de diagnóstico (copiar e colar)

Use 6–10 perguntas. As respostas já te dizem se o SMagenda resolve.

**Sobre o atendimento**
1) Hoje vocês agendam mais por WhatsApp, Instagram, telefone ou presencial?
2) Quem responde as mensagens? (dono, recepção, professor)
3) Quantas solicitações de agendamento chegam por dia/semana?

**Sobre o tipo de agenda**
4) Vocês agendam **aulas particulares** ou **turmas** com várias pessoas no mesmo horário?
5) Os horários são por profissional/sala? Tem mais de um professor ao mesmo tempo?
6) Vocês precisam de lista de espera, confirmação e lembrete automático?

**Sobre regras e cobrança**
7) Tem taxa de falta/no-show? Como vocês lidam com cancelamentos?
8) Vocês vendem planos/pacotes/mensalidade? Como controlam isso hoje?

**Sobre o que mais dói**
9) O que mais te dá trabalho hoje: responder, organizar, lembrar o cliente, ou evitar furos?
10) Se você pudesse resolver 1 coisa esse mês, qual seria?

### 4.3. Como “encaixar” o SMagenda sem forçar

**Se o negócio é por atendimento individual (encaixa bem):**
- Aula experimental (1:1)
- Avaliação inicial
- Aulas particulares
- Sessões (pilates, estética, terapia, etc.)

**Se o negócio é por turmas (encaixe parcial):**
- O SMagenda resolve muito bem a parte de **aula experimental / avaliação / particular**.
- Para **turmas com vagas por horário**, pode precisar de uma solução específica. Mesmo assim, muitos estúdios usam o SMagenda para captar e organizar novos alunos e marcar atendimentos individuais.
```

---

5. Modelos de Mensagens (WhatsApp/Instagram)

5.1. Primeira mensagem (curta, sem parecer spam)

"Oi, tudo bem? Eu sou [Seu Nome]. Vi o [Nome do Estúdio] aqui na região e queria te fazer uma pergunta rápida: hoje vocês agendam mais por WhatsApp/Instagram ou já usam algum sistema?"

5.2. Mensagem consultiva (quando respondem “WhatsApp”)

"Entendi. E vocês trabalham mais com turmas ou com atendimentos individuais (aula experimental/avaliação/aula particular)? Pergunto porque eu ajudo estúdios a reduzir o vai-e-vem de mensagens e o cliente consegue agendar sozinho por um link."

5.3. Mensagem de valor (depois do diagnóstico)

"Pelo que você me disse (\"[dor do cliente]\"), dá pra resolver assim: você define seus horários e o cliente agenda sozinho pelo link, sem ficar pedindo horário por mensagem. Posso te mostrar em 2 minutos um exemplo no celular?"

5.4. Follow-up 1 (no dia seguinte)

"Oi [Nome], só passando pra saber se faz sentido eu te mostrar rapidamente. Prometo ser bem objetivo(a): 2 minutos e você já entende se vale a pena."

5.5. Follow-up 2 (encerramento elegante)

"Última mensagem pra não incomodar: se hoje não for prioridade, tudo bem. Se você quiser, eu te mando um link de demonstração e você olha quando der."

---

6. Cenários Prontos (com exemplos)

6.1. Estúdio de Yoga (o seu exemplo)

Objetivo: descobrir se é turma, particular ou misto.

Você (mensagem):
"Oi! Uma dúvida rápida: vocês vendem mais turmas fixas com mensalidade ou fazem aulas particulares/avaliação por horário marcado?"

Se responder “turmas/mensalidade”:
"Perfeito. Nesse caso, normalmente o maior gargalo é captar e organizar o aluno novo: aula experimental, avaliação inicial e remarcações. O SMagenda resolve exatamente isso com um link de agendamento e lembretes, e vocês continuam com as turmas do jeito que já funciona. Quer que eu te mostre como ficaria a aula experimental no link?"

Se responder “aulas particulares/avaliação”:
"Aí encaixa 100%. Você define horários e dias, e o aluno agenda sozinho por um link (sem troca de mensagens). E você ainda consegue confirmar/cancelar e ter histórico. Posso te mandar um link de demonstração e, se fizer sentido, eu configuro seu primeiro serviço em 5 min."

6.2. Barbearia/Salão (múltiplos profissionais)

Pergunta-chave:
"Vocês têm mais de um profissional atendendo ao mesmo tempo? Cada um tem horários diferentes?"

Gancho:
"Com o link, o cliente escolhe o serviço e o profissional, vê horário livre e agenda sozinho. Isso reduz desistência por demora no WhatsApp."

6.3. Estética/Lava-jato (no-show e regras)

Pergunta-chave:
"O que mais acontece aí: cancelamento em cima da hora ou cliente que não aparece?"

Gancho:
"Dá pra confirmar e lembrar automaticamente, e você fica com histórico organizado."

6.4. Clínica/Consultório (agenda organizada e histórico)

Pergunta-chave:

```text
"Hoje vocês anotam em papel/planilha ou já tem agenda digital?"

Gancho:
"Agenda organizada por período, busca por cliente, e menos erro de anotação."

---

## 7. Objeções Comuns (respostas curtas)

**“Eu já me viro no WhatsApp.”**
"Total. A ideia não é complicar: é reduzir a troca de mensagens. O link faz o cliente escolher sozinho e cai pronto pra
você."

**“Meus clientes são ruins de tecnologia.”**
"Por isso o link é forte: parece um ‘cardápio de horários’. É bem mais simples do que ficar perguntando horário por
mensagem."

**“Aqui é por turmas/mensalidade.”**
"Perfeito. Então vamos focar onde dá mais retorno: aula experimental/avaliação/particular e novos alunos. Isso
normalmente já paga o sistema."

**“Depois eu vejo.”**
"Fechado. Me diz só: hoje sua prioridade é reduzir mensagens, evitar furos, ou organizar a agenda? Se eu souber isso, eu
te mando só o que é relevante."

---

## Resumo de Dicas
1.  **Não venda "Software", venda "Tempo Livre" e "Organização".** O dono não quer aprender mexer em computador, ele
quer parar de perder tempo no WhatsApp.
2.  **Foque no Visual:** A tela de agendamento bonita no celular vende mais que mil palavras.
3.  **Seja breve:** Eles estão trabalhando.
4.  **Pegue o contato:** Nunca saia sem o telefone do dono para fazer o *follow-up* (cobrar uma resposta depois).
```

scheduling-system-doc (1).md

~~~
- Botão "Compartilhar no WhatsApp"
- Botão "Ir para minha agenda"

---

## 🔧 Funcionalidades Avançadas Detalhadas

### 1️⃣ **Lógica de Validação de Limites por Plano**

#### Como Funciona

Cada vez que uma ação é realizada, o sistema verifica se o usuário tem permissão baseado no plano dele:

```javascript
// services/planLimitsService.js

// Definição dos limites por plano
const PLAN_LIMITS = {
  free: {
    agendamentos_mes: 30,
    funcionarios: 0,
    servicos: 3,
    mensagens_auto: false,
    relatorios_avancados: false,
    personalizacao_pagina: false
  },
  basic: {
    agendamentos_mes: -1, // -1 = ilimitado
    funcionarios: 0,
    servicos: -1,
    mensagens_auto: true,
    relatorios_avancados: false,
    personalizacao_pagina: false
  },
  pro: {
    agendamentos_mes: -1,
    funcionarios: 2,
    servicos: -1,
    mensagens_auto: true,
    relatorios_avancados: true,
    personalizacao_pagina: true
  },
  team: {
    agendamentos_mes: -1,
    funcionarios: 5,
    servicos: -1,
    mensagens_auto: true,
    relatorios_avancados: true,
    personalizacao_pagina: true
  },
  enterprise: {
    agendamentos_mes: -1,
    funcionarios: -1, // ilimitado
    servicos: -1,
    mensagens_auto: true,
    relatorios_avancados: true,
    personalizacao_pagina: true
  }
};

// Função para verificar limite de agendamentos
async function checkAgendamentoLimit(usuarioId) {
  // Busca o usuário e seu plano
  const { data: usuario } = await supabase
    .from('usuarios')
    .select('plano, status_pagamento')
    .eq('id', usuarioId)
    .single();

  // Verifica status do pagamento
~~~

```
  if (usuario.status_pagamento !== 'ativo') {
    throw new Error('Assinatura inativa. Regularize seu pagamento.');
  }

  const limite = PLAN_LIMITS[usuario.plano].agendamentos_mes;

  // Se ilimitado, libera
  if (limite === -1) return true;

  // Conta agendamentos do mês atual
  const iniciodoMes = new Date();
  iniciodoMes.setDate(1);
  iniciodoMes.setHours(0, 0, 0, 0);

  const { count } = await supabase
    .from('agendamentos')
    .select('*', { count: 'exact', head: true })
    .eq('usuario_id', usuarioId)
    .gte('criado_em', iniciodoMes.toISOString());

  if (count >= limite) {
    throw new Error(`Limite de ${limite} agendamentos/mês atingido. Faça upgrade do seu plano.`);
  }

  return true;
}

// Função para verificar limite de funcionários
async function checkFuncionarioLimit(usuarioId) {
  const { data: usuario } = await supabase
    .from('usuarios')
    .select('plano, status_pagamento')
    .eq('id', usuarioId)
    .single();

  if (usuario.status_pagamento !== 'ativo') {
    throw new Error('Assinatura inativa.');
  }

  const limite = PLAN_LIMITS[usuario.plano].funcionarios;

  // Se ilimitado, libera
  if (limite === -1) return true;

  // Se não permite funcionários (FREE, BASIC)
  if (limite === 0) {
    throw new Error('Seu plano não permite funcionários. Faça upgrade para PRO.');
  }

  // Conta funcionários ativos
  const { count } = await supabase
    .from('funcionarios')
    .select('*', { count: 'exact', head: true })
    .eq('usuario_master_id', usuarioId)
    .eq('ativo', true);

  if (count >= limite) {
    throw new Error(`Limite de ${limite} funcionários atingido. Faça upgrade para TEAM ou ENTERPRISE.`);
  }

  return true;
}

// Função para verificar funcionalidade
function checkFeatureAccess(plano, feature) {
  const hasAccess = PLAN_LIMITS[plano][feature];

  if (!hasAccess) {
    const upgradeTo = getMinimumPlanForFeature(feature);
    throw new Error(`Funcionalidade disponível apenas no plano ${upgradeTo.toUpperCase()}. Faça upgrade.`);
  }
```

~~~
  return true;
}

// Função auxiliar para descobrir plano mínimo necessário
function getMinimumPlanForFeature(feature) {
  const plans = ['free', 'basic', 'pro', 'team', 'enterprise'];

  for (const plan of plans) {
    if (PLAN_LIMITS[plan][feature]) {
      return plan;
    }
  }

  return 'enterprise';
}

// Exportar funções
export {
  checkAgendamentoLimit,
  checkFuncionarioLimit,
  checkFeatureAccess,
  PLAN_LIMITS
};
```

#### Uso nas Rotas da API

```javascript
// routes/agendamentos.js
import { checkAgendamentoLimit } from '../services/planLimitsService.js';

app.post('/api/agendamentos', async (req, res) => {
  try {
    const { usuario_id, servico_id, cliente_nome, data, hora } = req.body;

    // 🔒 VALIDAÇÃO DO LIMITE
    await checkAgendamentoLimit(usuario_id);

    // Se passou, cria o agendamento
    const { data: agendamento } = await supabase
      .from('agendamentos')
      .insert({
        usuario_id,
        servico_id,
        cliente_nome,
        data,
        hora_inicio: hora,
        status: 'confirmado'
      })
      .select()
      .single();

    res.json({ success: true, agendamento });

  } catch (error) {
    // Retorna erro amigável
    res.status(403).json({
      error: error.message,
      needsUpgrade: true,
      currentPlan: 'free',
      suggestedPlan: 'basic'
    });
  }
});
```

```javascript
// routes/funcionarios.js
import { checkFuncionarioLimit } from '../services/planLimitsService.js';

app.post('/api/funcionarios', async (req, res) => {
  try {
~~~

~~~
    const { usuario_master_id, nome, email } = req.body;

    // 🔒 VALIDAÇÃO DO LIMITE
    await checkFuncionarioLimit(usuario_master_id);

    // Cria funcionário...
    const { data: funcionario } = await supabase
      .from('funcionarios')
      .insert({ usuario_master_id, nome, email, ativo: true })
      .select()
      .single();

    res.json({ success: true, funcionario });

  } catch (error) {
    res.status(403).json({
      error: error.message,
      needsUpgrade: true,
      suggestedPlan: 'pro'
    });
  }
});
```

#### Modal de Bloqueio no Frontend

```jsx
// components/UpgradeModal.jsx
import React from 'react';

export default function UpgradeModal({ error, onClose }) {
  return (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50">
      <div className="bg-white rounded-lg p-6 max-w-md w-full mx-4">
        <div className="text-center">
          <div className="text-4xl mb-4">⚠️</div>
          <h2 className="text-xl font-bold mb-2">Limite Atingido</h2>
          <p className="text-gray-600 mb-6">{error.message}</p>

          <div className="bg-blue-50 border border-blue-200 rounded-lg p-4 mb-6">
            <p className="text-sm text-gray-700 mb-2">
              Seu plano atual: <strong>{error.currentPlan.toUpperCase()}</strong>
            </p>
            <p className="text-sm text-gray-700">
              Upgrade sugerido: <strong>{error.suggestedPlan.toUpperCase()}</strong>
            </p>
          </div>

          <div className="flex gap-3">
            <button
              onClick={onClose}
              className="flex-1 px-4 py-2 border border-gray-300 rounded-lg hover:bg-gray-50"
            >
              Voltar
            </button>
            <a
              href="/configuracoes/plano"
              className="flex-1 px-4 py-2 bg-blue-600 text-white rounded-lg hover:bg-blue-700"
            >
              Fazer Upgrade
            </a>
          </div>
        </div>
      </div>
    </div>
  );
}
```

---

### 2️⃣ **Sistema de Notificações para Inadimplentes**
~~~

~~~
#### Fluxo Automático de Cobrança

```javascript
// services/paymentNotificationService.js

// Cron job que roda TODO DIA às 9h da manhã
import cron from 'node-cron';

// Etapas de cobrança
const NOTIFICATION_STAGES = {
  vencimento: 0,        // No dia do vencimento
  atraso_3: 3,          // 3 dias de atraso
  atraso_7: 7,          // 7 dias de atraso
  suspensao_14: 14,     // 14 dias - suspende conta
  cancelamento_30: 30   // 30 dias - cancela conta
};

// Função principal executada diariamente
async function checkPendingPayments() {
  console.log('🔍 Verificando pagamentos pendentes...');

  const hoje = new Date();
  hoje.setHours(0, 0, 0, 0);

  // Busca todas as assinaturas ativas ou inadimplentes
  const { data: assinaturas } = await supabase
    .from('assinaturas')
    .select(`
      *,
      usuarios:usuario_id (
        nome_completo,
        email,
        telefone,
        nome_negocio
      )
    `)
    .in('status', ['ativa', 'inadimplente']);

  for (const assinatura of assinaturas) {
    const vencimento = new Date(assinatura.data_vencimento);
    vencimento.setHours(0, 0, 0, 0);

    const diasAtraso = Math.floor((hoje - vencimento) / (1000 * 60 * 60 * 24));

    // Vencimento hoje
    if (diasAtraso === NOTIFICATION_STAGES.vencimento) {
      await sendPaymentReminder(assinatura, 'vencimento');
    }

    // 3 dias de atraso
    if (diasAtraso === NOTIFICATION_STAGES.atraso_3) {
      await sendPaymentReminder(assinatura, 'atraso_3');
      await updateAssinaturaStatus(assinatura.id, 'inadimplente');
    }

    // 7 dias de atraso
    if (diasAtraso === NOTIFICATION_STAGES.atraso_7) {
      await sendPaymentReminder(assinatura, 'atraso_7');
    }

    // 14 dias - SUSPENDE CONTA
    if (diasAtraso === NOTIFICATION_STAGES.suspensao_14) {
      await suspendAccount(assinatura);
      await sendPaymentReminder(assinatura, 'suspensao');
    }

    // 30 dias - CANCELA CONTA
    if (diasAtraso === NOTIFICATION_STAGES.cancelamento_30) {
      await cancelAccount(assinatura);
      await sendPaymentReminder(assinatura, 'cancelamento');
    }
~~~

```
  }

  console.log('✅ Verificação de pagamentos concluída.');
}

// Função para enviar notificação por email
async function sendPaymentReminder(assinatura, tipo) {
  const usuario = assinatura.usuarios;

  const templates = {
    vencimento: {
      assunto: '💳 Seu pagamento vence hoje - SMagenda',
      mensagem: `
        Olá ${usuario.nome_completo},

        Seu pagamento do plano ${assinatura.plano.toUpperCase()} vence hoje!

        Valor: R$ ${assinatura.valor.toFixed(2)}
        Vencimento: ${formatDate(assinatura.data_vencimento)}

        Para evitar a suspensão do serviço, regularize seu pagamento:
        ${process.env.APP_URL}/pagamento/${assinatura.id}

        Atenciosamente,
        Equipe SMagenda
      `
    },

    atraso_3: {
      assunto: '⚠️ Pagamento em atraso - SMagenda',
      mensagem: `
        Olá ${usuario.nome_completo},

        Seu pagamento está com 3 dias de atraso.

        Valor: R$ ${assinatura.valor.toFixed(2)}
        Vencimento: ${formatDate(assinatura.data_vencimento)}

        ⚠️ ATENÇÃO: Se não regularizar em 11 dias, sua conta será suspensa.

        Regularize agora:
        ${process.env.APP_URL}/pagamento/${assinatura.id}

        Dúvidas? Responda este email.

        Equipe SMagenda
      `
    },

    atraso_7: {
      assunto: '🚨 URGENTE: Pagamento em atraso - SMagenda',
      mensagem: `
        Olá ${usuario.nome_completo},

        Seu pagamento está com 7 dias de atraso.

        ⚠️ Sua conta será SUSPENSA em 7 dias se não regularizar.

        Isso significa que você e seus funcionários não poderão:
        • Acessar o sistema
        • Receber novos agendamentos
        • Enviar lembretes automáticos

        Valor: R$ ${assinatura.valor.toFixed(2)}

        Regularize AGORA:
        ${process.env.APP_URL}/pagamento/${assinatura.id}

        Precisa de ajuda? Entre em contato: suporte@smagenda.com

        Equipe SMagenda
      `
```

```
    },

    suspensao: {
      assunto: '🔒 Conta Suspensa - SMagenda',
      mensagem: `
        Olá ${usuario.nome_completo},

        Sua conta foi SUSPENSA por falta de pagamento.

        Você não consegue mais:
        ❌ Acessar o sistema
        ❌ Receber agendamentos
        ❌ Seus funcionários estão bloqueados

        ⚠️ Em 16 dias seus dados serão EXCLUÍDOS permanentemente.

        Regularize URGENTE:
        ${process.env.APP_URL}/pagamento/${assinatura.id}

        Valor: R$ ${assinatura.valor.toFixed(2)} + multa de 2%

        Atenciosamente,
        Equipe SMagenda
      `
    },

    cancelamento: {
      assunto: '❌ Conta Cancelada - SMagenda',
      mensagem: `
        Olá ${usuario.nome_completo},

        Sua conta foi CANCELADA após 30 dias de inadimplência.

        Seus dados serão excluídos em 24 horas.

        Se deseja reativar, entre em contato:
        suporte@smagenda.com

        Será necessário pagar o valor em atraso.

        Lamentamos o ocorrido.

        Equipe SMagenda
      `
    }
  };

  const { assunto, mensagem } = templates[tipo];

  // Envia email (usando serviço como SendGrid, Resend, etc)
  await sendEmail({
    to: usuario.email,
    subject: assunto,
    text: mensagem
  });

  // Envia WhatsApp (opcional)
  if (tipo === 'suspensao' || tipo === 'atraso_7') {
    await sendWhatsAppNotification(usuario.telefone, mensagem);
  }

  // Registra no banco
  await supabase.from('notificacoes_enviadas').insert({
    usuario_id: usuario.id,
    tipo: 'cobranca',
    subtipo: tipo,
    enviado_em: new Date().toISOString()
  });

  console.log(`📧 Notificação "${tipo}" enviada para ${usuario.email}`);
}
```

~~~
// Função para suspender conta
async function suspendAccount(assinatura) {
  // Atualiza status do usuário
  await supabase
    .from('usuarios')
    .update({
      ativo: false,
      status_pagamento: 'suspenso'
    })
    .eq('id', assinatura.usuario_id);

  // Desativa todos os funcionários
  await supabase
    .from('funcionarios')
    .update({ ativo: false })
    .eq('usuario_master_id', assinatura.usuario_id);

  // Atualiza status da assinatura
  await supabase
    .from('assinaturas')
    .update({ status: 'suspensa' })
    .eq('id', assinatura.id);

  console.log(`🔒 Conta suspensa: ${assinatura.usuario_id}`);
}

// Função para cancelar conta
async function cancelAccount(assinatura) {
  // Marca para exclusão
  await supabase
    .from('usuarios')
    .update({
      ativo: false,
      status_pagamento: 'cancelado',
      data_exclusao_agendada: new Date(Date.now() + 24 * 60 * 60 * 1000) // 24h
    })
    .eq('id', assinatura.usuario_id);

  await supabase
    .from('assinaturas')
    .update({
      status: 'cancelada',
      data_cancelamento: new Date().toISOString()
    })
    .eq('id', assinatura.id);

  console.log(`❌ Conta cancelada: ${assinatura.usuario_id}`);
}

// Atualiza status da assinatura
async function updateAssinaturaStatus(assinaturaId, status) {
  await supabase
    .from('assinaturas')
    .update({ status })
    .eq('id', assinaturaId);
}

// Formata data
function formatDate(dateString) {
  return new Date(dateString).toLocaleDateString('pt-BR');
}

// Agenda o cron job para rodar todo dia às 9h
cron.schedule('0 9 * * *', () => {
  checkPendingPayments();
});

export { checkPendingPayments };
```

#### Dashboard de Notificações (Super Admin)
~~~

~~~
```jsx
// Adicionar na Tela SA-2 do Super Admin

<div className="mt-6">
  <h3 className="font-bold mb-3">⚠️ Ações Necessárias</h3>

  <div className="space-y-2">
    <div className="bg-yellow-50 border border-yellow-200 rounded p-3">
      <div className="flex justify-between items-center">
        <div>
          <p className="font-semibold">5 clientes em trial acabando</p>
          <p className="text-sm text-gray-600">Trial termina em menos de 3 dias</p>
        </div>
        <button className="text-blue-600 hover:underline">
          Ver lista
        </button>
      </div>
    </div>

    <div className="bg-red-50 border border-red-200 rounded p-3">
      <div className="flex justify-between items-center">
        <div>
          <p className="font-semibold">4 clientes inadimplentes</p>
          <p className="text-sm text-gray-600">Atraso de 3-14 dias</p>
        </div>
        <button className="text-blue-600 hover:underline">
          Enviar cobrança
        </button>
      </div>
    </div>

    <div className="bg-gray-50 border border-gray-200 rounded p-3">
      <div className="flex justify-between items-center">
        <div>
          <p className="font-semibold">2 contas suspensas</p>
          <p className="text-sm text-gray-600">Há mais de 14 dias</p>
        </div>
        <button className="text-blue-600 hover:underline">
          Verificar
        </button>
      </div>
    </div>
  </div>
</div>
```

---

### 3️⃣ **Exportação de Relatórios Financeiros**

#### Tipos de Relatórios

```javascript
// services/reportService.js

// 1. Relatório de MRR (Monthly Recurring Revenue)
async function generateMRRReport(mes, ano) {
  const { data: assinaturas } = await supabase
    .from('assinaturas')
    .select(`
      id,
      valor,
      plano,
      status,
      data_inicio,
      usuarios:usuario_id (nome_negocio, email)
    `)
    .eq('status', 'ativa')
    .gte('data_inicio', `${ano}-${mes}-01`)
    .lt('data_inicio', `${ano}-${mes + 1}-01`);

  const mrrTotal = assinaturas.reduce((sum, sub) => sum + parseFloat(sub.valor), 0);
~~~

```
  const porPlano = {
    free: 0,
    basic: 0,
    pro: 0,
    team: 0,
    enterprise: 0
  };

  assinaturas.forEach(sub => {
    porPlano[sub.plano] += parseFloat(sub.valor);
  });

  return {
    mes,
    ano,
    mrr_total: mrrTotal,
    total_assinaturas: assinaturas.length,
    por_plano: porPlano,
    assinaturas: assinaturas.map(sub => ({
      cliente: sub.usuarios.nome_negocio,
      email: sub.usuarios.email,
      plano: sub.plano,
      valor: sub.valor
    }))
  };
}

// 2. Relatório de Churn (cancelamentos)
async function generateChurnReport(mes, ano) {
  // Assinaturas ativas no início do mês
  const { count: ativasInicio } = await supabase
    .from('assinaturas')
    .select('*', { count: 'exact', head: true })
    .lt('data_inicio', `${ano}-${mes}-01`)
    .in('status', ['ativa', 'inadimplente']);

  // Assinaturas canceladas no mês
  const { data: canceladas, count: totalCanceladas } = await supabase
    .from('assinaturas')
    .select(`
      *,
      usuarios:usuario_id (nome_negocio, email, data_cadastro)
    `)
    .eq('status', 'cancelada')
    .gte('data_cancelamento', `${ano}-${mes}-01`)
    .lt('data_cancelamento', `${ano}-${mes + 1}-01`);

  const taxaChurn = ((totalCanceladas / ativasInicio) * 100).toFixed(2);

  // Motivos de cancelamento
  const motivos = canceladas.reduce((acc, sub) => {
    const dias = Math.floor(
      (new Date(sub.data_cancelamento) - new Date(sub.usuarios.data_cadastro)) /
      (1000 * 60 * 60 * 24)
    );

    let motivo = 'Outro';
    if (dias < 7) motivo = 'Cancelou durante trial';
    else if (dias < 30) motivo = 'Cancelou no primeiro mês';
    else motivo = 'Inadimplência';

    acc[motivo] = (acc[motivo] || 0) + 1;
    return acc;
  }, {});

  return {
    mes,
    ano,
    assinaturas_ativas_inicio: ativasInicio,
    total_canceladas: totalCanceladas,
    taxa_churn: `${taxaChurn}%`,
```

```
    motivos_cancelamento: motivos,
    detalhes: canceladas.map(sub => ({
      cliente: sub.usuarios.nome_negocio,
      email: sub.usuarios.email,
      plano: sub.plano,
      valor_perdido: sub.valor,
      data_cadastro: sub.usuarios.data_cadastro,
      data_cancelamento: sub.data_cancelamento,
      tempo_como_cliente: Math.floor(
        (new Date(sub.data_cancelamento) - new Date(sub.usuarios.data_cadastro)) /
        (1000 * 60 * 60 * 24)
      ) + ' dias'
    }))
  };
}

// 3. Relatório de Inadimplência
async function generateInadimplenciaReport() {
  const { data: inadimplentes } = await supabase
    .from('assinaturas')
    .select(`
      *,
      usuarios:usuario_id (nome_negocio, email, telefone)
    `)
    .eq('status', 'inadimplente');

  const hoje = new Date();

  const detalhado = inadimplentes.map(sub => {
    const vencimento = new Date(sub.data_vencimento);
    const diasAtraso = Math.floor((hoje - vencimento) / (1000 * 60 * 60 * 24));

    let gravidade = '';
    if (diasAtraso <= 7) gravidade = '🟡 Leve';
    else if (diasAtraso <= 14) gravidade = '🟠 Moderada';
    else gravidade = '🔴 Grave';

    return {
      cliente: sub.usuarios.nome_negocio,
      email: sub.usuarios.email,
      telefone: sub.usuarios.telefone,
      plano: sub.plano,
      valor: sub.valor,
      vencimento: sub.data_vencimento,
      dias_atraso: diasAtraso,
      gravidade,
      valor_total_devido: (parseFloat(sub.valor) * 1.02).toFixed(2) // +2% multa
    };
  });

  const totalDevido = detalhado.reduce((sum, item) =>
    sum + parseFloat(item.valor_total_devido), 0
  );

  return {
    total_inadimplentes: inadimplentes.length,
    valor_total_devido: totalDevido.toFixed(2),
    por_gravidade: {
      leve: detalhado.filter(d => d.gravidade.includes('Leve')).length,
      moderada: detalhado.filter(d => d.gravidade.includes('Moderada')).length,
      grave: detalhado.filter(d => d.gravidade.includes('Grave')).length
    },
    detalhes: detalhado
  };
}

// 4. Exportar para CSV
function exportToCSV(data, filename) {
  const headers = Object.keys(data[0]).join(',');
  const rows = data.map(row =>
    Object.values(row).map(val => `"${val}"`).join(',')
  );
```

```
  const csv = [headers, ...rows].join('\n');

  return {
    content: csv,
    filename: `${filename}_${new Date().toISOString().split('T')[0]}.csv`,
    mimeType: 'text/csv'
  };
}

// 5. Exportar para Excel (usando xlsx)
import XLSX from 'xlsx';

function exportToExcel(reports, filename) {
  const workbook = XLSX.utils.book_new();

  // Aba 1: MRR
  const mrrSheet = XLSX.utils.json_to_sheet(reports.mrr.assinaturas);
  XLSX.utils.book_append_sheet(workbook, mrrSheet, 'MRR');

  // Aba 2: Churn
  const churnSheet = XLSX.utils.json_to_sheet(reports.churn.detalhes);
  XLSX.utils.book_append_sheet(workbook, churnSheet, 'Churn');

  // Aba 3: Inadimplência
  const inadimSheet = XLSX.utils.json_to_sheet(reports.inadimplencia.detalhes);
  XLSX.utils.book_append_sheet(workbook, inadimSheet, 'Inadimplência');

  // Aba 4: Resumo
  const resumo = [
    { Métrica: 'MRR Total', Valor: `R$ ${reports.mrr.mrr_total}` },
    { Métrica: 'Total Assinaturas', Valor: reports.mrr.total_assinaturas },
    { Métrica: 'Taxa de Churn', Valor: reports.churn.taxa_churn },
    { Métrica: 'Inadimplentes', Valor: reports.inadimplencia.total_inadimplentes },
    { Métrica: 'Valor Devido', Valor: `R$ ${reports.inadimplencia.valor_total_devido}` }
  ];
  const resumoSheet = XLSX.utils.json_to_sheet(resumo);
  XLSX.utils.book_append_sheet(workbook, resumoSheet, 'Resumo');# 📱 SMagenda - Sistema de Agendamento Inteligente -
Documentação Completa

## 🎯 Visão Geral do Produto

**Nome:** SMagenda
**Proposta de Valor:** "Pare de perder dinheiro com clientes que não aparecem. Automatize lembretes e organize sua
agenda em um só lugar."

**Diferencial Principal:** Sistema 100% mobile, link direto (sem necessidade de app), lembretes automáticos via WhatsApp
e setup em menos de 10 minutos.

### Persona 4: **Dono de Barbearia com Equipe** ⭐ NOVO
- **Idade:** 30-50 anos
- **Dor:** Desorganização da equipe, não sabe quem atendeu quem, dificuldade de controlar horários
- **Comportamento:** Quer profissionalizar o negócio, precisa de controle sem microgerenciar
- **Onde encontrar:** Associações de barbearias, grupos no WhatsApp, eventos do setor
- **Ticket médio:** R$ 179-299/mês (maior valor!)

---

#### Tela 9: **Gestão de Funcionários** (Somente Master) ⭐ NOVO
**URL:** `/funcionarios`

**Layout:**
```

```
┌─────────────────────────────────────┐
│ [☰ Menu]  Funcionários    [+ Novo] │
├─────────────────────────────────────┤
│                                     │
│  ┌─────────────────────────────────┐│
│  │ 👤 Carlos Silva          [✏️] ││
│  │ carlos@email.com                ││
│  │ 📱 (11) 98888-8888              ││
│  │                                 ││
```

~~~~
│  │ 🟢 Ativo • Funcionário          ││
│  │ 📅 Atende: Seg-Sex 9h-18h      ││
│  │ ✂️ 23 agendamentos este mês     ││
│  │                                 ││
│  │ Permissões:                     ││
│  │ ✅ Ver agenda                   ││
│  │ ✅ Criar agendamentos          ││
│  │ ❌ Ver financeiro               ││
│  │ ❌ Gerenciar serviços           ││
│  │                                 ││
│  │ [Editar] [Desativar]            ││
│  └─────────────────────────────────┘│
│                                     │
│                                     │
│  ┌─────────────────────────────────┐│
│  │ 👤 Maria Santos          [✏️]  ││
│  │ maria@email.com                 ││
│  │ 📱 (11) 97777-7777              ││
│  │                                 ││
│  │ 🟢 Ativo • Admin                ││
│  │ 📅 Atende: Ter-Sab 10h-19h     ││
│  │ ✂️ 31 agendamentos este mês     ││
│  │                                 ││
│  │ Permissões:                     ││
│  │ ✅ Ver agenda                   ││
│  │ ✅ Criar agendamentos          ││
│  │ ✅ Ver financeiro               ││
│  │ ✅ Gerenciar serviços           ││
│  │                                 ││
│  │ [Editar] [Desativar]            ││
│  └─────────────────────────────────┘│
└─────────────────────────────────────┘
```

**Modal de Adicionar Funcionário:**
```
┌─────────────────────────────────────┐
│ Adicionar Funcionário         [✕]  │
├─────────────────────────────────────┤
│                                     │
│ Dados Básicos:                     │
│ ┌─────────────────────────────┐    │
│ │ Nome completo               │    │
│ └─────────────────────────────┘    │
│                                     │
│ ┌─────────────────────────────┐    │
│ │ Email (para login)          │    │
│ └─────────────────────────────┘    │
│                                     │
│ ┌─────────────────────────────┐    │
│ │ (11) 9____-____             │    │
│ └─────────────────────────────┘    │
│                                     │
│ ┌─────────────────────────────┐    │
│ │ Senha inicial               │    │
│ └─────────────────────────────┘    │
│                                     │
│ Nível de Acesso:                   │
│ ( ) Admin - Acesso quase total     │
│ (•) Funcionário - Acesso limitado  │
│                                     │
│ Horário de Trabalho:               │
│ Das [09:00] às [18:00]             │
│ Dias: [S][T][Q][Q][S][S][D]        │
│                                     │
│ Permissões Detalhadas:             │
│ [✓] Ver agenda                     │
│ [✓] Criar agendamentos             │
│ [✓] Cancelar agendamentos          │
│ [✓] Bloquear horários próprios     │
│ [ ] Ver valores/financeiro         │
│ [ ] Gerenciar serviços             │
│ [ ] Ver clientes de outros         │
~~~~

~~~~
│                                    │
│      [Cancelar] [Criar Acesso]    │
└────────────────────────────────────┘
```

---

#### Tela 10: **Dashboard do Funcionário** (Visão Limitada) ⭐ NOVO
**URL:** `/funcionario/agenda`

**Diferenças do Dashboard Master:**

```
┌────────────────────────────────────┐
│ [☰ Menu]    SMagenda       [🔔][👤] │
│ Olá, Carlos! 👋                     │
├────────────────────────────────────┤
│                                    │
│  [< Hoje - 25 Dez >]    [+ Novo]   │
│                                    │
│  🔍 Mostrando: Meus Agendamentos    │
│                                    │
│  ┌──────────────────────────────┐  │
│  │ 09:00 - João Silva           │  │
│  │ 📱 (11) 99999-9999            │  │
│  │ ✂️ Corte Masculino             │  │
│  │ [✓ Confirmar] [✗ Cancelar]   │  │
│  ├──────────────────────────────┤  │
│  │ 10:00 - LIVRE                │  │
│  ├──────────────────────────────┤  │
│  │ 11:00 - Maria Santos         │  │
│  │ 📱 (11) 98888-8888            │  │
│  │ 💇 Escova                      │  │
│  │ ✅ Confirmado                 │  │
│  └──────────────────────────────┘  │
│                                    │
│  Resumo do Seu Dia:                │
│  ⏰ 5 agendamentos                  │
│  (valores ocultos)                 │
└────────────────────────────────────┘
```

**Menu do Funcionário (Limitado):**
- 📅 Minha Agenda
- 👥 Meus Clientes (só os que ele atendeu)
- 🔒 Meus Horários Bloqueados
- ⚙️ Meu Perfil
- ❓ Ajuda
- 🚪 Sair

**O que o Funcionário NÃO vê:**
- ❌ Valores/receitas (a menos que tenha permissão)
- ❌ Agendamentos de outros funcionários
- ❌ Configurações do negócio
- ❌ Gestão de serviços (preços)
- ❌ Mensagens automáticas
- ❌ Evolution API
- ❌ Planos/pagamentos

---

#### Tela 11: **Agenda Completa do Dono** (Visão Master) ⭐ NOVO
**URL:** `/dashboard` (com filtros)

```
┌────────────────────────────────────┐
│ [☰ Menu]    AgendaFácil    [🔔][👤] │
├────────────────────────────────────┤
│                                    │
│  [< Hoje - 25 Dez >]    [+ Novo]   │
│                                    │
│  Filtrar por:                      │
~~~~

~~~
│  [Todos] [Carlos] [Maria] [João]  │
│                                    │
│  ┌──────────────────────────────┐ │
│  │ 09:00 👤 Carlos              │ │
│  │ Cliente: João Silva          │ │
│  │ ✂️ Corte - R$ 50              │ │
│  ├──────────────────────────────┤ │
│  │ 09:00 👤 Maria               │ │
│  │ Cliente: Ana Costa           │ │
│  │ 💇‍♀️ Escova - R$ 40             │ │
│  ├──────────────────────────────┤ │
│  │ 10:00 👤 Carlos - LIVRE      │ │
│  ├──────────────────────────────┤ │
│  │ 10:00 👤 Maria               │ │
│  │ Cliente: Beatriz             │ │
│  │ 💅 Manicure - R$ 35           │ │
│  └──────────────────────────────┘ │
│                                    │
│  Resumo do Dia:                   │
│  💰 R$ 650,00 • 15 agendamentos    │
│  👤 Carlos: R$ 250 (5 clientes)   │
│  👤 Maria: R$ 400 (10 clientes)   │
└────────────────────────────────────┘
```

---

### 👤 LADO DO CLIENTE (Página de Agendamento com Funcionários)

#### Tela 12: **Escolher Funcionário** ⭐ NOVO
**URL:** `/agendar/{slug}`

**Após escolher o serviço, antes da data:**

```
┌────────────────────────────────────┐
│  [← Voltar]                       │
│                                    │
│  Com qual profissional?           │
│                                    │
│  ┌─────────────────────────────────┐│
│  │ [Foto] Carlos Silva            ││
│  │ ⭐⭐⭐⭐⭐ (45 avaliações)       ││
│  │ Especialidade: Cortes modernos ││
│  │ [Selecionar]                   ││
│  └─────────────────────────────────┘│
│                                    │
│  ┌─────────────────────────────────┐│
│  │ [Foto] Maria Santos            ││
│  │ ⭐⭐⭐⭐⭐ (62 avaliações)       ││
│  │ Especialidade: Coloração       ││
│  │ [Selecionar]                   ││
│  └─────────────────────────────────┘│
│                                    │
│  ┌─────────────────────────────────┐│
│  │ 🤷‍♀️ Tanto faz, primeiro disponível││
│  │ [Selecionar]                   ││
│  └─────────────────────────────────┘│
│                                    │
└────────────────────────────────────┘
```

**Lógica:**
- Cliente escolhe o profissional OU "tanto faz"
- Sistema mostra horários apenas daquele profissional
- Se escolher "tanto faz", mostra TODOS os horários livres de qualquer um

---

## 🏗️ Arquitetura Técnica

### Stack Escolhida (100% Gratuita)
~~~

~~~
| Componente | Tecnologia | Custo | Limite Gratuito |
|------------|------------|-------|-----------------|
| **Frontend** | React + Vite + TailwindCSS | R$ 0 | Ilimitado |
| **Backend/BD** | Supabase | R$ 0 | 500MB BD + 2GB storage + 50k usuários |
| **Hospedagem** | Vercel | R$ 0 | 100GB bandwidth/mês |
| **WhatsApp** | Evolution API v2 | R$ 0 | Self-hosted (Railway free tier) |
| **Domínio** | Hostinger/Registro.br | R$ 40/ano | - |

### Tipo de Aplicação
**PWA (Progressive Web App)** - Funciona como site + app sem precisar de lojas

---

## 🗄️ Estrutura do Banco de Dados

### Tabelas Principais

#### 1. **usuarios** (Profissionais - Conta Master)
```sql
id: UUID (PK)
nome_completo: TEXT
nome_negocio: TEXT
slug: TEXT (UNIQUE) -- ex: "barbearia-do-joao"
telefone: TEXT
email: TEXT (UNIQUE)
senha_hash: TEXT
foto_perfil: TEXT (URL)
endereco: TEXT (opcional)
horario_inicio: TIME -- ex: "08:00"
horario_fim: TIME -- ex: "18:00"
dias_trabalho: JSONB -- ex: [1,2,3,4,5,6] (seg a sab)
intervalo_inicio: TIME (opcional) -- ex: "12:00"
intervalo_fim: TIME (opcional) -- ex: "13:00"
whatsapp_api_url: TEXT (URL da Evolution API)
whatsapp_api_key=[REDACTED]
plano: TEXT (free, basic, pro, team, enterprise)
tipo_conta: TEXT -- 'master', 'individual'
limite_funcionarios: INTEGER -- baseado no plano
status_pagamento: TEXT -- 'ativo', 'inadimplente', 'cancelado', 'trial'
data_cadastro: TIMESTAMP
data_vencimento: DATE -- próximo pagamento
ativo: BOOLEAN
```

#### 0. **super_admin** (VOCÊ - Administrador do Sistema) ⭐⭐ NOVO
```sql
id: UUID (PK)
nome: TEXT
email: TEXT (UNIQUE)
senha_hash: TEXT
nivel: TEXT -- 'super_admin', 'suporte'
ultimo_acesso: TIMESTAMP
criado_em: TIMESTAMP
```

#### 1.1. **funcionarios** (Sub-contas/Colaboradores) ⭐ NOVO
```sql
id: UUID (PK)
usuario_master_id: UUID (FK -> usuarios) -- dono do negócio
nome_completo: TEXT
email: TEXT (UNIQUE)
senha_hash: TEXT
telefone: TEXT
foto_perfil: TEXT (URL)
permissao: TEXT -- 'admin', 'funcionario'
horario_inicio: TIME -- horário de trabalho deste funcionário
horario_fim: TIME
dias_trabalho: JSONB
intervalo_inicio: TIME (opcional)
intervalo_fim: TIME (opcional)
pode_ver_financeiro: BOOLEAN -- se pode ver valores/receitas
~~~

~~~~
pode_gerenciar_servicos: BOOLEAN
pode_bloquear_horarios: BOOLEAN
pode_cancelar_agendamentos: BOOLEAN
ativo: BOOLEAN
criado_em: TIMESTAMP
desativado_em: TIMESTAMP (nullable)
```

#### 2. **servicos**
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios)
nome: TEXT -- ex: "Corte Masculino"
descricao: TEXT (opcional)
duracao_minutos: INTEGER -- ex: 45
preco: DECIMAL(10,2)
cor: TEXT -- para visualização na agenda (ex: "#FF5733")
ativo: BOOLEAN
ordem: INTEGER -- para ordenar na listagem
```

#### 3. **agendamentos**
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios) -- dono do negócio
funcionario_id: UUID (FK -> funcionarios) -- quem vai atender ⭐ NOVO
servico_id: UUID (FK -> servicos)
cliente_nome: TEXT
cliente_telefone: TEXT
data: DATE
hora_inicio: TIME
hora_fim: TIME (calculado automaticamente)
status: TEXT -- 'confirmado', 'cancelado', 'concluido', 'nao_compareceu'
lembrete_enviado: BOOLEAN
data_lembrete: TIMESTAMP (quando foi enviado)
observacoes: TEXT (opcional)
criado_em: TIMESTAMP
criado_por: UUID -- pode ser usuario_id ou funcionario_id ⭐ NOVO
cancelado_em: TIMESTAMP (nullable)
cancelado_por: UUID (nullable) ⭐ NOVO
```

#### 4. **clientes** (Cache de clientes recorrentes)
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios)
nome: TEXT
telefone: TEXT
total_agendamentos: INTEGER
ultimo_agendamento: TIMESTAMP
criado_em: TIMESTAMP

UNIQUE(usuario_id, telefone)
```

#### 5. **bloqueios** (Horários bloqueados pelo profissional)
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios)
data: DATE
hora_inicio: TIME
hora_fim: TIME
motivo: TEXT (opcional) -- "Almoço", "Compromisso pessoal"
criado_em: TIMESTAMP
```

#### 6. **configuracoes_mensagens**
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios)
mensagem_confirmacao: TEXT
mensagem_lembrete: TEXT
~~~~

~~~
horas_antecedencia: INTEGER -- default: 24
enviar_confirmacao: BOOLEAN
enviar_lembrete: BOOLEAN
```

#### 7. **logs_admin** (Auditoria das suas ações) ⭐⭐ NOVO
```sql
id: UUID (PK)
super_admin_id: UUID (FK -> super_admin)
usuario_afetado_id: UUID (FK -> usuarios) -- qual cliente foi afetado
acao: TEXT -- 'login_como_usuario', 'excluir_funcionario', 'alterar_plano', 'suspender_conta', 'reativar_conta'
detalhes: JSONB -- dados da ação
ip_address: TEXT
criado_em: TIMESTAMP
```

#### 8. **assinaturas** (Controle de pagamentos) ⭐⭐ NOVO
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios)
plano: TEXT
valor: DECIMAL(10,2)
status: TEXT -- 'ativa', 'cancelada', 'inadimplente', 'trial'
metodo_pagamento: TEXT -- 'cartao', 'boleto', 'pix'
data_inicio: DATE
data_vencimento: DATE
data_cancelamento: DATE (nullable)
gateway_subscription_id: TEXT -- ID do Stripe/Mercado Pago
criado_em: TIMESTAMP
```

---

## 🎨 Telas Detalhadas

---

## 🔧 PAINEL SUPER ADMIN (VOCÊ - Dono do Sistema) ⭐⭐ NOVO

### Tela SA-1: **Login Super Admin**
**URL:** `/admin/login`

```
┌─────────────────────────────────────┐
│                                     │
│         🛠️ SUPER ADMIN               │
│         SMagenda                    │
│                                     │
│  ┌─────────────────────────────┐   │
│  │ Email                       │   │
│  └─────────────────────────────┘   │
│                                     │
│  ┌─────────────────────────────┐   │
│  │ Senha                       │   │
│  └─────────────────────────────┘   │
│                                     │
│  [Entrar no Painel Admin]         │
│                                     │
│  ⚠️ Acesso restrito                 │
└─────────────────────────────────────┘
```

---

### Tela SA-2: **Dashboard Super Admin**
**URL:** `/admin/dashboard`

```
┌─────────────────────────────────────┐
│ [☰] SMagenda - Admin      [👤][🚪]│
├─────────────────────────────────────┤
│                                     │
~~~

~~~~
│  📊 Visão Geral                    │
│                                     │
│  ┌──────┐  ┌──────┐  ┌──────┐      │
│  │  47  │  │ R$   │  │ 312  │      │
│  │Clien-│  │7.320 │  │Func. │      │
│  │tes   │  │ MRR  │  │Total │      │
│  └──────┘  └──────┘  └──────┘      │
│                                     │
│  ┌─────────────────┐  ┌─────────┐  │
│  │ 🟢 38 ativos     │  │ Últimas ││
│  │ 🟡 5 trial       │  │ Ações:  ││
│  │ 🔴 4 inadimp.    │  │         ││
│  └─────────────────┘  │ • Login ││
│                       │   João  ││
│  Crescimento:         │         ││
│  📈 +12% este mês      │ • Plano ││
│                       │   Maria ││
│  Churn: 8%            │   ->PRO ││
│  ✅ Meta: <10%         │         ││
│                       └─────────┘│
└─────────────────────────────────────┘
```

**Menu Lateral:**
- 🏠 Dashboard
- 👥 Todos os Clientes
- 💰 Assinaturas
- 📊 Relatórios
- 🔍 Logs de Auditoria
- ⚙️ Configurações do Sistema
- 🚪 Sair

---

### Tela SA-3: **Lista de Clientes (Todos os Masters)**
**URL:** `/admin/clientes`

```
┌─────────────────────────────────────┐
│ [☰] Todos os Clientes               │
├─────────────────────────────────────┤
│                                     │
│ Buscar: [____________] [🔍]         │
│                                     │
│ Filtros:                           │
│ [Todos] [Ativos] [Trial] [Inadimp]│
│ [Free] [Basic] [Pro] [Team] [Ent] │
│                                     │
│ ┌─────────────────────────────────┐│
│ │ 🟢 Barbearia do João             ││
│ │ João Silva • joao@email.com     ││
│ │ 📱 (11) 99999-9999              ││
│ │                                 ││
│ │ Plano: PRO (R$ 99,90/mês)      ││
│ │ Status: ✅ Ativo                 ││
│ │ Venc: 05/01/2026                ││
│ │ Funcionários: 2/2               ││
│ │ Agendamentos: 234 (total)       ││
│ │ Cadastro: 15/08/2025            ││
│ │                                 ││
│ │ [🔐 Logar Como] [✏️ Editar]     ││
│ │ [📊 Detalhes] [⚠️ Suspender]    ││
│ └─────────────────────────────────┘│
│                                     │
│ ┌─────────────────────────────────┐│
│ │ 🟡 Salão da Maria                ││
│ │ Maria Santos • maria@email.com  ││
│ │ 📱 (11) 98888-8888              ││
│ │                                 ││
│ │ Plano: TRIAL → PRO              ││
│ │ Status: ⏳ Trial (5 dias rest.) ││
│ │ Venc: 30/12/2025                ││
~~~~

~~~
│ │ Funcionários: 1/2          ││
│ │ Agendamentos: 23 (trial)   ││
│ │ Cadastro: 20/12/2025       ││
│ │                            ││
│ │ [🔐 Logar Como] [✏️ Editar]  ││
│ │ [📊 Detalhes] [💰 Cobrar]    ││
│ └────────────────────────────┘│
│                               │
│ ┌────────────────────────────┐│
│ │ 🔴 Studio de Tatuagem       ││
│ │ Carlos Mendes • carlos@email.com││
│ │ 📱 (11) 97777-7777          ││
│ │                            ││
│ │ Plano: TEAM (R$ 179,90/mês) ││
│ │ Status: ⚠️ Inadimplente (12d) ││
│ │ Venc: 13/12/2025           ││
│ │ Funcionários: 4/5 (bloqueados) ││
│ │ Agendamentos: 567 (total)  ││
│ │ Cadastro: 03/05/2025       ││
│ │                            ││
│ │ [🔐 Logar Como] [✏️ Editar]  ││
│ │ [📧 Enviar Cobrança] [❌ Canc.]││
│ └────────────────────────────┘│
│                               │
│ Mostrando 3 de 47 clientes    │
│ [← 1 2 3 4 5 →]               │
└───────────────────────────────┘
```

---

### Tela SA-4: **Detalhes do Cliente**
**URL:** `/admin/clientes/{id}`

```
┌───────────────────────────────┐
│ [← Voltar] Barbearia do João  │
├───────────────────────────────┤
│                               │
│ 👤 Informações do Dono         │
│ ┌────────────────────────────┐│
│ │ Nome: João Silva           ││
│ │ Email: joao@email.com      ││
│ │ Telefone: (11) 99999-9999  ││
│ │ Slug: barbearia-do-joao    ││
│ │ Link: smagenda.com/...     ││
│ │ Cadastro: 15/08/2025       ││
│ │ Último acesso: Há 2 horas  ││
│ └────────────────────────────┘│
│                               │
│ 💳 Assinatura                  │
│ ┌────────────────────────────┐│
│ │ Plano: PRO                 ││
│ │ Valor: R$ 99,90/mês        ││
│ │ Status: 🟢 Ativo            ││
│ │ Próximo venc: 05/01/2026   ││
│ │ Método: Cartão (••4532)    ││
│ │ [Alterar Plano] [Ver Histórico]││
│ └────────────────────────────┘│
│                               │
│ 👥 Funcionários (2/2 permitidos) │
│ ┌────────────────────────────┐│
│ │ • Carlos Silva (Funcionário) ││
│ │   carlos@email.com         ││
│ │   [Ver] [❌ Excluir]         ││
│ │                            ││
│ │ • Ana Costa (Admin)        ││
│ │   ana@email.com            ││
│ │   [Ver] [❌ Excluir]         ││
│ └────────────────────────────┘│
│                               │
│ 📊 Estatísticas                │
~~~

~~~
│ ┌─────────────────────────────┐│
│ │ • 234 agendamentos (total)  ││
│ │ • 187 agendamentos (últimos 30d)││
│ │ • 12 clientes cadastrados   ││
│ │ • 5 serviços ativos         ││
│ │ • Taxa no-show: 8%          ││
│ │ • WhatsApp conectado: ✅     ││
│ └─────────────────────────────┘│
│                                │
│ 🛠 Ações Administrativas         │
│ ┌─────────────────────────────┐│
│ │ [🔐 Logar como este usuário] ││
│ │ [✏️ Editar dados]             ││
│ │ [📝 Adicionar observação]     ││
│ │ [⬆️ Upgrade de plano]         ││
│ │ [⬇️ Downgrade de plano]       ││
│ │ [⏸️ Suspender temporariamente] ││
│ │ [❌ Cancelar assinatura]      ││
│ │ [🔄 Resetar senha]            ││
│ └─────────────────────────────┘│
└────────────────────────────────┘
```

---

### Tela SA-5: **Logar Como Cliente** (Impersonation)
**URL:** Ação que redireciona para `/dashboard`

**Fluxo:**
```
1. Você clica em "Logar Como" no painel admin

2. Sistema registra no log:
   - Quem: seu email admin
   - Quando: timestamp
   - Cliente: qual conta acessou
   - IP: seu endereço IP

3. Você é redirecionado para o dashboard DO CLIENTE
   - Vê exatamente o que ele vê
   - Pode fazer tudo que ele pode

4. Banner de alerta aparece no topo:
   ┌──────────────────────────────────────┐
   │ ⚠️ MODO ADMIN: Você está logado      │
   │ como "João Silva"                    │
   │ [🚪 Voltar ao Admin]                  │
   └──────────────────────────────────────┘

5. Pode:
   - Ver agenda dele
   - Acessar funcionários dele
   - Excluir funcionários se necessário
   - Testar funcionalidades
   - Resolver problemas

6. Ao clicar "Voltar ao Admin":
   - Sistema registra saída no log
   - Você volta para seu painel admin
```

---

### Tela SA-6: **Gerenciar Funcionários do Cliente**
**URL:** `/admin/clientes/{id}/funcionarios`

```
┌────────────────────────────────────────┐
│ [← Voltar] Funcionários                │
│ Barbearia do João                      │
├────────────────────────────────────────┤
│                                        │
~~~

~~~
│ Plano atual: PRO (2/2 funcionários) │
│                                     │
│ ┌─────────────────────────────────┐ │
│ │ 👤 Carlos Silva                  │ │
│ │ carlos@email.com                │ │
│ │ 📱 (11) 98888-8888               │ │
│ │                                 │ │
│ │ Tipo: Funcionário               │ │
│ │ Status: 🟢 Ativo                 │ │
│ │ Cadastrado: 20/08/2025          │ │
│ │ Último acesso: Há 3 horas       │ │
│ │ Agendamentos: 89 (total)        │ │
│ │                                 │ │
│ │ Permissões:                     │ │
│ │ ✅ Ver agenda                    │ │
│ │ ✅ Criar agendamentos            │ │
│ │ ❌ Ver financeiro                │ │
│ │                                 │ │
│ │ [✏️ Editar] [❌ Excluir]          │ │
│ │ [⏸️ Desativar] [🔄 Resetar Senha]││
│ └─────────────────────────────────┘ │
│                                     │
│ ┌─────────────────────────────────┐ │
│ │ 👤 Ana Costa                     │ │
│ │ ana@email.com                   │ │
│ │ 📱 (11) 97777-7777               │ │
│ │                                 │ │
│ │ Tipo: Admin                     │ │
│ │ Status: 🟢 Ativo                 │ │
│ │ Cadastrado: 15/09/2025          │ │
│ │ Último acesso: Há 1 dia         │ │
│ │ Agendamentos: 45 (total)        │ │
│ │                                 │ │
│ │ Permissões:                     │ │
│ │ ✅ Ver agenda                    │ │
│ │ ✅ Criar agendamentos            │ │
│ │ ✅ Ver financeiro                │ │
│ │ ✅ Gerenciar serviços            │ │
│ │                                 │ │
│ │ [✏️ Editar] [❌ Excluir]          │ │
│ │ [⏸️ Desativar] [🔄 Resetar Senha]││
│ └─────────────────────────────────┘ │
│                                     │
│ ⚠️ Limite atingido (2/2)             │
│ Para adicionar mais funcionários,   │
│ upgrade para TEAM necessário.       │
│                                     │
│ [⬆️ Fazer Upgrade para TEAM]         │
└─────────────────────────────────────┘
```

**Ao clicar em "Excluir Funcionário":**
```
┌─────────────────────────────────────┐
│ ⚠️ Confirmar Exclusão                │
├─────────────────────────────────────┤
│                                     │
│ Você está prestes a EXCLUIR:        │
│                                     │
│ 👤 Carlos Silva                      │
│ carlos@email.com                    │
│                                     │
│ ⚠️ ATENÇÃO:                          │
│ • 89 agendamentos vinculados a ele  │
│ • Agendamentos futuros (12) serão   │
│   mantidos mas sem responsável      │
│ • Histórico será preservado         │
│ • Ação irreversível                 │
│                                     │
│ Motivo (opcional):                  │
│ ┌───────────────────────────────┐   │
│ │ Ex: Saiu da empresa           │   │
~~~

~~~
│ └──────────────────────────────┘    │
│                                      │
│ [Cancelar] [❌ Confirmar Exclusão] │
└──────────────────────────────────────┘
```

---

### Tela SA-7: **Alterar Plano do Cliente**
**URL:** Modal em `/admin/clientes/{id}`

```
┌──────────────────────────────────────┐
│ Alterar Plano - Barbearia do João  │
├──────────────────────────────────────┤
│                                      │
│ Plano Atual: PRO (R$ 99,90/mês)    │
│                                      │
│ Escolha o novo plano:               │
│                                      │
│ ( ) FREE                            │
│     R$ 0 • 30 agends/mês • 0 func  │
│                                      │
│ ( ) BASIC                           │
│     R$ 59,90 • Ilimitado • 0 func  │
│                                      │
│ (•) PRO (atual)                     │
│     R$ 99,90 • Ilimitado • 2 func  │
│                                      │
│ ( ) TEAM                            │
│     R$ 179,90 • Ilimitado • 5 func │
│                                      │
│ ( ) ENTERPRISE                      │
│     R$ 299,90 • Ilimitado • ∞ func │
│                                      │
│ ⚠️ Atenção ao mudar:                  │
│ • Downgrade: funcionalidades podem │
│   ser bloqueadas imediatamente     │
│ • Upgrade: cobra diferença propor- │
│   cional no próximo venc.          │
│                                      │
│ Motivo da alteração:                │
│ ┌──────────────────────────────┐    │
│ │ Cliente solicitou upgrade    │    │
│ └──────────────────────────────┘    │
│                                      │
│ [Cancelar] [Confirmar Alteração]  │
└──────────────────────────────────────┘
```

---

### Tela SA-8: **Logs de Auditoria**
**URL:** `/admin/logs`

```
┌──────────────────────────────────────┐
│ [☰] Logs de Auditoria               │
├──────────────────────────────────────┤
│                                      │
│ Filtros:                            │
│ [Todas Ações] [Últimos 7 dias]     │
│                                      │
│ Buscar: [____________] [🔍]         │
│                                      │
│ ┌──────────────────────────────────┐│
│ │ 🔐 LOGIN COMO USUÁRIO             ││
│ │ 25/12/2025 14:32                ││
│ │                                  ││
│ │ Admin: voce@admin.com           ││
│ │ Cliente: João Silva             ││
│ │ (Barbearia do João)             ││
~~~

~~~
│ │ IP: 192.168.1.100                ││
│ │ Duração: 8 minutos               ││
│ │                                  ││
│ │ [Ver Detalhes]                   ││
│ └──────────────────────────────────┘│
│                                      │
│ ┌──────────────────────────────────┐│
│ │ ❌ EXCLUIR FUNCIONÁRIO            ││
│ │ 25/12/2025 11:15                 ││
│ │                                  ││
│ │ Admin: voce@admin.com            ││
│ │ Cliente: Maria Santos            ││
│ │ Funcionário: Pedro Costa         ││
│ │ Motivo: "Saiu da empresa"        ││
│ │ IP: 192.168.1.100                ││
│ │                                  ││
│ │ [Ver Detalhes]                   ││
│ └──────────────────────────────────┘│
│                                      │
│ ┌──────────────────────────────────┐│
│ │ ⬆️ ALTERAR PLANO                  ││
│ │ 24/12/2025 16:45                 ││
│ │                                  ││
│ │ Admin: voce@admin.com            ││
│ │ Cliente: Carlos Mendes           ││
│ │ De: BASIC → Para: PRO            ││
│ │ Motivo: "Cliente solicitou"      ││
│ │ IP: 192.168.1.100                ││
│ │                                  ││
│ │ [Ver Detalhes]                   ││
│ └──────────────────────────────────┘│
│                                      │
│ Mostrando 3 de 127 registros        │
│ [← 1 2 3 4 5 →]                    │
└──────────────────────────────────────┘
```

---

### Tela SA-9: **Configurações do Sistema**
**URL:** `/admin/configuracoes`

```
┌──────────────────────────────────────┐
│ [☰] Configurações do Sistema         │
├──────────────────────────────────────┤
│                                      │
│ 🎨 Planos e Limites                  │
│ ┌──────────────────────────────────┐│
│ │ FREE:                            ││
│ │ Agendamentos/mês: [30]           ││
│ │ Funcionários: [0]                ││
│ │                                  ││
│ │ BASIC: R$ [59.90]                ││
│ │ Agendamentos/mês: [Ilimitado]    ││
│ │ Funcionários: [0]                ││
│ │                                  ││
│ │ PRO: R$ [99.90]                  ││
│ │ Agendamentos/mês: [Ilimitado]    ││
│ │ Funcionários: [2]                ││
│ │                                  ││
│ │ TEAM: R$ [179.90]                ││
│ │ Funcionários: [5]                ││
│ │                                  ││
│ │ ENTERPRISE: R$ [299.90]          ││
│ │ Funcionários: [Ilimitado]        ││
│ │                                  ││
│ │ [Salvar Alterações]              ││
│ └──────────────────────────────────┘│
│                                      │
│ 💳 Gateways de Pagamento              │
│ ┌──────────────────────────────────┐│
~~~

~~~
│ │ [ ] Stripe                    ││
│ │     API Key: [___________]    ││
│ │                               ││
│ │ [✓] Mercado Pago              ││
│ │     Access Token: [•••••••••]  ││
│ │     Status: 🟢 Conectado       ││
│ │                               ││
│ │ [Salvar]                      ││
│ └───────────────────────────────┘│
│                                  │
│ 📧 Email/Notificações             │
│ ┌───────────────────────────────┐│
│ │ Email de suporte:             ││
│ │ [suporte@smagenda.com]        ││
│ │                               ││
│ │ [✓] Notificar novos cadastros ││
│ │ [✓] Alertas de inadimplência  ││
│ │ [✓] Resumo diário (9h)        ││
│ │                               ││
│ │ [Salvar]                      ││
│ └───────────────────────────────┘│
└──────────────────────────────────┘
```

---

## 📱 LADO DO PROFISSIONAL (Dashboard)

#### Tela 1: **Login/Cadastro**
**URL:** `/login` e `/cadastro`

**Campos de Cadastro:**
- Nome completo
- Nome do negócio
- Telefone (com WhatsApp)
- Email
- Senha
- Slug personalizado (ex: "joao-barbeiro") - auto-gerado mas editável

**Layout:**
- Logo centralizado
- Formulário limpo
- Botão "Criar conta grátis"
- Link "Já tenho conta"

---

#### Tela 2: **Onboarding (Pós-cadastro)**
**URL:** `/onboarding`

**Etapa 1 - Horário de Funcionamento:**
- Seletor de horário início (ex: 08:00)
- Seletor de horário fim (ex: 18:00)
- Checkbox dos dias da semana
- Intervalo (opcional): início e fim

**Etapa 2 - Primeiro Serviço:**
- Nome do serviço
- Duração (em minutos)
- Preço
- Botão "Adicionar mais serviços depois"

**Etapa 3 - Configurar WhatsApp:**
- Opções:
  - [ ] "Enviar manualmente por enquanto" (gera link do WhatsApp)
  - [ ] "Configurar Evolution API agora" (mostra tutorial)
- Link do tutorial: como instalar Evolution API no Railway (gratuito)

**Etapa 4 - Pronto!:**
- Mostra o link de agendamento: `agendafacil.com/joao-barbeiro`
- Botão "Copiar link"
- Botão "Compartilhar no WhatsApp"
~~~

~~~
- Botão "Ir para minha agenda"

---

#### Tela 3: **Dashboard Principal (Agenda)**
**URL:** `/dashboard`

**Layout:**
```
┌─────────────────────────────────────┐
│ [☰ Menu]    AgendaFácil    [🔔][👤] │
├─────────────────────────────────────┤
│                                     │
│  [< Hoje - 25 Dez >]    [+ Novo]   │
│                                     │
│  ┌───────────────────────────────┐ │
│  │ 08:00 - LIVRE                 │ │
│  ├───────────────────────────────┤ │
│  │ 09:00 - João Silva            │ │
│  │ 📱 (11) 99999-9999            │ │
│  │ ✂️ Corte Masculino - R$ 50     │ │
│  │ [✓ Confirmar] [✗ Cancelar]    │ │
│  ├───────────────────────────────┤ │
│  │ 10:00 - LIVRE                 │ │
│  ├───────────────────────────────┤ │
│  │ 11:00 - Maria Santos          │ │
│  │ 📱 (11) 98888-8888            │ │
│  │ 💇‍♀️ Escova - R$ 40             │ │
│  │ ✅ Confirmado                  │ │
│  ├───────────────────────────────┤ │
│  │ 12:00 - 🔒 BLOQUEADO (Almoço)│ │
│  └───────────────────────────────┘ │
│                                     │
│  Resumo do Dia:                    │
│  💰 R$ 250,00 • 5 agendamentos     │
└─────────────────────────────────────┘
```

**Funcionalidades:**
- Navegação entre dias (setas < >)
- Visualização semanal (toggle)
- Cards de agendamento clicáveis
- Status visual com cores
- Botão flutuante "+ Novo Agendamento"
- Filtro rápido: "Todos", "Confirmados", "Pendentes"

---

#### Tela 4: **Menu Lateral**
**URL:** Slide-in menu

**Itens:**
- 📅 Agenda (tela principal)
- ✂️ Meus Serviços
- 👥 Clientes
- 👨‍💼 Funcionários (só para Master/Admin) ⭐ NOVO
- 🔗 Meu Link de Agendamento
- ⚙️ Configurações
- 💬 Mensagens Automáticas
- 📊 Relatórios (futuro)
- 🎨 Personalizar Página
- ❓ Ajuda
- 🚪 Sair

---

#### Tela 5: **Meus Serviços**
**URL:** `/servicos`

**Layout:**
- Lista de cards de serviços
- Cada card mostra:
~~~

~~~
  - Nome do serviço
  - Duração e preço
  - Toggle ativo/inativo
  - Ícone de editar e deletar
- Botão "+ Adicionar Serviço"
- Drag-and-drop para reordenar (mobile friendly)

**Modal de Adicionar/Editar:**
- Nome
- Descrição (opcional)
- Duração (em minutos)
- Preço
- Cor (seletor visual)
- Toggle "Ativo"

---

#### Tela 6: **Configurações de WhatsApp**
**URL:** `/configuracoes/whatsapp`

**Seção 1 - Status da Conexão:**
```
┌─────────────────────────────────────┐
│ Status: 🟢 Conectado                 │
│ Instância: minha-barbearia          │
│ Número: +55 11 99999-9999           │
│ [Desconectar] [Testar Envio]        │
└─────────────────────────────────────┘
```

**Seção 2 - Configurar Evolution API:**
- Campo: URL da API (ex: `https://sua-instancia.railway.app`)
- Campo: API Key
- Botão "Conectar"
- Link: "Não tem Evolution API? Veja como criar grátis"

**Seção 3 - Preferências de Envio:**
- Toggle: "Enviar confirmação ao agendar"
- Toggle: "Enviar lembrete automático"
- Slider: "Enviar lembrete X horas antes" (4h a 72h)

---

#### Tela 7: **Mensagens Automáticas**
**URL:** `/configuracoes/mensagens`

**Mensagem de Confirmação:**
```
┌─────────────────────────────────────┐
│ Olá {nome}! 👋                       │
│                                     │
│ Seu agendamento foi confirmado:     │
│ 📅 {data} às {hora}                  │
│ ✂️ {servico}                         │
│ 💰 {preco}                           │
│                                     │
│ Local: {endereco}                   │
│                                     │
│ Nos vemos em breve!                 │
│ {nome_negocio}                      │
│                                     │
│ [Editar Mensagem]                   │
└─────────────────────────────────────┘
```

**Mensagem de Lembrete:**
```
┌─────────────────────────────────────┐
│ Oi {nome}! 🔔                        │
│                                     │
│ Lembrete: você tem agendamento      │
│ AMANHÃ às {hora}                    │
~~~

~~~
│                                      │
│ Se não puder comparecer, me avise! │
│ {telefone_profissional}             │
│                                      │
│ [Editar Mensagem]                   │
└──────────────────────────────────────┘
```

**Variáveis disponíveis:**
- {nome} - nome do cliente
- {data} - data do agendamento
- {hora} - horário
- {servico} - nome do serviço
- {preco} - valor
- {endereco} - endereço do profissional
- {nome_negocio} - nome do negócio

---

### 👤 LADO DO CLIENTE (Página de Agendamento)

#### Tela 8: **Página Pública de Agendamento**
**URL:** `/agendar/{slug}` ex: `/agendar/joao-barbeiro`

**Layout:**
```
┌──────────────────────────────────────┐
│  [Foto]                             │
│  Barbearia do João                  │
│  ⭐⭐⭐⭐⭐ (23 avaliações)          │
│  📍 Rua das Flores, 123            │
│                                      │
├──────────────────────────────────────┤
│ Escolha o serviço:                  │
│                                      │
│ [✂️ Corte Masculino                 │
│  ⏱️ 45 min • 💰 R$ 50,00 ]          │
│                                      │
│ [💁 Barba                            │
│  ⏱️ 30 min • 💰 R$ 35,00 ]          │
│                                      │
│ [🎨 Corte + Barba (COMBO)           │
│  ⏱️ 1h 15min • 💰 R$ 75,00 ]        │
│                                      │
└──────────────────────────────────────┘
```

**Após selecionar serviço → Tela de Data:**
```
┌──────────────────────────────────────┐
│  [← Voltar]                         │
│                                      │
│  Escolha a data:                    │
│                                      │
│  [ Dezembro 2025 ]                  │
│                                      │
│  D  S  T  Q  Q  S  S               │
│     1  2  3  4  5  6               │
│  7  8  9 10 11 12 13               │
│ 14 15 16 17 18 19 20               │
│ 21 22 23 24[25]26 27               │
│ 28 29 30 31                         │
│                                      │
│  Dias com ❌ estão indisponíveis    │
└──────────────────────────────────────┘
```

**Após selecionar data → Tela de Horário:**
```
┌──────────────────────────────────────┐
│  [← Voltar]                         │
│                                      │
~~~

~~~~
│  Horários disponíveis:             │
│  25 de Dezembro                    │
│                                     │
│  Manhã:                            │
│  [08:00] [09:00] [10:00] [11:00]  │
│                                     │
│  Tarde:                            │
│  [14:00] [15:00] [16:00] [17:00]  │
│                                     │
│  (12:00 e 13:00 indisponíveis)    │
└─────────────────────────────────────┘
```

**Após selecionar horário → Tela de Dados:**
```
┌─────────────────────────────────────┐
│  [← Voltar]                        │
│                                     │
│  Resumo do Agendamento:            │
│  ✂️ Corte Masculino                  │
│  📅 25/12/2025 às 09:00             │
│  💰 R$ 50,00                         │
│                                     │
│  Seus dados:                       │
│  ┌──────────────────────────────┐  │
│  │ Nome completo                │  │
│  └──────────────────────────────┘  │
│                                     │
│  ┌──────────────────────────────┐  │
│  │ (11) 9____-____              │  │
│  └──────────────────────────────┘  │
│                                     │
│  [ ] Já sou cliente                │
│                                     │
│  [Confirmar Agendamento]           │
│                                     │
│  Você receberá uma confirmação    │
│  no WhatsApp 📱                     │
└─────────────────────────────────────┘
```

**Tela de Sucesso:**
```
┌─────────────────────────────────────┐
│         ✅                           │
│                                     │
│  Agendamento Confirmado!           │
│                                     │
│  João Silva, seu horário está     │
│  garantido!                        │
│                                     │
│  📅 25 de Dezembro às 09:00         │
│  ✂️ Corte Masculino                  │
│  📍 Rua das Flores, 123            │
│                                     │
│  Você receberá:                    │
│  • Confirmação agora no WhatsApp  │
│  • Lembrete 24h antes             │
│                                     │
│  [Adicionar ao Calendário]        │
│  [Voltar ao Início]               │
│                                     │
│  Precisa cancelar?                 │
│  Fale com João: (11) 99999-9999   │
└─────────────────────────────────────┘
```

---

## 🤖 Integração com Evolution API (WhatsApp)

### Como Funciona a Automação
~~~~

~~~
#### 1. **Setup da Evolution API (Gratuito)**

**Opções de Hospedagem:**
- **Railway** (recomendado): 500h gratuitas/mês
- **Render**: 750h gratuitas/mês
- **VPS própria**: $5-10/mês (alternativa)

**Instalação no Railway:**
```bash
1. Criar conta no Railway (railway.app)
2. Usar template da Evolution API:
   - GitHub: https://github.com/EvolutionAPI/evolution-api
3. Clicar em "Deploy"
4. Configurar variáveis de ambiente:
   - AUTHENTICATION_API_KEY=[REDACTED]
   - DATABASE_PROVIDER=postgresql
5. Deploy automático em ~3 minutos
6. Copiar URL: https://seu-app.railway.app
```

#### 2. **Conectar Instância no Sistema**

**Fluxo no Dashboard:**
```
Profissional → Configurações → WhatsApp
↓
Informar URL da Evolution API + API Key
↓
Sistema testa conexão (endpoint /instance/connect)
↓
Gera QR Code para conectar WhatsApp
↓
Profissional escaneia com WhatsApp
↓
✅ Conectado!
```

#### 3. **Endpoints Utilizados**

**a) Criar/Conectar Instância:**
```javascript
POST /instance/create
{
  "instanceName": "barbearia-joao",
  "token": "sua-chave-api",
  "qrcode": true
}

// Retorna QR Code para conectar
```

**b) Enviar Mensagem de Confirmação:**
```javascript
POST /message/sendText/{instanceName}
{
  "number": "5511999999999",
  "text": "Olá João Silva! 👏\n\nSeu agendamento foi confirmado:\n📅 25/12/2025 às 09:00\n✂️ Corte Masculino\n💰 R$
50,00\n\nNos vemos em breve!\nBarbearia do João"
}
```

**c) Agendar Lembrete (usando cron job):**
```javascript
// No backend, criar job agendado:
// Verifica a cada 1 hora se há agendamentos nas próximas 24h

SELECT * FROM agendamentos
WHERE data = CURRENT_DATE + 1
  AND lembrete_enviado = false
  AND status = 'confirmado'
~~~

~~~
// Para cada agendamento encontrado:
POST /message/sendText/{instanceName}
{
  "number": "5511999999999",
  "text": "Oi João Silva! 🔔\n\nLembrete: você tem agendamento AMANHÃ às 09:00\n\nSe não puder comparecer, me
avise!\n(11) 98888-8888"
}

// Após enviar, marcar lembrete_enviado = true
```

#### 4. **Fallback Manual (Caso Evolution API não esteja configurada)**

**Quando profissional não tem WhatsApp automatizado:**

```javascript
// Sistema gera link do WhatsApp Web com mensagem pronta

function gerarLinkWhatsApp(agendamento) {
  const numero = agendamento.cliente_telefone.replace(/\D/g, '');
  const mensagem = encodeURIComponent(
    `Olá ${agendamento.cliente_nome}! 👋\n\n` +
    `Seu agendamento foi confirmado:\n` +
    `📅 ${agendamento.data} às ${agendamento.hora_inicio}\n` +
    `✂️ ${agendamento.servico.nome}\n` +
    `💰 R$ ${agendamento.servico.preco}\n\n` +
    `Nos vemos em breve!\n` +
    `${usuario.nome_negocio}`
  );

  return `https://wa.me/55${numero}?text=${mensagem}`;
}

// No dashboard, mostrar:
// "Clique para enviar confirmação: [Abrir WhatsApp]"
```

---

## ⚙️ Funcionalidades Avançadas (Diferenciais)

### 1. **Sistema de Notificações Inteligentes**
- Envio automático de confirmação (imediato)
- Lembrete 24h antes (padrão, configurável)
- Lembrete 2h antes (opcional)
- Mensagem de agradecimento pós-atendimento (opcional)

### 2. **Gestão de No-Shows**
- Marcar cliente como "não compareceu"
- Sistema sugere enviar mensagem automática perguntando motivo
- Histórico de no-shows por cliente
- Opção de pedir confirmação prévia para clientes com histórico

### 3. **Bloqueios e Disponibilidade**
- Bloquear horários pontuais (almoço, compromissos)
- Bloquear dias inteiros (férias, feriados)
- Bloqueios recorrentes (ex: toda segunda 12h-13h)

### 4. **Clientes Recorrentes**
- Sistema reconhece telefone do cliente
- Autopreenchimento de dados
- Histórico de agendamentos
- Sugestão de "mesmo horário de sempre"

### 5. **Personalização da Página**
- Upload de foto/logo
- Cores personalizadas (tema)
- Descrição do negócio
- Link para Instagram/redes sociais
- Galeria de fotos (trabalhos realizados)

### 6. **Métricas Básicas**
~~~

```
- Taxa de no-show (%)
- Receita prevista vs realizada
- Serviços mais agendados
- Horários de pico
- Clientes novos vs recorrentes

### 8. **Gestão de Equipe (Multi-usuário)** ⭐ ESSENCIAL
- Sistema de permissões granulares (Owner/Admin/Funcionário)
- Funcionários veem apenas seus próprios agendamentos
- Dono vê tudo e pode filtrar por funcionário
- Controle de visibilidade de valores financeiros
- Relatórios segmentados por profissional
- Cliente escolhe o profissional ao agendar (ou "qualquer um disponível")

### 9. **Integração com Google Calendar** (Futuro)
- Sincronização bidirecional
- Bloquear horários marcados no Google automaticamente

---

## 🚀 Roadmap de Desenvolvimento

### FASE 1 - MVP
**Objetivo:** Sistema funcional para validar com primeiros clientes

- [x] Setup do projeto (React + Vite + Supabase)
- [x] Sistema de autenticação (login/cadastro)
- [x] CRUD de serviços
- [x] Lógica de cálculo de horários disponíveis
- [x] Página pública de agendamento
- [x] Dashboard com visualização de agenda (dia)
- [x] Sistema de bloqueios simples
- [x] Geração de link de WhatsApp manual (sem Evolution API)
- [x] Deploy na Vercel

**Entrega:** Sistema usável onde profissional pode receber agendamentos e enviar confirmações manualmente via WhatsApp.

---

### FASE 2 - Automação
**Objetivo:** Reduzir trabalho manual do profissional

- [x] Integração com Evolution API
- [x] QR Code para conectar WhatsApp
- [x] Envio automático de confirmação
- [x] Cron job para lembretes automáticos
- [x] Configuração de mensagens personalizadas
- [x] Teste de envio de mensagens

**Entrega:** Sistema 100% automatizado para mensagens.

---

### FASE 3 - Experiência do Usuário
**Objetivo:** Melhorar usabilidade e conversão

- [x] Onboarding completo pós-cadastro
- [x] Tutorial interativo no dashboard
- [x] Visualização semanal da agenda
- [x] Filtros e busca na agenda
- [x] Status visual de agendamentos (cores)
- [x] Página de agendamento com foto/logo
- [x] Responsividade total (mobile-first)

**Entrega:** Sistema polido e fácil de usar.

---

### FASE 4 - Gestão Avançada
**Objetivo:** Dar mais controle ao profissional

- [x] Gestão de clientes recorrentes
```

```
- [x] Histórico de agendamentos por cliente
- [x] Sistema de no-shows
- [x] Relatórios básicos (dashboard de métricas)
- [x] Bloqueios recorrentes
- [x] Exportação de agenda (CSV)

**Entrega:** Sistema completo de gestão.

---

### FASE 5 - Painel Super Admin (CRÍTICO) ⭐⭐
**Objetivo:** Você conseguir gerenciar TODOS os clientes

- [x] Sistema de autenticação super admin (OBS: depende da tabela `public.super_admin` + RLS do bloco em `/admin/configuracoes`)
- [x] Dashboard com visão geral (MRR, clientes ativos, etc) (OBS: hoje mostra total/ativos/inadimplentes; não calcula MRR)
- [x] Lista de todos os clientes (com filtros) (OBS: busca + filtros por `plano`, `status_pagamento`, `ativo`; lista até 500)
- [x] Detalhes completos de cada cliente
- [x] **"Logar como cliente" (impersonation)** (OBS: troca o `appPrincipal` no front; não troca o JWT/sessão do Supabase)
- [x] Gerenciar funcionários dos clientes (OBS: cria via Edge Function `admin-create-funcionario`; precisa deploy + `SERVICE_ROLE_KEY`)
- [x] Alterar planos manualmente (OBS: altera `usuarios.plano`, `usuarios.status_pagamento` e `usuarios.limite_funcionarios`)
- [x] Suspender/reativar contas (OBS: seta `usuarios.ativo`; bloqueia login/acesso via `RequireAuth`)
- [x] Logs de auditoria (tudo que você faz é registrado) (OBS: tela `/admin/logs` + SQL de triggers em `/admin/configuracoes`; precisa executar no Supabase)
- [ ] Configurações de planos e preços (OBS: planos/preços ainda ficam hardcoded + env vars do Stripe; sem tela/tabela de configuração)
- [x] Integração com gateway de pagamento (OBS: Stripe Checkout via Edge Function `payments` + webhook na própria função; requer configurar `STRIPE_WEBHOOK_SECRET` e endpoint de webhook no Stripe)

**Entrega:** Você com controle total do sistema.

---

### FASE 6 - Sistema Multi-usuário (ALTA PRIORIDADE)
**Objetivo:** Permitir que donos de negócio gerenciem funcionários

- [x] Sistema de permissões (Owner/Admin/Funcionário)
- [x] CRUD de funcionários
- [x] Dashboard do funcionário (visão limitada)
- [x] Agendamentos por funcionário
- [x] Relatórios segmentados por funcionário
- [x] Controle de acesso granular
- [x] **Validação de limites por plano**

**Entrega:** Sistema completo para salões/barbearias com equipe.

---

### FASE 7 - Monetização
**Objetivo:** Implementar planos pagos

- [x] Integração com gateway de pagamento (Stripe)
- [x] Sistema de planos (Free, Basic, Pro, Team, Enterprise)
- [x] Limites por plano (ex: free = 30 agendamentos/mês)
- [x] Página de upgrade
- [x] Painel administrativo (para você gerenciar clientes)

**Entrega:** Sistema pronto para gerar receita recorrente.

---

### BACKLOG
- [ ] App mobile nativo (quando tiver CNPJ)
- [ ] Integração com Google Calendar
- [ ] Sistema de avaliações/reviews
- [ ] Pagamento online na hora de agendar
- [ ] Programa de fidelidade (pontos)
```

```
- [ ] WhatsApp chatbot para agendar por conversa

---

## 💰 Estratégia de Monetização

### Planos Sugeridos

#### 🆓 Plano FREE
**R$ 0/mês - Para testar**
- Até 30 agendamentos por mês
- 1 profissional
- Lembretes manuais (link do WhatsApp)
- Suporte por email

#### ⭐ Plano BASIC
**R$ 49,90/mês**
- Agendamentos 60 por mês
- 1 profissional
- **Lembretes automáticos via WhatsApp**
- Até 3 serviços
- Personalização básica da página
- Suporte prioritário

#### 🚀 Plano PRO
**R$ 79,90/mês**
- Tudo do Basic +
- Agendamentos 180 por mês
- Serviços ilimitados
- Gestão de clientes (histórico completo)
- Relatórios avançados
- Bloqueios recorrentes
- **Até 3 funcionários** ⭐
- Logo e galeria de fotos
- Suporte via WhatsApp

#### 💼 Plano TEAM ⭐ NOVO
**R$ 119,90/mês**
- Tudo do Pro +
- **Até 5 funcionários**
- Agendamentos 300 por mês
- Agenda unificada (filtro por profissional)
- Relatórios por funcionário
- Controle de permissões detalhado
- Cliente escolhe o profissional
- Suporte prioritário

#### 🏢 Plano ENTERPRISE ⭐ NOVO
**R$ 199,90/mês**
- Tudo do Team +
- Agendamentos ilimitados
- **Funcionários ilimitados**
- Multi-unidades (filiais)
- API de integração
- Suporte dedicado via WhatsApp
- Treinamento da equipe incluso

---

### Serviços Adicionais

**Setup Completo:** R$ 150 (uma vez)
- Você configura tudo para o cliente
- Cadastra serviços, fotos, horários
- Conecta WhatsApp
- Testa envios
- Treina o cliente em 15 minutos

**Consultoria por Hora:** R$ 80/hora
- Ajuda com configurações avançadas
- Sugestões de otimização
- Dúvidas gerais
```

~~~
---

## 📊 Métricas de Sucesso

### Para Validação do MVP
- ✅ 5 profissionais usando ativamente
- ✅ 100+ agendamentos realizados
- ✅ Taxa de no-show reduzida em pelo menos 30%
- ✅ NPS (Net Promoter Score) acima de 8

### Para Crescimento
- 50 profissionais pagantes em 6 meses
- MRR de R$ 3.000 em 6 meses
- Taxa de churn abaixo de 10%/mês
- Tempo médio de setup: menos de 10 minutos

---

## 🎯 Público-Alvo Detalhado

### Persona 1: **Barbeiro Autônomo**
- **Idade:** 25-40 anos
- **Dor:** Clientes marcam e não aparecem (perde R$ 200-500/mês)
- **Comportamento:** Usa WhatsApp para tudo, não gosta de sistemas complicados
- **Onde encontrar:** Instagram, grupos de Facebook, indicação

### Persona 2: **Manicure Home-Care**
- **Idade:** 28-45 anos
- **Dor:** Desorganização da agenda, esquecem horários, perdem tempo no telefone
- **Comportamento:** Atende em domicílio, precisa de mobilidade
- **Onde encontrar:** Instagram, WhatsApp Status, boca a boca

### Persona 3: **Personal Trainer**
- **Idade:** 25-38 anos
- **Dor:** Alunos cancelam em cima da hora, dificulta reposição
- **Comportamento:** Tech-savvy, gosta de otimizar tempo
- **Onde encontrar:** Instagram fitness, grupos de crossfit/academias

---

## 🔧 Código Base - Estrutura de <invoke name="artifacts">
<parameter name="command">update</parameter>
<parameter name="id">scheduling-system-doc</parameter>
<parameter name="old_str">- Botão "Compartilhar no WhatsApp"
- Botão "Ir para minha agenda"

---</parameter>
<parameter name="new_str">
  const buffer = XLSX.write(workbook, { type: 'buffer', bookType: 'xlsx' });

  return {
    content: buffer,
    filename: `${filename}_${new Date().toISOString().split('T')[0]}.xlsx`,
    mimeType: 'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet'
  };
}

export {
  generateMRRReport,
  generateChurnReport,
  generateInadimplenciaReport,
  exportToCSV,
  exportToExcel
};
```

#### Rota da API para Gerar Relatórios

```javascript
// routes/admin/reports.js
~~~

~~~
import express from 'express';
import {
  generateMRRReport,
  generateChurnReport,
  generateInadimplenciaReport,
  exportToCSV,
  exportToExcel
} from '../services/reportService.js';
import { requireSuperAdmin } from '../middleware/superAdmin.js';

const router = express.Router();

// Endpoint para gerar relatório completo
router.get('/relatorios/completo', requireSuperAdmin, async (req, res) => {
  try {
    const { mes, ano, formato } = req.query;

    // Gera os 3 relatórios
    const [mrr, churn, inadimplencia] = await Promise.all([
      generateMRRReport(parseInt(mes), parseInt(ano)),
      generateChurnReport(parseInt(mes), parseInt(ano)),
      generateInadimplenciaReport()
    ]);

    const reports = { mrr, churn, inadimplencia };

    // Retorna JSON ou exporta arquivo
    if (formato === 'json') {
      return res.json(reports);
    }

    if (formato === 'csv') {
      // Gera CSV separado para cada relatório
      const mrrCSV = exportToCSV(mrr.assinaturas, 'mrr');
      res.setHeader('Content-Type', mrrCSV.mimeType);
      res.setHeader('Content-Disposition', `attachment; filename="${mrrCSV.filename}"`);
      return res.send(mrrCSV.content);
    }

    if (formato === 'excel') {
      const excel = exportToExcel(reports, 'relatorio_financeiro');
      res.setHeader('Content-Type', excel.mimeType);
      res.setHeader('Content-Disposition', `attachment; filename="${excel.filename}"`);
      return res.send(excel.content);
    }

    res.status(400).json({ error: 'Formato inválido. Use: json, csv ou excel' });

  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Erro ao gerar relatório' });
  }
});

// Endpoint para relatório específico de MRR
router.get('/relatorios/mrr', requireSuperAdmin, async (req, res) => {
  const { mes, ano } = req.query;
  const report = await generateMRRReport(parseInt(mes), parseInt(ano));
  res.json(report);
});

// Endpoint para relatório de inadimplência
router.get('/relatorios/inadimplencia', requireSuperAdmin, async (req, res) => {
  const report = await generateInadimplenciaReport();
  res.json(report);
});

export default router;
```

#### Tela de Relatórios (Super Admin)
~~~

~~~
**Nova Tela SA-10:**
```
┌─────────────────────────────────────┐
│ [☰] Relatórios Financeiros          │
├─────────────────────────────────────┤
│                                     │
│ 📊 Gerar Relatório                  │
│                                     │
│ Período:                            │
│ Mês: [Dezembro ▼] Ano: [2025 ▼]     │
│                                     │
│ Formato de Exportação:              │
│ (•) Excel (.xlsx)                   │
│ ( ) CSV                             │
│ ( ) JSON                            │
│                                     │
│ Incluir:                            │
│ [✓] MRR (Receita Recorrente)        │
│ [✓] Churn (Cancelamentos)           │
│ [✓] Inadimplência                   │
│                                     │
│ [Gerar e Baixar Relatório]          │
│                                     │
│ ─────────────────────────────────   │
│                                     │
│ 📈 Relatórios Rápidos               │
│                                     │
│ ┌─────────────────────────────────┐ │
│ │ MRR Atual                       │ │
│ │ R$ 7.320,00                     │ │
│ │ +12% vs mês anterior            │ │
│ │ [Ver Detalhes]                  │ │
│ └─────────────────────────────────┘ │
│                                     │
│ ┌─────────────────────────────────┐ │
│ │ Taxa de Churn                   │ │
│ │ 8% (4 cancelamentos)            │ │
│ │ ✅ Abaixo da meta (10%)          │ │
│ │ [Ver Detalhes]                  │ │
│ └─────────────────────────────────┘ │
│                                     │
│ ┌─────────────────────────────────┐ │
│ │ Inadimplência                   │ │
│ │ R$ 718,00 (4 clientes)          │ │
│ │ ⚠️ Requer atenção                │ │
│ │ [Ver Lista]                     │ │
│ └─────────────────────────────────┘ │
│                                     │
│ ─────────────────────────────────   │
│                                     │
│ 📁 Relatórios Anteriores            │
│                                     │
│ • relatorio_financeiro_2025-11.xlsx│
│   [Baixar] - 2.3 MB - 01/12/2025   │
│                                     │
│ • relatorio_financeiro_2025-10.xlsx│
│   [Baixar] - 2.1 MB - 01/11/2025   │
│                                     │
│ [Ver Todos]                         │
└─────────────────────────────────────┘
```

---

## 📧 Exemplo de Email de Cobrança

```html
<!DOCTYPE html>
<html>
<head>
  <style>
    body { font-family: Arial, sans-serif; line-height: 1.6; color: #333; }
~~~

~~~
    .container { max-width: 600px; margin: 0 auto; padding: 20px; }
    .header { background: #2563eb; color: white; padding: 20px; text-align: center; }
    .content { padding: 30px 20px; background: #f9fafb; }
    .alert { background: #fef2f2; border-left: 4px solid #dc2626; padding: 15px; margin: 20px 0; }
    .button { display: inline-block; background: #2563eb; color: white; padding: 12px 30px; text-decoration: none;
border-radius: 5px; margin: 20px 0; }
    .footer { text-align: center; color: #6b7280; font-size: 12px; padding: 20px; }
  </style>
</head>
<body>
  <div class="container">
    <div class="header">
      <h1>⚠️ Pagamento em Atraso</h1>
    </div>

    <div class="content">
      <p>Olá <strong>João Silva</strong>,</p>

      <p>Identificamos que seu pagamento do <strong>SMagenda</strong> está com <strong>7 dias de atraso</strong>.</p>

      <div class="alert">
        <strong>⚠️ ATENÇÃO:</strong> Sua conta será <strong>SUSPENSA em 7 dias</strong> se não regularizar o pagamento.
      </div>

      <p><strong>Detalhes da Cobrança:</strong></p>
      <ul>
        <li>Plano: <strong>PRO</strong></li>
        <li>Valor: <strong>R$ 99,90</strong></li>
        <li>Vencimento: <strong>13/12/2025</strong></li>
        <li>Dias de atraso: <strong>7 dias</strong></li>
      </ul>

      <p><strong>O que acontece se minha conta for suspensa?</strong></p>
      <ul>
        <li>❌ Você não consegue acessar o sistema</li>
        <li>❌ Seus clientes não conseguem agendar</li>
        <li>❌ Lembretes automáticos param de funcionar</li>
        <li>❌ Seus funcionários ficam bloqueados</li>
      </ul>

      <center>
        <a href="https://smagenda.com/pagamento/abc123" class="button">
          💳 Regularizar Pagamento Agora
        </a>
      </center>

      <p>Caso já tenha realizado o pagamento, desconsidere este email.</p>

      <p>Precisa de ajuda? Responda este email ou entre em contato:</p>
      <p>📧 suporte@smagenda.com<br>
         📱 WhatsApp: (11) 9xxxx-xxxx</p>

      <p>Atenciosamente,<br>
      <strong>Equipe SMagenda</strong></p>
    </div>

    <div class="footer">
      <p>SMagenda - Sistema de Agendamento Inteligente</p>
      <p>© 2025 Todos os direitos reservados</p>
    </div>
  </div>
</body>
</html>
```

---

- Botão "Compartilhar no WhatsApp"
- Botão "Ir para minha agenda"

---</parameter>
~~~

scheduling-system-doc.md

~~~
# 📱 SMagenda - Sistema de Agendamento Inteligente - Documentação Completa

## 🎯 Visão Geral do Produto

**Nome:** SMagenda
**Proposta de Valor:** "Pare de perder dinheiro com clientes que não aparecem. Automatize lembretes e organize sua agenda em um só lugar."

**Diferencial Principal:** Sistema 100% mobile, link direto (sem necessidade de app), lembretes automáticos via WhatsApp e setup em menos de 10 minutos.

### Persona 4: **Dono de Barbearia com Equipe** ⭐ NOVO
- **Idade:** 30-50 anos
- **Dor:** Desorganização da equipe, não sabe quem atendeu quem, dificuldade de controlar horários
- **Comportamento:** Quer profissionalizar o negócio, precisa de controle sem microgerenciar
- **Onde encontrar:** Associações de barbearias, grupos no WhatsApp, eventos do setor
- **Ticket médio:** R$ 179-299/mês (maior valor!)

---

#### Tela 9: **Gestão de Funcionários** (Somente Master) ⭐ NOVO
**URL:** `/funcionarios`

**Layout:**
```
┌─────────────────────────────────────┐
│ [☰ Menu]  Funcionários    [+ Novo] │
├─────────────────────────────────────┤
│                                     │
│  ┌─────────────────────────────────┐│
│  │ 👤 Carlos Silva          [✏️] ││
│  │ carlos@email.com                ││
│  │ 📱 (11) 98888-8888              ││
│  │                                 ││
│  │ 🟢 Ativo • Funcionário           ││
│  │ 📅 Atende: Seg-Sex 9h-18h        ││
│  │ ✂️ 23 agendamentos este mês        ││
│  │                                 ││
│  │ Permissões:                     ││
│  │ ✅ Ver agenda                    ││
│  │ ✅ Criar agendamentos            ││
│  │ ❌ Ver financeiro                ││
│  │ ❌ Gerenciar serviços            ││
│  │                                 ││
│  │ [Editar] [Desativar]            ││
│  └─────────────────────────────────┘│
│                                     │
│  ┌─────────────────────────────────┐│
│  │ 👤 Maria Santos          [✏️] ││
│  │ maria@email.com                 ││
│  │ 📱 (11) 97777-7777              ││
│  │                                 ││
│  │ 🟢 Ativo • Admin                 ││
│  │ 📅 Atende: Ter-Sab 10h-19h       ││
│  │ ✂️ 31 agendamentos este mês        ││
│  │                                 ││
│  │ Permissões:                     ││
│  │ ✅ Ver agenda                    ││
│  │ ✅ Criar agendamentos            ││
│  │ ✅ Ver financeiro                ││
│  │ ✅ Gerenciar serviços            ││
│  │                                 ││
│  │ [Editar] [Desativar]            ││
│  └─────────────────────────────────┘│
└─────────────────────────────────────┘
```

**Modal de Adicionar Funcionário:**
```
┌─────────────────────────────────────┐
~~~

~~~
│ Adicionar Funcionário        [✕]  │
├───────────────────────────────────┤
│                                   │
│ Dados Básicos:                   │
│ ┌───────────────────────────┐    │
│ │ Nome completo             │    │
│ └───────────────────────────┘    │
│                                   │
│ ┌───────────────────────────┐    │
│ │ Email (para login)        │    │
│ └───────────────────────────┘    │
│                                   │
│ ┌───────────────────────────┐    │
│ │ (11) 9____-____           │    │
│ └───────────────────────────┘    │
│                                   │
│ ┌───────────────────────────┐    │
│ │ Senha inicial             │    │
│ └───────────────────────────┘    │
│                                   │
│ Nível de Acesso:                 │
│ ( ) Admin - Acesso quase total   │
│ (•) Funcionário - Acesso limitado │
│                                   │
│ Horário de Trabalho:             │
│ Das [09:00] às [18:00]           │
│ Dias: [S][T][Q][Q][S][S][D]      │
│                                   │
│ Permissões Detalhadas:           │
│ [✓] Ver agenda                   │
│ [✓] Criar agendamentos           │
│ [✓] Cancelar agendamentos        │
│ [✓] Bloquear horários próprios   │
│ [ ] Ver valores/financeiro       │
│ [ ] Gerenciar serviços           │
│ [ ] Ver clientes de outros       │
│                                   │
│     [Cancelar] [Criar Acesso]   │
└───────────────────────────────────┘
```

---

#### Tela 10: **Dashboard do Funcionário** (Visão Limitada) ⭐ NOVO
**URL:** `/funcionario/agenda`

**Diferenças do Dashboard Master:**

```
┌───────────────────────────────────┐
│ [☰ Menu]    SMagenda      [🔔][👤] │
│ Olá, Carlos! 👋                    │
├───────────────────────────────────┤
│                                   │
│  [< Hoje - 25 Dez >]    [+ Novo]  │
│                                   │
│  🔍 Mostrando: Meus Agendamentos   │
│                                   │
│  ┌─────────────────────────────┐  │
│  │ 09:00 - João Silva          │  │
│  │ 📱 (11) 99999-9999          │  │
│  │ ✂️ Corte Masculino           │  │
│  │ [✓ Confirmar] [✗ Cancelar]  │  │
│  ├─────────────────────────────┤  │
│  │ 10:00 - LIVRE               │  │
│  ├─────────────────────────────┤  │
│  │ 11:00 - Maria Santos        │  │
│  │ 📱 (11) 98888-8888          │  │
│  │ 💇‍♀️ Escova                   │  │
│  │ ✅ Confirmado                │  │
│  └─────────────────────────────┘  │
│                                   │
~~~

~~~
│  Resumo do Seu Dia:             │
│  ⏰ 5 agendamentos               │
│  (valores ocultos)              │
└─────────────────────────────────┘
```

**Menu do Funcionário (Limitado):**
- 📅 Minha Agenda
- 👥 Meus Clientes (só os que ele atendeu)
- 🔒 Meus Horários Bloqueados
- ⚙️ Meu Perfil
- ❓ Ajuda
- 🚪 Sair

**O que o Funcionário NÃO vê:**
- ❌ Valores/receitas (a menos que tenha permissão)
- ❌ Agendamentos de outros funcionários
- ❌ Configurações do negócio
- ❌ Gestão de serviços (preços)
- ❌ Mensagens automáticas
- ❌ Evolution API
- ❌ Planos/pagamentos

---

#### Tela 11: **Agenda Completa do Dono** (Visão Master) ⭐ NOVO
**URL:** `/dashboard` (com filtros)

```
┌─────────────────────────────────┐
│ [☰ Menu]    AgendaFácil    [🔔][👤] │
├─────────────────────────────────┤
│                                 │
│  [< Hoje - 25 Dez >]    [+ Novo]  │
│                                 │
│  Filtrar por:                   │
│  [Todos] [Carlos] [Maria] [João]  │
│                                 │
│  ┌───────────────────────────┐  │
│  │ 09:00 👤 Carlos            │  │
│  │ Cliente: João Silva        │  │
│  │ ✂️ Corte - R$ 50             │  │
│  ├───────────────────────────┤  │
│  │ 09:00 👤 Maria             │  │
│  │ Cliente: Ana Costa         │  │
│  │ 💇 Escova - R$ 40            │  │
│  ├───────────────────────────┤  │
│  │ 10:00 👤 Carlos - LIVRE    │  │
│  ├───────────────────────────┤  │
│  │ 10:00 👤 Maria             │  │
│  │ Cliente: Beatriz           │  │
│  │ 💅 Manicure - R$ 35          │  │
│  └───────────────────────────┘  │
│                                 │
│  Resumo do Dia:                 │
│  💰 R$ 650,00 • 15 agendamentos   │
│  👤 Carlos: R$ 250 (5 clientes)   │
│  👤 Maria: R$ 400 (10 clientes)   │
└─────────────────────────────────┘
```

---

### 👤 LADO DO CLIENTE (Página de Agendamento com Funcionários)

#### Tela 12: **Escolher Funcionário** ⭐ NOVO
**URL:** `/agendar/{slug}`

**Após escolher o serviço, antes da data:**

```
┌─────────────────────────────────┐
~~~

~~~~
│  [← Voltar]                          │
│                                       │
│  Com qual profissional?              │
│                                       │
│  ┌─────────────────────────────────┐ │
│  │ [Foto] Carlos Silva             ││
│  │ ⭐⭐⭐⭐⭐ (45 avaliações)       ││
│  │ Especialidade: Cortes modernos  ││
│  │ [Selecionar]                    ││
│  └─────────────────────────────────┘ │
│                                       │
│  ┌─────────────────────────────────┐ │
│  │ [Foto] Maria Santos             ││
│  │ ⭐⭐⭐⭐⭐ (62 avaliações)       ││
│  │ Especialidade: Coloração        ││
│  │ [Selecionar]                    ││
│  └─────────────────────────────────┘ │
│                                       │
│  ┌─────────────────────────────────┐ │
│  │ 🤷‍♀️ Tanto faz, primeiro disponível││
│  │ [Selecionar]                    ││
│  └─────────────────────────────────┘ │
│                                       │
└───────────────────────────────────────┘
```

**Lógica:**
- Cliente escolhe o profissional OU "tanto faz"
- Sistema mostra horários apenas daquele profissional
- Se escolher "tanto faz", mostra TODOS os horários livres de qualquer um

---

## 🏗️ Arquitetura Técnica

### Stack Escolhida (100% Gratuita)

| Componente | Tecnologia | Custo | Limite Gratuito |
|------------|------------|-------|-----------------|
| **Frontend** | React + Vite + TailwindCSS | R$ 0 | Ilimitado |
| **Backend/BD** | Supabase | R$ 0 | 500MB BD + 2GB storage + 50k usuários |
| **Hospedagem** | Vercel | R$ 0 | 100GB bandwidth/mês |
| **WhatsApp** | Evolution API v2 | R$ 0 | Self-hosted (Railway free tier) |
| **Domínio** | Hostinger/Registro.br | R$ 40/ano | - |

### Tipo de Aplicação
**PWA (Progressive Web App)** - Funciona como site + app sem precisar de lojas

---

## 🗄️ Estrutura do Banco de Dados

### Tabelas Principais

#### 1. **usuarios** (Profissionais - Conta Master)
```sql
id: UUID (PK)
nome_completo: TEXT
nome_negocio: TEXT
slug: TEXT (UNIQUE) -- ex: "barbearia-do-joao"
telefone: TEXT
email: TEXT (UNIQUE)
senha_hash: TEXT
foto_perfil: TEXT (URL)
endereco: TEXT (opcional)
horario_inicio: TIME -- ex: "08:00"
horario_fim: TIME -- ex: "18:00"
dias_trabalho: JSONB -- ex: [1,2,3,4,5,6] (seg a sab)
intervalo_inicio: TIME (opcional) -- ex: "12:00"
intervalo_fim: TIME (opcional) -- ex: "13:00"
whatsapp_api_url: TEXT (URL da Evolution API)
whatsapp_api_key=[REDACTED]
~~~~

~~~
plano: TEXT (free, basic, pro, team, enterprise)
tipo_conta: TEXT -- 'master', 'individual'
limite_funcionarios: INTEGER -- baseado no plano
status_pagamento: TEXT -- 'ativo', 'inadimplente', 'cancelado', 'trial'
data_cadastro: TIMESTAMP
data_vencimento: DATE -- próximo pagamento
ativo: BOOLEAN
```

#### 0. **super_admin** (VOCÊ - Administrador do Sistema) ⭐⭐ NOVO
```sql
id: UUID (PK)
nome: TEXT
email: TEXT (UNIQUE)
senha_hash: TEXT
nivel: TEXT -- 'super_admin', 'suporte'
ultimo_acesso: TIMESTAMP
criado_em: TIMESTAMP
```

#### 1.1. **funcionarios** (Sub-contas/Colaboradores) ⭐ NOVO
```sql
id: UUID (PK)
usuario_master_id: UUID (FK -> usuarios) -- dono do negócio
nome_completo: TEXT
email: TEXT (UNIQUE)
senha_hash: TEXT
telefone: TEXT
foto_perfil: TEXT (URL)
permissao: TEXT -- 'admin', 'funcionario'
horario_inicio: TIME -- horário de trabalho deste funcionário
horario_fim: TIME
dias_trabalho: JSONB
intervalo_inicio: TIME (opcional)
intervalo_fim: TIME (opcional)
pode_ver_financeiro: BOOLEAN -- se pode ver valores/receitas
pode_gerenciar_servicos: BOOLEAN
pode_bloquear_horarios: BOOLEAN
pode_cancelar_agendamentos: BOOLEAN
ativo: BOOLEAN
criado_em: TIMESTAMP
desativado_em: TIMESTAMP (nullable)
```

#### 2. **servicos**
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios)
nome: TEXT -- ex: "Corte Masculino"
descricao: TEXT (opcional)
duracao_minutos: INTEGER -- ex: 45
preco: DECIMAL(10,2)
cor: TEXT -- para visualização na agenda (ex: "#FF5733")
ativo: BOOLEAN
ordem: INTEGER -- para ordenar na listagem
```

#### 3. **agendamentos**
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios) -- dono do negócio
funcionario_id: UUID (FK -> funcionarios) -- quem vai atender ⭐ NOVO
servico_id: UUID (FK -> servicos)
cliente_nome: TEXT
cliente_telefone: TEXT
data: DATE
hora_inicio: TIME
hora_fim: TIME (calculado automaticamente)
status: TEXT -- 'confirmado', 'cancelado', 'concluido', 'nao_compareceu'
lembrete_enviado: BOOLEAN
data_lembrete: TIMESTAMP (quando foi enviado)
observacoes: TEXT (opcional)
~~~

~~~
criado_em: TIMESTAMP
criado_por: UUID -- pode ser usuario_id ou funcionario_id ⭐ NOVO
cancelado_em: TIMESTAMP (nullable)
cancelado_por: UUID (nullable) ⭐ NOVO
```

#### 4. **clientes** (Cache de clientes recorrentes)
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios)
nome: TEXT
telefone: TEXT
total_agendamentos: INTEGER
ultimo_agendamento: TIMESTAMP
criado_em: TIMESTAMP

UNIQUE(usuario_id, telefone)
```

#### 5. **bloqueios** (Horários bloqueados pelo profissional)
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios)
data: DATE
hora_inicio: TIME
hora_fim: TIME
motivo: TEXT (opcional) -- "Almoço", "Compromisso pessoal"
criado_em: TIMESTAMP
```

#### 6. **configuracoes_mensagens**
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios)
mensagem_confirmacao: TEXT
mensagem_lembrete: TEXT
horas_antecedencia: INTEGER -- default: 24
enviar_confirmacao: BOOLEAN
enviar_lembrete: BOOLEAN
```

#### 7. **logs_admin** (Auditoria das suas ações) ⭐⭐ NOVO
```sql
id: UUID (PK)
super_admin_id: UUID (FK -> super_admin)
usuario_afetado_id: UUID (FK -> usuarios) -- qual cliente foi afetado
acao: TEXT -- 'login_como_usuario', 'excluir_funcionario', 'alterar_plano', 'suspender_conta', 'reativar_conta'
detalhes: JSONB -- dados da ação
ip_address: TEXT
criado_em: TIMESTAMP
```

#### 8. **assinaturas** (Controle de pagamentos) ⭐⭐ NOVO
```sql
id: UUID (PK)
usuario_id: UUID (FK -> usuarios)
plano: TEXT
valor: DECIMAL(10,2)
status: TEXT -- 'ativa', 'cancelada', 'inadimplente', 'trial'
metodo_pagamento: TEXT -- 'cartao', 'boleto', 'pix'
data_inicio: DATE
data_vencimento: DATE
data_cancelamento: DATE (nullable)
gateway_subscription_id: TEXT -- ID do Stripe/Mercado Pago
criado_em: TIMESTAMP
```

---

## 🎨 Telas Detalhadas

---
~~~

~~~
## 🔧 PAINEL SUPER ADMIN (VOCÊ - Dono do Sistema) ⭐⭐ NOVO

### Tela SA-1: **Login Super Admin**
**URL:** `/admin/login`

```
┌─────────────────────────────────────┐
│                                     │
│         🛠️ SUPER ADMIN                │
│         SMagenda                    │
│                                     │
│  ┌─────────────────────────────┐   │
│  │ Email                       │   │
│  └─────────────────────────────┘   │
│                                     │
│  ┌─────────────────────────────┐   │
│  │ Senha                       │   │
│  └─────────────────────────────┘   │
│                                     │
│  [Entrar no Painel Admin]         │
│                                     │
│  ⚠️ Acesso restrito                  │
└─────────────────────────────────────┘
```

---

### Tela SA-2: **Dashboard Super Admin**
**URL:** `/admin/dashboard`

```
┌─────────────────────────────────────┐
│ [☰] SMagenda - Admin       [👤][🚪]│
├─────────────────────────────────────┤
│                                     │
│  📊 Visão Geral                    │
│                                     │
│  ┌──────┐ ┌──────┐ ┌──────┐        │
│  │  47  │ │ R$   │ │ 312  │        │
│  │Clien-│ │7.320 │ │Func. │        │
│  │tes   │ │ MRR  │ │Total │        │
│  └──────┘ └──────┘ └──────┘        │
│                                     │
│  ┌──────────────────┐ ┌─────────┐  │
│  │ 🟢 38 ativos      │ │ Últimas ││
│  │ 🟡 5 trial        │ │ Ações:  ││
│  │ 🔴 4 inadimp.     │ │         ││
│  └──────────────────┘ │ • Login ││
│                       │   João  ││
│  Crescimento:         │         ││
│  📈 +12% este mês      │ • Plano ││
│                       │   Maria ││
│  Churn: 8%            │   ->PRO ││
│  ✅ Meta: <10%         │         ││
│                       └─────────┘│
└─────────────────────────────────────┘
```

**Menu Lateral:**
- 🏠 Dashboard
- 👥 Todos os Clientes
- 💰 Assinaturas
- 📊 Relatórios
- 🔍 Logs de Auditoria
- ⚙️ Configurações do Sistema
- 🚪 Sair

---

### Tela SA-3: **Lista de Clientes (Todos os Masters)**
**URL:** `/admin/clientes`
~~~

~~~
```
┌─────────────────────────────────────┐
│ [☰] Todos os Clientes               │
├─────────────────────────────────────┤
│                                     │
│ Buscar: [____________] [🔍]         │
│                                     │
│ Filtros:                            │
│ [Todos] [Ativos] [Trial] [Inadimp]│
│ [Free] [Basic] [Pro] [Team] [Ent] │
│                                     │
│ ┌─────────────────────────────────┐│
│ │ 🟢 Barbearia do João            ││
│ │ João Silva • joao@email.com     ││
│ │ 📱 (11) 99999-9999              ││
│ │                                 ││
│ │ Plano: PRO (R$ 99,90/mês)       ││
│ │ Status: ✅ Ativo                ││
│ │ Venc: 05/01/2026                ││
│ │ Funcionários: 2/2               ││
│ │ Agendamentos: 234 (total)       ││
│ │ Cadastro: 15/08/2025            ││
│ │                                 ││
│ │ [🔐 Logar Como] [✏️ Editar]     ││
│ │ [📊 Detalhes] [⚠️ Suspender]    ││
│ └─────────────────────────────────┘│
│                                     │
│ ┌─────────────────────────────────┐│
│ │ 🟡 Salão da Maria               ││
│ │ Maria Santos • maria@email.com  ││
│ │ 📱 (11) 98888-8888              ││
│ │                                 ││
│ │ Plano: TRIAL → PRO              ││
│ │ Status: ⏳ Trial (5 dias rest.) ││
│ │ Venc: 30/12/2025                ││
│ │ Funcionários: 1/2               ││
│ │ Agendamentos: 23 (trial)        ││
│ │ Cadastro: 20/12/2025            ││
│ │                                 ││
│ │ [🔐 Logar Como] [✏️ Editar]     ││
│ │ [📊 Detalhes] [💰 Cobrar]       ││
│ └─────────────────────────────────┘│
│                                     │
│ ┌─────────────────────────────────┐│
│ │ 🔴 Studio de Tatuagem           ││
│ │ Carlos Mendes • carlos@email.com││
│ │ 📱 (11) 97777-7777              ││
│ │                                 ││
│ │ Plano: TEAM (R$ 179,90/mês)     ││
│ │ Status: ⚠️ Inadimplente (12d)   ││
│ │ Venc: 13/12/2025                ││
│ │ Funcionários: 4/5 (bloqueados) ││
│ │ Agendamentos: 567 (total)       ││
│ │ Cadastro: 03/05/2025            ││
│ │                                 ││
│ │ [🔐 Logar Como] [✏️ Editar]     ││
│ │ [📧 Enviar Cobrança] [❌ Canc.]││
│ └─────────────────────────────────┘│
│                                     │
│ Mostrando 3 de 47 clientes          │
│ [← 1 2 3 4 5 →]                    │
└─────────────────────────────────────┘
```

---

### Tela SA-4: **Detalhes do Cliente**
**URL:** `/admin/clientes/{id}`

```
┌─────────────────────────────────────┐
~~~

~~~
│ [← Voltar] Barbearia do João         │
├──────────────────────────────────────┤
│                                      │
│ 👤 Informações do Dono               │
│ ┌─────────────────────────────────┐ │
│ │ Nome: João Silva                 ││
│ │ Email: joao@email.com            ││
│ │ Telefone: (11) 99999-9999        ││
│ │ Slug: barbearia-do-joao          ││
│ │ Link: smagenda.com/...           ││
│ │ Cadastro: 15/08/2025             ││
│ │ Último acesso: Há 2 horas        ││
│ └─────────────────────────────────┘ │
│                                      │
│ 💳 Assinatura                        │
│ ┌─────────────────────────────────┐ │
│ │ Plano: PRO                       ││
│ │ Valor: R$ 99,90/mês              ││
│ │ Status: 🟢 Ativo                  ││
│ │ Próximo venc: 05/01/2026         ││
│ │ Método: Cartão (••4532)          ││
│ │ [Alterar Plano] [Ver Histórico]││
│ └─────────────────────────────────┘ │
│                                      │
│ 👥 Funcionários (2/2 permitidos)     │
│ ┌─────────────────────────────────┐ │
│ │ • Carlos Silva (Funcionário)     ││
│ │   carlos@email.com               ││
│ │   [Ver] [❌ Excluir]               ││
│ │                                  ││
│ │ • Ana Costa (Admin)              ││
│ │   ana@email.com                  ││
│ │   [Ver] [❌ Excluir]               ││
│ └─────────────────────────────────┘ │
│                                      │
│ 📊 Estatísticas                      │
│ ┌─────────────────────────────────┐ │
│ │ • 234 agendamentos (total)       ││
│ │ • 187 agendamentos (últimos 30d)││
│ │ • 12 clientes cadastrados        ││
│ │ • 5 serviços ativos              ││
│ │ • Taxa no-show: 8%               ││
│ │ • WhatsApp conectado: ✅          ││
│ └─────────────────────────────────┘ │
│                                      │
│ 🛠️ Ações Administrativas             │
│ ┌─────────────────────────────────┐ │
│ │ [🔐 Logar como este usuário]   ││
│ │ [✏️ Editar dados]                 ││
│ │ [📝 Adicionar observação]         ││
│ │ [⬆️ Upgrade de plano]             ││
│ │ [⬇️ Downgrade de plano]           ││
│ │ [⏸️ Suspender temporariamente]    ││
│ │ [❌ Cancelar assinatura]          ││
│ │ [🔄 Resetar senha]                ││
│ └─────────────────────────────────┘ │
└──────────────────────────────────────┘
```

---

### Tela SA-5: **Logar Como Cliente** (Impersonation)
**URL:** Ação que redireciona para `/dashboard`

**Fluxo:**
```
1. Você clica em "Logar Como" no painel admin

2. Sistema registra no log:
   - Quem: seu email admin
   - Quando: timestamp
   - Cliente: qual conta acessou
~~~

~~~
   - IP: seu endereço IP

3. Você é redirecionado para o dashboard DO CLIENTE
   - Vê exatamente o que ele vê
   - Pode fazer tudo que ele pode

4. Banner de alerta aparece no topo:
   ┌──────────────────────────────────────┐
   │ ⚠️ MODO ADMIN: Você está logado       │
   │ como "João Silva"                    │
   │ [🚪 Voltar ao Admin]                  │
   └──────────────────────────────────────┘

5. Pode:
   - Ver agenda dele
   - Acessar funcionários dele
   - Excluir funcionários se necessário
   - Testar funcionalidades
   - Resolver problemas

6. Ao clicar "Voltar ao Admin":
   - Sistema registra saída no log
   - Você volta para seu painel admin
```

---

### Tela SA-6: **Gerenciar Funcionários do Cliente**
**URL:** `/admin/clientes/{id}/funcionarios`

```
┌─────────────────────────────────────┐
│ [← Voltar] Funcionários             │
│ Barbearia do João                   │
├─────────────────────────────────────┤
│                                     │
│ Plano atual: PRO (2/2 funcionários) │
│                                     │
│ ┌──────────────────────────────────┐│
│ │ 👤 Carlos Silva                   ││
│ │ carlos@email.com                 ││
│ │ 📱 (11) 98888-8888                ││
│ │                                  ││
│ │ Tipo: Funcionário                ││
│ │ Status: 🟢 Ativo                  ││
│ │ Cadastrado: 20/08/2025           ││
│ │ Último acesso: Há 3 horas        ││
│ │ Agendamentos: 89 (total)         ││
│ │                                  ││
│ │ Permissões:                      ││
│ │ ✅ Ver agenda                     ││
│ │ ✅ Criar agendamentos             ││
│ │ ❌ Ver financeiro                 ││
│ │                                  ││
│ │ [✏️ Editar] [❌ Excluir]           ││
│ │ [⏸️ Desativar] [🔄 Resetar Senha]  ││
│ └──────────────────────────────────┘│
│                                     │
│ ┌──────────────────────────────────┐│
│ │ 👤 Ana Costa                      ││
│ │ ana@email.com                    ││
│ │ 📱 (11) 97777-7777                ││
│ │                                  ││
│ │ Tipo: Admin                      ││
│ │ Status: 🟢 Ativo                  ││
│ │ Cadastrado: 15/09/2025           ││
│ │ Último acesso: Há 1 dia          ││
│ │ Agendamentos: 45 (total)         ││
│ │                                  ││
│ │ Permissões:                      ││
│ │ ✅ Ver agenda                     ││
│ │ ✅ Criar agendamentos             ││
~~~

~~~
│ │ ✅ Ver financeiro            ││
│ │ ✅ Gerenciar serviços        ││
│ │                              ││
│ │ [✏️ Editar] [❌ Excluir]       ││
│ │ [⏸️ Desativar] [🔄 Resetar Senha]││
│ └──────────────────────────────┘│
│                                  │
│ ⚠️ Limite atingido (2/2)          │
│ Para adicionar mais funcionários, │
│ upgrade para TEAM necessário.     │
│                                   │
│ [⬆️ Fazer Upgrade para TEAM]       │
└───────────────────────────────────┘
```

**Ao clicar em "Excluir Funcionário":**
```
┌───────────────────────────────────┐
│ ⚠️ Confirmar Exclusão              │
├───────────────────────────────────┤
│                                   │
│ Você está prestes a EXCLUIR:      │
│                                   │
│ 👤 Carlos Silva                    │
│ carlos@email.com                  │
│                                   │
│ ⚠️ ATENÇÃO:                        │
│ • 89 agendamentos vinculados a ele │
│ • Agendamentos futuros (12) serão │
│   mantidos mas sem responsável    │
│ • Histórico será preservado       │
│ • Ação irreversível               │
│                                   │
│ Motivo (opcional):                │
│ ┌─────────────────────────────┐   │
│ │ Ex: Saiu da empresa         │   │
│ └─────────────────────────────┘   │
│                                   │
│ [Cancelar] [❌ Confirmar Exclusão] │
└───────────────────────────────────┘
```

---

### Tela SA-7: **Alterar Plano do Cliente**
**URL:** Modal em `/admin/clientes/{id}`

```
┌───────────────────────────────────┐
│ Alterar Plano - Barbearia do João │
├───────────────────────────────────┤
│                                   │
│ Plano Atual: PRO (R$ 99,90/mês)   │
│                                   │
│ Escolha o novo plano:             │
│                                   │
│ ( ) FREE                          │
│     R$ 0 • 30 agends/mês • 0 func │
│                                   │
│ ( ) BASIC                         │
│     R$ 59,90 • Ilimitado • 0 func │
│                                   │
│ (•) PRO (atual)                   │
│     R$ 99,90 • Ilimitado • 2 func │
│                                   │
│ ( ) TEAM                          │
│     R$ 179,90 • Ilimitado • 5 func │
│                                   │
│ ( ) ENTERPRISE                    │
│     R$ 299,90 • Ilimitado • ∞ func │
│                                   │
│ ⚠️ Atenção ao mudar:               │
~~~

~~~
│ • Downgrade: funcionalidades podem │
│   ser bloqueadas imediatamente     │
│ • Upgrade: cobra diferença propor- │
│   cional no próximo venc.          │
│                                     │
│ Motivo da alteração:               │
│ ┌──────────────────────────────┐  │
│ │ Cliente solicitou upgrade    │  │
│ └──────────────────────────────┘  │
│                                     │
│ [Cancelar] [Confirmar Alteração]  │
└─────────────────────────────────────┘
```

---

### Tela SA-8: **Logs de Auditoria**
**URL:** `/admin/logs`

```
┌─────────────────────────────────────┐
│ [☰] Logs de Auditoria               │
├─────────────────────────────────────┤
│                                     │
│ Filtros:                           │
│ [Todas Ações] [Últimos 7 dias]     │
│                                     │
│ Buscar: [____________] [🔍]         │
│                                     │
│ ┌─────────────────────────────────┐│
│ │ 🔐 LOGIN COMO USUÁRIO            ││
│ │ 25/12/2025 14:32                ││
│ │                                 ││
│ │ Admin: voce@admin.com           ││
│ │ Cliente: João Silva             ││
│ │ (Barbearia do João)             ││
│ │ IP: 192.168.1.100               ││
│ │ Duração: 8 minutos              ││
│ │                                 ││
│ │ [Ver Detalhes]                  ││
│ └─────────────────────────────────┘│
│                                     │
│ ┌─────────────────────────────────┐│
│ │ ❌ EXCLUIR FUNCIONÁRIO           ││
│ │ 25/12/2025 11:15                ││
│ │                                 ││
│ │ Admin: voce@admin.com           ││
│ │ Cliente: Maria Santos           ││
│ │ Funcionário: Pedro Costa        ││
│ │ Motivo: "Saiu da empresa"       ││
│ │ IP: 192.168.1.100               ││
│ │                                 ││
│ │ [Ver Detalhes]                  ││
│ └─────────────────────────────────┘│
│                                     │
│ ┌─────────────────────────────────┐│
│ │ ⬆️ ALTERAR PLANO                ││
│ │ 24/12/2025 16:45                ││
│ │                                 ││
│ │ Admin: voce@admin.com           ││
│ │ Cliente: Carlos Mendes          ││
│ │ De: BASIC → Para: PRO           ││
│ │ Motivo: "Cliente solicitou"     ││
│ │ IP: 192.168.1.100               ││
│ │                                 ││
│ │ [Ver Detalhes]                  ││
│ └─────────────────────────────────┘│
│                                     │
│ Mostrando 3 de 127 registros       │
│ [← 1 2 3 4 5 →]                   │
└─────────────────────────────────────┘
```
~~~

~~~
---

### Tela SA-9: **Configurações do Sistema**
**URL:** `/admin/configuracoes`

```
┌────────────────────────────────────┐
│ [☰] Configurações do Sistema       │
├────────────────────────────────────┤
│                                    │
│ 🎨 Planos e Limites                 │
│ ┌─────────────────────────────────┐│
│ │ FREE:                           ││
│ │ Agendamentos/mês: [30]          ││
│ │ Funcionários: [0]               ││
│ │                                 ││
│ │ BASIC: R$ [59.90]               ││
│ │ Agendamentos/mês: [Ilimitado]   ││
│ │ Funcionários: [0]               ││
│ │                                 ││
│ │ PRO: R$ [99.90]                 ││
│ │ Agendamentos/mês: [Ilimitado]   ││
│ │ Funcionários: [2]               ││
│ │                                 ││
│ │ TEAM: R$ [179.90]               ││
│ │ Funcionários: [5]               ││
│ │                                 ││
│ │ ENTERPRISE: R$ [299.90]         ││
│ │ Funcionários: [Ilimitado]       ││
│ │                                 ││
│ │ [Salvar Alterações]             ││
│ └─────────────────────────────────┘│
│                                    │
│ 💳 Gateways de Pagamento            │
│ ┌─────────────────────────────────┐│
│ │ [ ] Stripe                      ││
│ │     API Key: [___________]      ││
│ │                                 ││
│ │ [✓] Mercado Pago                ││
│ │     Access Token: [•••••••••]   ││
│ │     Status: 🟢 Conectado         ││
│ │                                 ││
│ │ [Salvar]                        ││
│ └─────────────────────────────────┘│
│                                    │
│ 📧 Email/Notificações               │
│ ┌─────────────────────────────────┐│
│ │ Email de suporte:               ││
│ │ [suporte@smagenda.com]          ││
│ │                                 ││
│ │ [✓] Notificar novos cadastros   ││
│ │ [✓] Alertas de inadimplência    ││
│ │ [✓] Resumo diário (9h)          ││
│ │                                 ││
│ │ [Salvar]                        ││
│ └─────────────────────────────────┘│
└────────────────────────────────────┘
```

---

## 📱 LADO DO PROFISSIONAL (Dashboard)

#### Tela 1: **Login/Cadastro**
**URL:** `/login` e `/cadastro`

**Campos de Cadastro:**
- Nome completo
- Nome do negócio
- Telefone (com WhatsApp)
- Email
~~~

~~~
- Senha
- Slug personalizado (ex: "joao-barbeiro") - auto-gerado mas editável

**Layout:**
- Logo centralizado
- Formulário limpo
- Botão "Criar conta grátis"
- Link "Já tenho conta"

---

#### Tela 2: **Onboarding (Pós-cadastro)**
**URL:** `/onboarding`

**Etapa 1 - Horário de Funcionamento:**
- Seletor de horário início (ex: 08:00)
- Seletor de horário fim (ex: 18:00)
- Checkbox dos dias da semana
- Intervalo (opcional): início e fim

**Etapa 2 - Primeiro Serviço:**
- Nome do serviço
- Duração (em minutos)
- Preço
- Botão "Adicionar mais serviços depois"

**Etapa 3 - Configurar WhatsApp:**
- Opções:
  - [ ] "Enviar manualmente por enquanto" (gera link do WhatsApp)
  - [ ] "Configurar Evolution API agora" (mostra tutorial)
- Link do tutorial: como instalar Evolution API no Railway (gratuito)

**Etapa 4 - Pronto!:**
- Mostra o link de agendamento: `agendafacil.com/joao-barbeiro`
- Botão "Copiar link"
- Botão "Compartilhar no WhatsApp"
- Botão "Ir para minha agenda"

---

#### Tela 3: **Dashboard Principal (Agenda)**
**URL:** `/dashboard`

**Layout:**
```
┌─────────────────────────────────────┐
│ [☰ Menu]    AgendaFácil    [🔔][👤] │
├─────────────────────────────────────┤
│                                     │
│  [< Hoje - 25 Dez >]    [+ Novo]   │
│                                     │
│  ┌───────────────────────────────┐ │
│  │ 08:00 - LIVRE                 │ │
│  ├───────────────────────────────┤ │
│  │ 09:00 - João Silva            │ │
│  │ 📱 (11) 99999-9999           │ │
│  │ ✂️ Corte Masculino - R$ 50    │ │
│  │ [✓ Confirmar] [✗ Cancelar]    │ │
│  ├───────────────────────────────┤ │
│  │ 10:00 - LIVRE                 │ │
│  ├───────────────────────────────┤ │
│  │ 11:00 - Maria Santos          │ │
│  │ 📱 (11) 98888-8888           │ │
│  │ 💇 Escova - R$ 40             │ │
│  │ ✅ Confirmado                 │ │
│  ├───────────────────────────────┤ │
│  │ 12:00 - 🔒 BLOQUEADO (Almoço)│ │
│  └───────────────────────────────┘ │
│                                     │
│  Resumo do Dia:                    │
│  💰 R$ 250,00 • 5 agendamentos     │
└─────────────────────────────────────┘
~~~

~~~
```

**Funcionalidades:**
- Navegação entre dias (setas < >)
- Visualização semanal (toggle)
- Cards de agendamento clicáveis
- Status visual com cores
- Botão flutuante "+ Novo Agendamento"
- Filtro rápido: "Todos", "Confirmados", "Pendentes"

---

#### Tela 4: **Menu Lateral**
**URL:** Slide-in menu

**Itens:**
- 📅 Agenda (tela principal)
- ✂️ Meus Serviços
- 👥 Clientes
- 👨‍💼 Funcionários (só para Master/Admin) ⭐ NOVO
- 🔗 Meu Link de Agendamento
- ⚙️ Configurações
- 💬 Mensagens Automáticas
- 📊 Relatórios (futuro)
- 🎨 Personalizar Página
- ❓ Ajuda
- 🚪 Sair

---

#### Tela 5: **Meus Serviços**
**URL:** `/servicos`

**Layout:**
- Lista de cards de serviços
- Cada card mostra:
  - Nome do serviço
  - Duração e preço
  - Toggle ativo/inativo
  - Ícone de editar e deletar
- Botão "+ Adicionar Serviço"
- Drag-and-drop para reordenar (mobile friendly)

**Modal de Adicionar/Editar:**
- Nome
- Descrição (opcional)
- Duração (em minutos)
- Preço
- Cor (seletor visual)
- Toggle "Ativo"

---

#### Tela 6: **Configurações de WhatsApp**
**URL:** `/configuracoes/whatsapp`

**Seção 1 - Status da Conexão:**
```
┌─────────────────────────────────────┐
│ Status: 🟢 Conectado                │
│ Instância: minha-barbearia          │
│ Número: +55 11 99999-9999           │
│ [Desconectar] [Testar Envio]        │
└─────────────────────────────────────┘
```

**Seção 2 - Configurar Evolution API:**
- Campo: URL da API (ex: `https://sua-instancia.railway.app`)
- Campo: API Key
- Botão "Conectar"
- Link: "Não tem Evolution API? Veja como criar grátis"
~~~

~~~
**Seção 3 - Preferências de Envio:**
- Toggle: "Enviar confirmação ao agendar"
- Toggle: "Enviar lembrete automático"
- Slider: "Enviar lembrete X horas antes" (4h a 72h)

---

#### Tela 7: **Mensagens Automáticas**
**URL:** `/configuracoes/mensagens`

**Mensagem de Confirmação:**
```
┌─────────────────────────────────────┐
│ Olá {nome}! 👋                       │
│                                     │
│ Seu agendamento foi confirmado:    │
│ 📅 {data} às {hora}                  │
│ ✂️ {servico}                         │
│ 💰 {preco}                           │
│                                     │
│ Local: {endereco}                   │
│                                     │
│ Nos vemos em breve!                 │
│ {nome_negocio}                      │
│                                     │
│ [Editar Mensagem]                   │
└─────────────────────────────────────┘
```

**Mensagem de Lembrete:**
```
┌─────────────────────────────────────┐
│ Oi {nome}! 🔔                        │
│                                     │
│ Lembrete: você tem agendamento     │
│ AMANHÃ às {hora}                    │
│                                     │
│ Se não puder comparecer, me avise! │
│ {telefone_profissional}             │
│                                     │
│ [Editar Mensagem]                   │
└─────────────────────────────────────┘
```

**Variáveis disponíveis:**
- {nome} - nome do cliente
- {data} - data do agendamento
- {hora} - horário
- {servico} - nome do serviço
- {preco} - valor
- {endereco} - endereço do profissional
- {nome_negocio} - nome do negócio

---

### 👤 LADO DO CLIENTE (Página de Agendamento)

#### Tela 8: **Página Pública de Agendamento**
**URL:** `/agendar/{slug}` ex: `/agendar/joao-barbeiro`

**Layout:**
```
┌─────────────────────────────────────┐
│  [Foto]                             │
│  Barbearia do João                  │
│  ⭐⭐⭐⭐⭐ (23 avaliações)              │
│  📍 Rua das Flores, 123              │
│                                     │
├─────────────────────────────────────┤
│ Escolha o serviço:                  │
│                                     │
│ [✂️ Corte Masculino                  │
~~~

~~~~
│  ⏱️ 45 min • 💰 R$ 50,00 ]          │
│                                     │
│ [💇‍♀️ Barba                          │
│  ⏱️ 30 min • 💰 R$ 35,00 ]          │
│                                     │
│ [🎨 Corte + Barba (COMBO)           │
│  ⏱️ 1h 15min • 💰 R$ 75,00 ]        │
│                                     │
└─────────────────────────────────────┘
```

**Após selecionar serviço → Tela de Data:**
```
┌─────────────────────────────────────┐
│  [← Voltar]                        │
│                                     │
│  Escolha a data:                   │
│                                     │
│  [ Dezembro 2025 ]                 │
│                                     │
│  D  S  T  Q  Q  S  S              │
│     1  2  3  4  5  6              │
│  7  8  9 10 11 12 13              │
│ 14 15 16 17 18 19 20              │
│ 21 22 23 24[25]26 27              │
│ 28 29 30 31                        │
│                                     │
│  Dias com ❌ estão indisponíveis   │
└─────────────────────────────────────┘
```

**Após selecionar data → Tela de Horário:**
```
┌─────────────────────────────────────┐
│  [← Voltar]                        │
│                                     │
│  Horários disponíveis:             │
│  25 de Dezembro                    │
│                                     │
│  Manhã:                            │
│  [08:00] [09:00] [10:00] [11:00]  │
│                                     │
│  Tarde:                            │
│  [14:00] [15:00] [16:00] [17:00]  │
│                                     │
│  (12:00 e 13:00 indisponíveis)    │
└─────────────────────────────────────┘
```

**Após selecionar horário → Tela de Dados:**
```
┌─────────────────────────────────────┐
│  [← Voltar]                        │
│                                     │
│  Resumo do Agendamento:            │
│  ✂️ Corte Masculino                  │
│  📅 25/12/2025 às 09:00             │
│  💰 R$ 50,00                         │
│                                     │
│  Seus dados:                       │
│  ┌─────────────────────────────┐  │
│  │ Nome completo               │  │
│  └─────────────────────────────┘  │
│                                     │
│  ┌─────────────────────────────┐  │
│  │ (11) 9____-____             │  │
│  └─────────────────────────────┘  │
│                                     │
│  [ ] Já sou cliente               │
│                                     │
│  [Confirmar Agendamento]          │
│                                     │
~~~~

~~~~
│  Você receberá uma confirmação    │
│  no WhatsApp 📱                   │
└───────────────────────────────────┘
```

**Tela de Sucesso:**
```
┌───────────────────────────────────┐
│         ✅                         │
│                                   │
│  Agendamento Confirmado!          │
│                                   │
│  João Silva, seu horário está     │
│  garantido!                       │
│                                   │
│  📅 25 de Dezembro às 09:00       │
│  ✂️ Corte Masculino                │
│  📍 Rua das Flores, 123           │
│                                   │
│  Você receberá:                   │
│  • Confirmação agora no WhatsApp  │
│  • Lembrete 24h antes             │
│                                   │
│  [Adicionar ao Calendário]        │
│  [Voltar ao Início]               │
│                                   │
│  Precisa cancelar?                │
│  Fale com João: (11) 99999-9999   │
└───────────────────────────────────┘
```

---

## 🤖 Integração com Evolution API (WhatsApp)

### Como Funciona a Automação

#### 1. **Setup da Evolution API (Gratuito)**

**Opções de Hospedagem:**
- **Railway** (recomendado): 500h gratuitas/mês
- **Render**: 750h gratuitas/mês
- **VPS própria**: $5-10/mês (alternativa)

**Instalação no Railway:**
```bash
1. Criar conta no Railway (railway.app)
2. Usar template da Evolution API:
   - GitHub: https://github.com/EvolutionAPI/evolution-api
3. Clicar em "Deploy"
4. Configurar variáveis de ambiente:
   - AUTHENTICATION_API_KEY=[REDACTED]
   - DATABASE_PROVIDER=postgresql
5. Deploy automático em ~3 minutos
6. Copiar URL: https://seu-app.railway.app
```

#### 2. **Conectar Instância no Sistema**

**Fluxo no Dashboard:**
```
Profissional → Configurações → WhatsApp
↓
Informar URL da Evolution API + API Key
↓
Sistema testa conexão (endpoint /instance/connect)
↓
Gera QR Code para conectar WhatsApp
↓
Profissional escaneia com WhatsApp
↓
✅ Conectado!
~~~~

~~~
```

#### 3. **Endpoints Utilizados**

**a) Criar/Conectar Instância:**
```javascript
POST /instance/create
{
  "instanceName": "barbearia-joao",
  "token": "sua-chave-api",
  "qrcode": true
}

// Retorna QR Code para conectar
```

**b) Enviar Mensagem de Confirmação:**
```javascript
POST /message/sendText/{instanceName}
{
  "number": "5511999999999",
  "text": "Olá João Silva! 👋\n\nSeu agendamento foi confirmado:\n📅 25/12/2025 às 09:00\n✂️ Corte Masculino\n💰 R$
50,00\n\nNos vemos em breve!\nBarbearia do João"
}
```

**c) Agendar Lembrete (usando cron job):**
```javascript
// No backend, criar job agendado:
// Verifica a cada 1 hora se há agendamentos nas próximas 24h

SELECT * FROM agendamentos
WHERE data = CURRENT_DATE + 1
  AND lembrete_enviado = false
  AND status = 'confirmado'

// Para cada agendamento encontrado:
POST /message/sendText/{instanceName}
{
  "number": "5511999999999",
  "text": "Oi João Silva! 🔔\n\nLembrete: você tem agendamento AMANHÃ às 09:00\n\nSe não puder comparecer, me
avise!\n(11) 98888-8888"
}

// Após enviar, marcar lembrete_enviado = true
```

#### 4. **Fallback Manual (Caso Evolution API não esteja configurada)**

**Quando profissional não tem WhatsApp automatizado:**

```javascript
// Sistema gera link do WhatsApp Web com mensagem pronta

function gerarLinkWhatsApp(agendamento) {
  const numero = agendamento.cliente_telefone.replace(/\D/g, '');
  const mensagem = encodeURIComponent(
    `Olá ${agendamento.cliente_nome}! 👋\n\n` +
    `Seu agendamento foi confirmado:\n` +
    `📅 ${agendamento.data} às ${agendamento.hora_inicio}\n` +
    `✂️ ${agendamento.servico.nome}\n` +
    `💰 R$ ${agendamento.servico.preco}\n\n` +
    `Nos vemos em breve!\n` +
    `${usuario.nome_negocio}`
  );

  return `https://wa.me/55${numero}?text=${mensagem}`;
}

// No dashboard, mostrar:
// "Clique para enviar confirmação: [Abrir WhatsApp]"
```
~~~

```
---

## ⚙️ Funcionalidades Avançadas (Diferenciais)

### 1. **Sistema de Notificações Inteligentes**
- Envio automático de confirmação (imediato)
- Lembrete 24h antes (padrão, configurável)
- Lembrete 2h antes (opcional)
- Mensagem de agradecimento pós-atendimento (opcional)

### 2. **Gestão de No-Shows**
- Marcar cliente como "não compareceu"
- Sistema sugere enviar mensagem automática perguntando motivo
- Histórico de no-shows por cliente
- Opção de pedir confirmação prévia para clientes com histórico

### 3. **Bloqueios e Disponibilidade**
- Bloquear horários pontuais (almoço, compromissos)
- Bloquear dias inteiros (férias, feriados)
- Bloqueios recorrentes (ex: toda segunda 12h-13h)

### 4. **Clientes Recorrentes**
- Sistema reconhece telefone do cliente
- Autopreenchimento de dados
- Histórico de agendamentos
- Sugestão de "mesmo horário de sempre"

### 5. **Personalização da Página**
- Upload de foto/logo
- Cores personalizadas (tema)
- Descrição do negócio
- Link para Instagram/redes sociais
- Galeria de fotos (trabalhos realizados)

### 6. **Métricas Básicas**
- Taxa de no-show (%)
- Receita prevista vs realizada
- Serviços mais agendados
- Horários de pico
- Clientes novos vs recorrentes

### 8. **Gestão de Equipe (Multi-usuário)** ⭐ ESSENCIAL
- Sistema de permissões granulares (Owner/Admin/Funcionário)
- Funcionários veem apenas seus próprios agendamentos
- Dono vê tudo e pode filtrar por funcionário
- Controle de visibilidade de valores financeiros
- Relatórios segmentados por profissional
- Cliente escolhe o profissional ao agendar (ou "qualquer um disponível")

### 9. **Integração com Google Calendar** (Futuro)
- Sincronização bidirecional
- Bloquear horários marcados no Google automaticamente

---

## 🚀 Roadmap de Desenvolvimento

### FASE 1 - MVP (2-3 semanas)
**Objetivo:** Sistema funcional para validar com primeiros clientes

- [x] Setup do projeto (React + Vite + Supabase)
- [x] Sistema de autenticação (login/cadastro)
- [x] CRUD de serviços
- [x] Lógica de cálculo de horários disponíveis
- [x] Página pública de agendamento
- [x] Dashboard com visualização de agenda (dia)
- [ ] Sistema de bloqueios simples
- [x] Geração de link de WhatsApp manual (sem Evolution API)
- [ ] Deploy na Vercel

**Entrega:** Sistema usável onde profissional pode receber agendamentos e enviar confirmações manualmente via WhatsApp.
```

```markdown
---

### FASE 2 - Automação (1-2 semanas)
**Objetivo:** Reduzir trabalho manual do profissional

- [ ] Integração com Evolution API
- [ ] QR Code para conectar WhatsApp
- [ ] Envio automático de confirmação
- [ ] Cron job para lembretes automáticos
- [ ] Configuração de mensagens personalizadas
- [ ] Teste de envio de mensagens

**Entrega:** Sistema 100% automatizado para mensagens.

---

### FASE 3 - Experiência do Usuário (1 semana)
**Objetivo:** Melhorar usabilidade e conversão

- [x] Onboarding completo pós-cadastro
- [ ] Tutorial interativo no dashboard
- [ ] Visualização semanal da agenda
- [ ] Filtros e busca na agenda
- [ ] Status visual de agendamentos (cores)
- [ ] Página de agendamento com foto/logo
- [ ] Responsividade total (mobile-first)

**Entrega:** Sistema polido e fácil de usar.

---

### FASE 4 - Gestão Avançada (1-2 semanas)
**Objetivo:** Dar mais controle ao profissional

- [ ] Gestão de clientes recorrentes
- [ ] Histórico de agendamentos por cliente
- [ ] Sistema de no-shows
- [ ] Relatórios básicos (dashboard de métricas)
- [ ] Bloqueios recorrentes
- [ ] Exportação de agenda (CSV)

**Entrega:** Sistema completo de gestão.

---

### FASE 5 - Painel Super Admin (CRÍTICO) ⭐⭐
**Objetivo:** Você conseguir gerenciar TODOS os clientes

- [x] Sistema de autenticação super admin
- [x] Dashboard com visão geral (MRR, clientes ativos, etc)
- [ ] Lista de todos os clientes (com filtros)
- [x] Detalhes completos de cada cliente
- [ ] **"Logar como cliente" (impersonation)**
- [ ] Gerenciar funcionários dos clientes
- [ ] Alterar planos manualmente
- [ ] Suspender/reativar contas
- [ ] Logs de auditoria (tudo que você faz é registrado)
- [ ] Configurações de planos e preços
- [ ] Integração com gateway de pagamento

**Entrega:** Você com controle total do sistema.

---

### FASE 6 - Sistema Multi-usuário (ALTA PRIORIDADE)
**Objetivo:** Permitir que donos de negócio gerenciem funcionários

- [x] Sistema de permissões (Owner/Admin/Funcionário)
- [x] CRUD de funcionários
- [x] Dashboard do funcionário (visão limitada)
- [x] Agendamentos por funcionário
```

```markdown
- [ ] Relatórios segmentados por funcionário
- [ ] Controle de acesso granular
- [x] **Validação de limites por plano**

**Entrega:** Sistema completo para salões/barbearias com equipe.

---

### FASE 7 - Monetização (1 semana)
**Objetivo:** Implementar planos pagos

- [ ] Integração com gateway de pagamento (Stripe/Mercado Pago)
- [ ] Sistema de planos (Free, Basic, Pro)
- [ ] Limites por plano (ex: free = 30 agendamentos/mês)
- [ ] Página de upgrade
- [x] Painel administrativo (para você gerenciar clientes)

**Entrega:** Sistema pronto para gerar receita recorrente.

---

### FASE 6 - Sistema Multi-usuário (ALTA PRIORIDADE)
**Objetivo:** Permitir que donos de negócio gerenciem funcionários

- [x] Sistema de permissões (Owner/Admin/Funcionário)
- [x] CRUD de funcionários
- [x] Dashboard do funcionário (visão limitada)
- [x] Agendamentos por funcionário
- [ ] Relatórios segmentados por funcionário
- [ ] Controle de acesso granular

**Entrega:** Sistema completo para salões/barbearias com equipe.

---

### BACKLOG (Futuro)
- [ ] App mobile nativo (quando tiver CNPJ)
- [ ] Integração com Google Calendar
- [ ] Sistema de avaliações/reviews
- [ ] Pagamento online na hora de agendar
- [ ] Programa de fidelidade (pontos)
- [ ] WhatsApp chatbot para agendar por conversa

---

## 💰 Estratégia de Monetização

### Planos Sugeridos

#### 🆓 Plano FREE
**R$ 0/mês - Para testar**
- Até 30 agendamentos por mês
- 1 profissional
- Lembretes manuais (link do WhatsApp)
- Suporte por email

#### ⭐ Plano BASIC
**R$ 59,90/mês**
- Agendamentos ilimitados
- 1 profissional
- **Lembretes automáticos via WhatsApp**
- Até 3 serviços
- Personalização básica da página
- Suporte prioritário

#### 🚀 Plano PRO
**R$ 99,90/mês**
- Tudo do Basic +
- Serviços ilimitados
- Gestão de clientes (histórico completo)
- Relatórios avançados
- Bloqueios recorrentes
```

```
- **Até 2 funcionários** ⭐
- Logo e galeria de fotos
- Suporte via WhatsApp

#### 💼 Plano TEAM ⭐ NOVO
**R$ 179,90/mês**
- Tudo do Pro +
- **Até 5 funcionários**
- Agenda unificada (filtro por profissional)
- Relatórios por funcionário
- Controle de permissões detalhado
- Cliente escolhe o profissional
- Suporte prioritário

#### 🏢 Plano ENTERPRISE ⭐ NOVO
**R$ 299,90/mês**
- Tudo do Team +
- **Funcionários ilimitados**
- Multi-unidades (filiais)
- API de integração
- Suporte dedicado via WhatsApp
- Treinamento da equipe incluído

---

### Serviços Adicionais

**Setup Completo:** R$ 150 (uma vez)
- Você configura tudo para o cliente
- Cadastra serviços, fotos, horários
- Conecta WhatsApp
- Testa envios
- Treina o cliente em 15 minutos

**Consultoria por Hora:** R$ 80/hora
- Ajuda com configurações avançadas
- Sugestões de otimização
- Dúvidas gerais

---

## 📊 Métricas de Sucesso

### Para Validação do MVP
- ✅ 5 profissionais usando ativamente
- ✅ 100+ agendamentos realizados
- ✅ Taxa de no-show reduzida em pelo menos 30%
- ✅ NPS (Net Promoter Score) acima de 8

### Para Crescimento
- 50 profissionais pagantes em 6 meses
- MRR de R$ 3.000 em 6 meses
- Taxa de churn abaixo de 10%/mês
- Tempo médio de setup: menos de 10 minutos

---

## 🎯 Público-Alvo Detalhado

### Persona 1: **Barbeiro Autônomo**
- **Idade:** 25-40 anos
- **Dor:** Clientes marcam e não aparecem (perde R$ 200-500/mês)
- **Comportamento:** Usa WhatsApp para tudo, não gosta de sistemas complicados
- **Onde encontrar:** Instagram, grupos de Facebook, indicação

### Persona 2: **Manicure Home-Care**
- **Idade:** 28-45 anos
- **Dor:** Desorganização da agenda, esquecem horários, perdem tempo no telefone
- **Comportamento:** Atende em domicílio, precisa de mobilidade
- **Onde encontrar:** Instagram, WhatsApp Status, boca a boca

### Persona 3: **Personal Trainer**
```

```
- **Idade:** 25-38 anos
- **Dor:** Alunos cancelam em cima da hora, dificulta reposição
- **Comportamento:** Tech-savvy, gosta de otimizar tempo
- **Onde encontrar:** Instagram fitness, grupos de crossfit/academias

---

## 🔧 Código Base - Estrutura de
```

smagenda/docker-compose.evolution.yml

```
version: '3.9'

services:
  evolution-api:
    container_name: evolution_api
    image: evoapicloud/evolution-api:v2.3.7
    restart: always
    ports:
      - '8080:8080'
    environment:
      AUTHENTICATION_TYPE: apikey
      AUTHENTICATION_API_KEY=[REDACTED]
      CORS_ORIGIN: ${EVOLUTION_CORS_ORIGIN}
      CONFIG_SESSION_PHONE_VERSION: ${EVOLUTION_CONFIG_SESSION_PHONE_VERSION:-2.3000.1030444778}
      DATABASE_ENABLED: ${EVOLUTION_DATABASE_ENABLED}
      DATABASE_PROVIDER: ${EVOLUTION_DATABASE_PROVIDER}
      DATABASE_CONNECTION_URI: ${EVOLUTION_DATABASE_CONNECTION_URI}
      CACHE_REDIS_ENABLED: ${EVOLUTION_CACHE_REDIS_ENABLED}
      CACHE_REDIS_URI: ${EVOLUTION_CACHE_REDIS_URI}
      CACHE_REDIS_PREFIX_KEY: ${EVOLUTION_CACHE_REDIS_PREFIX_KEY}
    volumes:
      - evolution_instances:/evolution/instances
    depends_on:
      - evolution_db
      - evolution_redis

  evolution_db:
    container_name: evolution_db
    image: postgres:16-alpine
    restart: always
    environment:
      POSTGRES_DB: ${EVOLUTION_POSTGRES_DB}
      POSTGRES_USER: ${EVOLUTION_POSTGRES_USER}
      POSTGRES_PASSWORD: ${EVOLUTION_POSTGRES_PASSWORD}
    volumes:
      - evolution_db_data:/var/lib/postgresql/data

  evolution_redis:
    container_name: evolution_redis
    image: redis:7-alpine
    restart: always
    command: ["redis-server", "--appendonly", "yes"]
    volumes:
      - evolution_redis_data:/data

volumes:
  evolution_instances:
  evolution_db_data:
  evolution_redis_data:
```

smagenda/eslint.config.js

```
import js from '@eslint/js'
import globals from 'globals'
import reactHooks from 'eslint-plugin-react-hooks'
import reactRefresh from 'eslint-plugin-react-refresh'
import tseslint from 'typescript-eslint'
import { defineConfig, globalIgnores } from 'eslint/config'

export default defineConfig([
  globalIgnores(['dist']),
  {
    files: ['**/*.{ts,tsx}'],
    extends: [
      js.configs.recommended,
      tseslint.configs.recommended,
      reactHooks.configs.flat.recommended,
      reactRefresh.configs.vite,
    ],
    languageOptions: {
      ecmaVersion: 2020,
      globals: globals.browser,
    },
  },
])
```

smagenda/index.html

```
<!doctype html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8" />
    <meta name="theme-color" content="#0f172a" />
    <link rel="manifest" href="/manifest.webmanifest" />
    <link rel="icon" type="image/png" href="/favicon.png" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>SMagenda</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.tsx"></script>
  </body>
</html>
```

smagenda/package.json

```
{
  "name": "smagenda",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "tsc -b && vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "@supabase/supabase-js": "^2.89.0",
    "react": "^19.2.0",
    "react-dom": "^19.2.0",
    "react-router-dom": "^7.11.0"
  },
  "devDependencies": {
    "@eslint/js": "^9.39.1",
    "@types/node": "^24.10.1",
    "@types/react": "^19.2.5",
    "@types/react-dom": "^19.2.3",
    "@vitejs/plugin-react": "^5.1.1",
    "autoprefixer": "^10.4.23",
    "eslint": "^9.39.1",
    "eslint-plugin-react-hooks": "^7.0.1",
    "eslint-plugin-react-refresh": "^0.4.24",
    "globals": "^16.5.0",
    "postcss": "^8.5.6",
    "supabase": "^2.70.5",
    "tailwindcss": "^3.4.17",
    "typescript": "~5.9.3",
    "typescript-eslint": "^8.46.4",
    "vite": "^7.2.4"
  }
}
```

smagenda/postcss.config.js

```
export default {
  plugins: {
    tailwindcss: {},
    autoprefixer: {},
  },
}
```

smagenda/public/_headers

```
/
  Cache-Control: no-store

/index.html
  Cache-Control: no-store

/assets/*
  Cache-Control: public, max-age=31536000, immutable
```

smagenda/public/_redirects

```
/142555787po/* /142555787po/index.html 200
/* /index.html 200
```

smagenda/public/sw.js

```
const CACHE_NAME = 'smagenda-shell-v1'

self.addEventListener('install', (event) => {
  event.waitUntil(
    caches
      .open(CACHE_NAME)
      .then((cache) => cache.addAll(['/', '/index.html', '/manifest.webmanifest', '/favicon.png']))
      .then(() => self.skipWaiting())
  )
})

self.addEventListener('activate', (event) => {
  event.waitUntil(
    caches
      .keys()
      .then((keys) => Promise.all(keys.filter((k) => k !== CACHE_NAME).map((k) => caches.delete(k))))
      .then(() => self.clients.claim())
  )
})

self.addEventListener('fetch', (event) => {
  const req = event.request
  if (req.method !== 'GET') return

  const url = new URL(req.url)
  if (url.origin !== self.location.origin) return

  if (req.mode === 'navigate') {
    event.respondWith(
      fetch(req)
        .then((res) => {
          const copy = res.clone()
          caches.open(CACHE_NAME).then((cache) => cache.put(req, copy))
          return res
        })
        .catch(() => caches.match('/'))
    )
    return
  }

  event.respondWith(
    caches.match(req).then((cached) =>
      cached
        ? cached
        : fetch(req).then((res) => {
            const copy = res.clone()
            caches.open(CACHE_NAME).then((cache) => cache.put(req, copy))
            return res
          })
    )
  )
})
```

smagenda/public/vite.svg

```
<svg xmlns="http://www.w3.org/2000/svg" width="64" height="64" viewBox="0 0 64 64">
  <defs>
    <radialGradient id="bg" cx="20%" cy="90%" r="90%">
      <stop offset="0%" stop-color="#2f4c86" />
      <stop offset="55%" stop-color="#1f2655" />
      <stop offset="100%" stop-color="#14163a" />
    </radialGradient>
    <linearGradient id="stroke" x1="10%" y1="15%" x2="90%" y2="85%">
      <stop offset="0%" stop-color="#32d4ff" />
      <stop offset="55%" stop-color="#7a4bff" />
      <stop offset="100%" stop-color="#ff4b62" />
    </linearGradient>
    <filter id="soft" x="-30%" y="-30%" width="160%" height="160%">
      <feGaussianBlur stdDeviation="0.75" />
    </filter>
  </defs>

  <rect width="64" height="64" rx="14" fill="url(#bg)" />

  <path
    d="M46 18c-4-4.2-10.1-6.2-16-5.2c-4.6.8-8.5 3.6-9.8 7.1c-1.3 3.5.2 7.2 4 9.6c2.2 1.4 5.3 2.2 8.2 2.9c3.2.8 6.1 1.5 7.7 2.6c2.1 1.5 2.9 3.4 2.2 5.2c-.9 2.4-4.2 4.4-8.3 4.9c-5.2.7-10.8-1.2-14.2-4.9"
    fill="none"
    stroke="url(#stroke)"
    stroke-width="6"
    stroke-linecap="round"
    stroke-linejoin="round"
  />

  <path
    d="M50 17c-4.7-5.4-12.6-7.7-20.1-6.4c-6 1-11 4.8-12.8 9.7c-1.9 5 0.3 10.1 5.3 13.2c2.7 1.7 6.1 2.6 9.2 3.4c3 .7 5.6 1.4 6.8 2.2c.9.7 1.2 1.3 1 2c-.4 1.1-2.3 2.1-4.8 2.4c-4.2.6-8.7-.9-11.7-3.9"
    fill="none"
    stroke="rgba(255,255,255,0.18)"
    stroke-width="3"
    stroke-linecap="round"
    stroke-linejoin="round"
    filter="url(#soft)"
  />

  <circle cx="45" cy="18" r="3" fill="#bdf4ff" opacity="0.9" />
  <circle cx="18" cy="43" r="2.2" fill="#c9d2ff" opacity="0.6" />
</svg>
```

smagenda/README.md

~~~
# React + TypeScript + Vite

This template provides a minimal setup to get React working in Vite with HMR and some ESLint rules.

Currently, two official plugins are available:

- [@vitejs/plugin-react](https://github.com/vitejs/vite-plugin-react/blob/main/packages/plugin-react) uses [Babel](https://babeljs.io/) (or [oxc](https://oxc.rs) when used in [rolldown-vite](https://vite.dev/guide/rolldown)) for Fast Refresh
- [@vitejs/plugin-react-swc](https://github.com/vitejs/vite-plugin-react/blob/main/packages/plugin-react-swc) uses [SWC](https://swc.rs/) for Fast Refresh

## React Compiler

The React Compiler is not enabled on this template because of its impact on dev & build performances. To add it, see [this documentation](https://react.dev/learn/react-compiler/installation).

## Expanding the ESLint configuration

If you are developing a production application, we recommend updating the configuration to enable type-aware lint rules:

```js
export default defineConfig([
  globalIgnores(['dist']),
  {
    files: ['**/*.{ts,tsx}'],
    extends: [
      // Other configs...

      // Remove tseslint.configs.recommended and replace with this
      tseslint.configs.recommendedTypeChecked,
      // Alternatively, use this for stricter rules
      tseslint.configs.strictTypeChecked,
      // Optionally, add this for stylistic rules
      tseslint.configs.stylisticTypeChecked,

      // Other configs...
    ],
    languageOptions: {
      parserOptions: {
        project: ['./tsconfig.node.json', './tsconfig.app.json'],
        tsconfigRootDir: import.meta.dirname,
      },
      // other options...
    },
  },
])
```

You can also install [eslint-plugin-react-x](https://github.com/Rel1cx/eslint-react/tree/main/packages/plugins/eslint-plugin-react-x) and [eslint-plugin-react-dom](https://github.com/Rel1cx/eslint-react/tree/main/packages/plugins/eslint-plugin-react-dom) for React-specific lint rules:

```js
// eslint.config.js
import reactX from 'eslint-plugin-react-x'
import reactDom from 'eslint-plugin-react-dom'

export default defineConfig([
  globalIgnores(['dist']),
  {
    files: ['**/*.{ts,tsx}'],
    extends: [
      // Other configs...
      // Enable lint rules for React
      reactX.configs['recommended-typescript'],
      // Enable lint rules for React DOM
      reactDom.configs.recommended,
    ],
    languageOptions: {
~~~

~~~
      parserOptions: {
        project: ['./tsconfig.node.json', './tsconfig.app.json'],
        tsconfigRootDir: import.meta.dirname,
      },
      // other options...
    },
  },
])
```
~~~

smagenda/src/App.tsx

```
import { Suspense, lazy } from 'react'
import { Navigate, Route, Routes } from 'react-router-dom'
import { RequireAuth } from './state/auth/RequireAuth'

const PublicBookingPage = lazy(() => import('./views/public/PublicBookingPage').then((m) => ({ default:
m.PublicBookingPage })))
const LandingPage = lazy(() => import('./views/public/LandingPage').then((m) => ({ default: m.LandingPage })))
const TermosPage = lazy(() => import('./views/public/TermosPage').then((m) => ({ default: m.TermosPage })))
const PrivacidadePage = lazy(() => import('./views/public/PrivacidadePage').then((m) => ({ default: m.PrivacidadePage
})))
const AjudaPage = lazy(() => import('./views/public/AjudaPage').then((m) => ({ default: m.AjudaPage })))

const LoginPage = lazy(() => import('./views/auth/LoginPage').then((m) => ({ default: m.LoginPage })))
const CadastroPage = lazy(() => import('./views/auth/CadastroPage').then((m) => ({ default: m.CadastroPage })))
const OnboardingPage = lazy(() => import('./views/auth/OnboardingPage').then((m) => ({ default: m.OnboardingPage })))
const ForgotPasswordPage = lazy(() => import('./views/auth/ForgotPasswordPage').then((m) => ({ default:
m.ForgotPasswordPage })))
const ResetPasswordPage = lazy(() => import('./views/auth/ResetPasswordPage').then((m) => ({ default:
m.ResetPasswordPage })))

const DashboardPage = lazy(() => import('./views/app/DashboardPage').then((m) => ({ default: m.DashboardPage })))
const ServicosPage = lazy(() => import('./views/app/ServicosPage').then((m) => ({ default: m.ServicosPage })))
const FuncionariosPage = lazy(() => import('./views/app/FuncionariosPage').then((m) => ({ default: m.FuncionariosPage
})))
const ClientesPage = lazy(() => import('./views/app/ClientesPage').then((m) => ({ default: m.ClientesPage })))
const ClienteDetalhesPage = lazy(() => import('./views/app/ClienteDetalhesPage').then((m) => ({ default:
m.ClienteDetalhesPage })))
const RelatoriosPage = lazy(() => import('./views/app/RelatoriosPage').then((m) => ({ default: m.RelatoriosPage })))
const WhatsappSettingsPage = lazy(() => import('./views/app/WhatsappSettingsPage').then((m) => ({ default:
m.WhatsappSettingsPage })))
const MensagensSettingsPage = lazy(() => import('./views/app/MensagensSettingsPage').then((m) => ({ default:
m.MensagensSettingsPage })))
const PaginaPublicaSettingsPage = lazy(() =>
  import('./views/app/PaginaPublicaSettingsPage').then((m) => ({ default: m.PaginaPublicaSettingsPage }))
)
const FuncionarioAgendaPage = lazy(() => import('./views/app/FuncionarioAgendaPage').then((m) => ({ default:
m.FuncionarioAgendaPage })))
const PagamentoPage = lazy(() => import('./views/app/PagamentoPage').then((m) => ({ default: m.PagamentoPage })))

const AdminLoginPage = lazy(() => import('./views/admin/AdminLoginPage').then((m) => ({ default: m.AdminLoginPage })))
const AdminBootstrapPage = lazy(() => import('./views/admin/AdminBootstrapPage').then((m) => ({ default:
m.AdminBootstrapPage })))
const AdminDashboardPage = lazy(() => import('./views/admin/AdminDashboardPage').then((m) => ({ default:
m.AdminDashboardPage })))
const AdminClientesPage = lazy(() => import('./views/admin/AdminClientesPage').then((m) => ({ default:
m.AdminClientesPage })))
const AdminClienteDetalhesPage = lazy(() =>
  import('./views/admin/AdminClienteDetalhesPage').then((m) => ({ default: m.AdminClienteDetalhesPage }))
)
const AdminLogsPage = lazy(() => import('./views/admin/AdminLogsPage').then((m) => ({ default: m.AdminLogsPage })))
const AdminConfiguracoesPage = lazy(() => import('./views/admin/AdminConfiguracoesPage').then((m) => ({ default:
m.AdminConfiguracoesPage })))
const AdminWhatsappAvisosPage = lazy(() => import('./views/admin/AdminWhatsappAvisosPage').then((m) => ({ default:
m.AdminWhatsappAvisosPage })))

function AppFallback() {
  return (
    <div className="min-h-screen bg-slate-50 flex items-center justify-center px-4">
      <div className="text-sm text-slate-600">Carregando…</div>
    </div>
  )
}

function App() {
  return (
    <Suspense fallback={<AppFallback />}>
      <Routes>
        <Route path="/" element={<LandingPage />} />

        <Route path="/agendar/:slug/:unidadeSlug" element={<PublicBookingPage />} />
```

```
        <Route path="/agendar/:slug" element={<PublicBookingPage />} />

        <Route path="/termos" element={<TermosPage />} />
        <Route path="/privacidade" element={<PrivacidadePage />} />
        <Route path="/ajuda" element={<AjudaPage />} />

        <Route path="/login" element={<LoginPage />} />
        <Route path="/esqueci-senha" element={<ForgotPasswordPage />} />
        <Route path="/resetar-senha" element={<ResetPasswordPage />} />
        <Route path="/cadastro" element={<CadastroPage />} />
        <Route
          path="/onboarding"
          element={
            <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
              <OnboardingPage />
            </RequireAuth>
          }
        />

        <Route
          path="/dashboard"
          element={
            <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
              <DashboardPage />
            </RequireAuth>
          }
        />
        <Route
          path="/servicos"
          element={
            <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
              <ServicosPage />
            </RequireAuth>
          }
        />
        <Route
          path="/clientes"
          element={
            <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
              <ClientesPage />
            </RequireAuth>
          }
        />
        <Route
          path="/clientes/:telefone"
          element={
            <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
              <ClienteDetalhesPage />
            </RequireAuth>
          }
        />
        <Route
          path="/relatorios"
          element={
            <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
              <RelatoriosPage />
            </RequireAuth>
          }
        />
        <Route
          path="/pagamento"
          element={
            <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
              <PagamentoPage />
            </RequireAuth>
          }
        />
        <Route
          path="/funcionarios"
          element={
            <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
              <FuncionariosPage />
```

```
            </RequireAuth>
          }
        />
        <Route
          path="/configuracoes/whatsapp"
          element={
            <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
              <WhatsappSettingsPage />
            </RequireAuth>
          }
        />
        <Route
          path="/configuracoes/mensagens"
          element={
            <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
              <MensagensSettingsPage />
            </RequireAuth>
          }
        />
        <Route
          path="/configuracoes/pagina-publica"
          element={
            <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
              <PaginaPublicaSettingsPage />
            </RequireAuth>
          }
        />
        <Route
          path="/funcionario/agenda"
          element={
            <RequireAuth requiredKind="funcionario">
              <FuncionarioAgendaPage />
            </RequireAuth>
          }
        />

        {import.meta.env.DEV ? <Route path="/admin/bootstrap" element={<AdminBootstrapPage />} /> : null}
        <Route path="/admin/login" element={<AdminLoginPage />} />
        <Route
          path="/admin/dashboard"
          element={
            <RequireAuth requiredKind="super_admin">
              <AdminDashboardPage />
            </RequireAuth>
          }
        />
        <Route
          path="/admin/clientes"
          element={
            <RequireAuth requiredKind="super_admin">
              <AdminClientesPage />
            </RequireAuth>
          }
        />
        <Route
          path="/admin/clientes/:id"
          element={
            <RequireAuth requiredKind="super_admin">
              <AdminClienteDetalhesPage />
            </RequireAuth>
          }
        />
        <Route
          path="/admin/logs"
          element={
            <RequireAuth requiredKind="super_admin">
              <AdminLogsPage />
            </RequireAuth>
          }
        />
        <Route
          path="/admin/whatsapp"
```

```
          element={
            <RequireAuth requiredKind="super_admin">
              <AdminWhatsappAvisosPage />
            </RequireAuth>
          }
        />
        <Route
          path="/admin/configuracoes"
          element={
            <RequireAuth requiredKind="super_admin">
              <AdminConfiguracoesPage />
            </RequireAuth>
          }
        />

        <Route path="*" element={<Navigate to="/" replace />} />
      </Routes>
    </Suspense>
  )
}

export default App
```

smagenda/src/assets/react.svg

```
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" aria-hidden="true" role="img"
class="iconify iconify--logos" width="35.93" height="32" preserveAspectRatio="xMidYMid meet" viewBox="0 0 256 228"><path
fill="#00D8FF" d="M210.483 73.824a171.49 171.49 0 0 0-8.24-2.597c.465-1.9.893-3.777 1.273-5.621c6.238-30.281 2.16-
54.676-11.769-62.708c-13.355-7.7-35.196.329-57.254 19.526a171.23 171.23 0 0 0-6.375 5.848a155.866 155.866 0 0 0-4.241-
3.917C100.759 3.829 77.587-4.822 63.673 3.233C50.33 10.957 46.379 33.89 51.995 62.588a170.974 170.974 0 0 0 1.892 8.48c-
3.28.932-6.445 1.924-9.474 2.98C17.309 83.498 0 98.307 0 113.668c0 15.865 18.582 31.778 46.812 41.427a145.52 145.52 0 0
0 6.921 2.165a167.467 167.467 0 0 0-2.01 9.138c-5.354 28.2-1.173 50.591 12.134 58.266c13.744 7.926 36.812-.22 59.273-
19.855a145.567 145.567 0 0 0 5.342-4.923a168.064 168.064 0 0 0 6.92 6.314c21.758 18.722 43.246 26.282 56.54
18.586c13.731-7.949 18.194-32.003 12.4-61.268a145.016 145.016 0 0 0-1.535-6.842c1.62-.48 3.21-.974 4.76-1.488c29.348-
9.723 48.443-25.443 48.443-41.52c0-15.417-17.868-30.326-45.517-39.844Zm-6.365 70.984c-1.4.463-2.836.91-4.3 1.345c-3.24-
10.257-7.612-21.163-12.963-32.432c5.106-11 9.31-21.767 12.459-31.957c2.619.758 5.16 1.557 7.61 2.4c23.69 8.156 38.14
20.213 38.14 29.504c0 9.896-15.606 22.743-40.946 31.14Zm-10.514 20.834c2.562 12.94 2.927 24.64 1.23 33.787c-1.524 8.219-
4.59 13.698-8.382 15.893c-8.067 4.67-25.32-1.4-43.927-17.412a156.726 156.726 0 0 1-6.437-5.87c7.214-7.889 14.423-17.06
21.459-27.246c12.376-1.098 24.068-2.894 34.671-5.345a134.17 134.17 0 0 1 1.386 6.193ZM87.276 214.515c-7.882 2.783-14.16
2.863-17.955.675c-8.075-4.657-11.432-22.636-6.853-46.752a156.923 156.923 0 0 1 1.869-8.499c10.486 2.32 22.093 3.988
34.498 4.994c7.084 9.967 14.501 19.128 21.976 27.15a134.668 134.668 0 0 1-4.877 4.492c-9.933 8.682-19.886 14.842-28.658
17.94ZM50.35 144.747c-12.483-4.267-22.792-9.812-29.858-15.863c-6.35-5.437-9.555-10.836-9.555-15.216c0-9.322 13.897-
21.212 37.076-29.293c2.813-.98 5.757-1.905 8.812-2.773c3.204 10.42 7.406 21.315 12.477 32.332c-5.137 11.18-9.399 22.249-
12.634 32.792a134.718 134.718 0 0 1-6.318-1.979Zm12.378-84.26c-4.811-24.587-1.616-43.134 6.425-47.789c8.564-4.958 27.502
2.111 47.463 19.835a144.318 144.318 0 0 1 3.841 3.545c-7.438 7.987-14.787 17.08-21.808 26.988c-12.04 1.116-23.565 2.908-
34.161 5.309a160.342 160.342 0 0 1-1.76-7.887Zm110.427 27.268a347.8 347.8 0 0 0-7.785-12.803c8.168 1.033 15.994 2.404
23.343 4.08c-2.206 7.072-4.956 14.465-8.193 22.045a381.151 381.151 0 0 0-7.365-13.322Zm-45.032-43.861c5.044 5.465 10.096
11.566 15.065 18.186a322.04 322.04 0 0 0-30.257-.006c4.974-6.559 10.069-12.652 15.192-18.18ZM82.802 87.83a323.167
323.167 0 0 0-7.227 13.238c-3.184-7.553-5.909-14.98-8.134-22.152c7.304-1.634 15.093-2.97 23.209-3.984a321.524 321.524 0
0 0-7.848 12.897Zm8.081 65.352c-8.385-.936-16.291-2.203-23.593-3.793c2.26-7.3 5.045-14.885 8.298-22.6a321.187 321.187 0
0 0 7.257 13.246c2.594 4.48 5.28 8.868 8.038 13.147Zm37.542 31.03c-5.184-5.592-10.354-11.779-15.403-18.433c4.902.192
9.899.29 14.978.29c5.218 0 10.376-.117 15.453-.343c-4.985 6.774-10.018 12.97-15.028 18.486Zm52.198-57.817c3.422 7.8
6.306 15.345 8.596 22.52c-7.422 1.694-15.436 3.058-23.88 4.071a382.417 382.417 0 0 0 7.859-13.026a347.403 347.403 0 0 0
7.425-13.565Zm-16.898 8.101a358.557 358.557 0 0 1-12.281 19.815a329.4 329.4 0 0 1-23.444.823c-7.967 0-15.716-.248-
23.178-.732a310.202 310.202 0 0 1-12.513-19.846h.001a307.41 307.41 0 0 1-10.923-20.627a310.278 310.278 0 0 1 10.89-
20.637l-.001.001a307.318 307.318 0 0 1 12.413-19.761c7.613-.576 15.42-.876 23.31-.876H128c7.926 0 15.743.303
23.354.883a329.357 329.357 0 0 1 12.335 19.695a358.489 358.489 0 0 1 11.036 20.54a329.472 329.472 0 0 1-11
20.722Zm22.56-122.124c8.572 4.944 11.906 24.881 6.52 51.026c-.344 1.668-.73 3.367-1.15 5.09c-10.622-2.452-22.155-4.275-
34.23-5.408c-7.034-10.017-14.323-19.124-21.64-27.008a160.789 160.789 0 0 1 5.888-5.4c18.9-16.447 36.564-22.941 44.612-
18.3ZM128 90.808c12.625 0 22.86 10.235 22.86 22.86s-10.235 22.86-22.86 22.86s-22.86-10.235-22.86-22.86s10.235-22.86
22.86-22.86Z"></path></svg>
```

smagenda/src/components/layout/AdminShell.tsx

```
import { Link, useLocation } from 'react-router-dom'
import { useAuth } from '../../state/auth/useAuth'
import { Button } from '../ui/Button'

export function AdminShell({ children }: { children: React.ReactNode }) {
  const { signOut, impersonation, stopImpersonation } = useAuth()
  const location = useLocation()
  const nav = [
    { to: '/admin/dashboard', label: 'Dashboard' },
    { to: '/admin/clientes', label: 'Todos os Clientes' },
    { to: '/admin/whatsapp', label: 'WhatsApp Avisos' },
    { to: '/admin/logs', label: 'Logs de Auditoria' },
    { to: '/admin/configuracoes', label: 'Configurações' },
  ]
  return (
    <div className="min-h-screen bg-slate-50">
      <div className="mx-auto max-w-6xl px-4 py-6">
        <div className="mb-6 flex items-center justify-between">
          <div>
            <div className="text-xs font-semibold tracking-wide text-slate-500">SMagenda</div>
            <div className="text-lg font-semibold text-slate-900">Super Admin</div>
          </div>
          <div className="flex items-center gap-2">
            {impersonation ? (
              <Button
                variant="secondary"
                onClick={() => {
                  stopImpersonation()
                }}
              >
                Parar impersonation
              </Button>
            ) : null}
            <Button variant="secondary" onClick={() => signOut()}>
              Sair
            </Button>
          </div>
        </div>

        <div className="grid grid-cols-1 gap-6 md:grid-cols-[240px_1fr]">
          <nav className="md:sticky md:top-6 md:h-fit">
            <div className="rounded-xl border border-slate-200 bg-white p-2">
              {nav.map((item) => {
                const active = location.pathname === item.to
                return (
                  <Link
                    key={item.to}
                    to={item.to}
                    className={[
                      'block rounded-lg px-3 py-2 text-sm font-medium',
                      active ? 'bg-slate-900 text-white' : 'text-slate-700 hover:bg-slate-100',
                    ].join(' ')}
                  >
                    {item.label}
                  </Link>
                )
              })}
            </div>
          </nav>
          <main>{children}</main>
        </div>
      </div>
    </div>
  )
}
```

smagenda/src/components/layout/AppShell.tsx

```
import { Link, useLocation, useNavigate } from 'react-router-dom'
import { useAuth } from '../../state/auth/useAuth'
import { getOptionalEnv } from '../../lib/env'
import { Button } from '../ui/Button'

export function AppShell({ children }: { children: React.ReactNode }) {
  const { principal, appPrincipal, impersonation, stopImpersonation, signOut, masterUsuario } = useAuth()
  const location = useLocation()
  const navigate = useNavigate()
  const isFuncionario = appPrincipal?.kind === 'funcionario'
  const isFuncionarioAdmin = appPrincipal?.kind === 'funcionario' && appPrincipal.profile.permissao === 'admin'
  const funcionario = appPrincipal?.kind === 'funcionario' ? appPrincipal.profile : null
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : isFuncionarioAdmin ? masterUsuario : null
  const temaProspector = usuario?.tema_prospector_habilitado === true

  const plano = String(usuario?.plano ?? '').trim().toLowerCase()
  const isProPlus = plano === 'pro' || plano === 'team' || plano === 'enterprise'

  const supportNumber = getOptionalEnv('VITE_SUPPORT_WHATSAPP_NUMBER')
  const canUseWhatsappSupport = Boolean(supportNumber && usuario && isProPlus)

  const nav = isFuncionario && !isFuncionarioAdmin
    ? [
        { to: '/funcionario/agenda', label: 'Minha Agenda' },
        { to: '/ajuda', label: 'Ajuda' },
      ]
    : isFuncionarioAdmin
      ? [
          ...(funcionario?.pode_ver_agenda ? [{ to: '/dashboard', label: 'Agenda' }] : []),
          ...(funcionario?.pode_ver_agenda ? [{ to: '/funcionario/agenda', label: 'Minha Agenda' }] : []),
          ...(funcionario?.pode_gerenciar_servicos ? [{ to: '/servicos', label: 'Meus Serviços' }] : []),
          ...(funcionario?.pode_ver_clientes_de_outros ? [{ to: '/clientes', label: 'Clientes' }] : []),
          ...(isProPlus && funcionario?.pode_ver_financeiro ? [{ to: '/relatorios', label: 'Relatórios' }] : []),
          { to: '/ajuda', label: 'Ajuda' },
        ]
      : [
          { to: '/dashboard', label: 'Agenda' },
          { to: '/servicos', label: 'Meus Serviços' },
          { to: '/clientes', label: 'Clientes' },
          ...(isProPlus ? [{ to: '/relatorios', label: 'Relatórios' }] : []),
          { to: '/pagamento', label: 'Pagamento' },
          { to: '/funcionarios', label: 'Funcionários' },
          { to: '/configuracoes/whatsapp', label: 'WhatsApp' },
          { to: '/configuracoes/mensagens', label: 'Mensagens Automáticas' },
          { to: '/configuracoes/pagina-publica', label: 'Página Pública' },
          { to: '/ajuda', label: 'Ajuda' },
        ]

  return (
    <div className={['min-h-screen bg-slate-50', temaProspector ? 'theme-prospector' : ''].filter(Boolean).join(' ')}>
      <div className="mx-auto max-w-6xl px-4 py-6">
        <div className="mb-6 flex items-center justify-between">
          <div>
            <div className="text-xs font-semibold tracking-wide text-slate-500">SMagenda</div>
            <div className="text-lg font-semibold text-slate-900">
              {appPrincipal?.kind === 'usuario'
                ? appPrincipal.profile.nome_negocio
                : appPrincipal?.kind === 'funcionario'
                  ? `Olá, ${appPrincipal.profile.nome_completo}`
                  : ''}
            </div>
          </div>
          <div className="flex items-center gap-2">
            {canUseWhatsappSupport ? (
              <a
                className="inline-flex items-center justify-center rounded-lg px-4 py-2 text-sm font-medium transition
focus:outline-none focus:ring-2 focus:ring-slate-300 bg-emerald-600 text-white hover:bg-emerald-500"
                href={`https://wa.me/${encodeURIComponent(String(supportNumber))}?text=${encodeURIComponent('Olá!
Preciso de suporte no SMagenda.')}`}
                target="_blank"
```

```
              rel="noreferrer"
            >
              Suporte WhatsApp
            </a>
          ) : null}
          {principal?.kind === 'super_admin' && impersonation ? (
            <Button
              variant="secondary"
              onClick={() => {
                stopImpersonation()
                navigate('/admin/clientes')
              }}
            >
              Voltar ao admin
            </Button>
          ) : null}
          <Button variant="secondary" onClick={() => signOut()}>
            Sair
          </Button>
        </div>
      </div>

      <div className="grid grid-cols-1 gap-6 md:grid-cols-[240px_1fr]">
        <nav className="md:sticky md:top-6 md:h-fit">
          <div className="rounded-xl border border-slate-200 bg-white p-2">
            {nav.map((item) => {
              const active = location.pathname === item.to
              return (
                <Link
                  key={item.to}
                  to={item.to}
                  className={[
                    'block rounded-lg px-3 py-2 text-sm font-medium',
                    active ? 'bg-slate-900 text-white' : 'text-slate-700 hover:bg-slate-100',
                  ].join(' ')}
                >
                  {item.label}
                </Link>
              )
            })}
          </div>
        </nav>

        <main>{children}</main>
      </div>
    </div>
  </div>
)
}
```

smagenda/src/components/ui/Badge.tsx

```
export function Badge({
  children,
  tone = 'slate',
}: {
  children: React.ReactNode
  tone?: 'slate' | 'green' | 'yellow' | 'red'
}) {
  const map: Record<string, string> = {
    slate: 'bg-slate-100 text-slate-700',
    green: 'bg-emerald-100 text-emerald-800',
    yellow: 'bg-amber-100 text-amber-800',
    red: 'bg-rose-100 text-rose-800',
  }
  return (
    <span className={['inline-flex items-center rounded-full px-2 py-0.5 text-xs font-medium', map[tone]].join(' ')}>
      {children}
    </span>
  )
}
```

smagenda/src/components/ui/Button.tsx

```
import type { ButtonHTMLAttributes } from 'react'

type Props = ButtonHTMLAttributes<HTMLButtonElement> & {
  variant?: 'primary' | 'secondary' | 'ghost' | 'danger'
  fullWidth?: boolean
}

export function Button({ variant = 'primary', fullWidth, className, ...props }: Props) {
  const base =
    'inline-flex items-center justify-center rounded-lg px-4 py-2 text-sm font-medium transition focus:outline-none
focus:ring-2 focus:ring-slate-300 disabled:opacity-60 disabled:cursor-not-allowed transform transition-transform
active:scale-[0.98]'
  const styles: Record<string, string> = {
    primary: 'bg-slate-900 text-white hover:bg-slate-800',
    secondary: 'bg-white text-slate-900 border border-slate-200 hover:bg-slate-50',
    ghost: 'bg-transparent text-slate-700 hover:bg-slate-100',
    danger: 'bg-rose-600 text-white hover:bg-rose-500',
  }

  return (
    <button
      className={[base, styles[variant], fullWidth ? 'w-full' : '', className ?? '']
        .filter(Boolean)
        .join(' ')}
      {...props}
    />
  )
}
```

smagenda/src/components/ui/Card.tsx

```
export function Card({ children }: { children: React.ReactNode }) {
  return <div className="rounded-xl border border-slate-200 bg-white shadow-sm">{children}</div>
}
```

smagenda/src/components/ui/Input.tsx

```
import { useState, type InputHTMLAttributes } from 'react'

type Props = InputHTMLAttributes<HTMLInputElement> & {
  label?: string
}

export function Input({ label, className, type, ...props }: Props) {
  const isPassword = String(type ?? '') === 'password'
  const [showPassword, setShowPassword] = useState(false)

  const effectiveType = isPassword ? (showPassword ? 'text' : 'password') : type

  return (
    <label className="block">
      {label ? <div className="text-sm font-medium text-slate-700 mb-1">{label}</div> : null}
      <div className="relative">
        <input
          className={[
            'w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900 outline-none 
focus:ring-2 focus:ring-slate-300',
            isPassword ? 'pr-12' : '',
            className ?? '',
          ]
            .filter(Boolean)
            .join(' ')}
          type={effectiveType}
          {...props}
        />
        {isPassword ? (
          <button
            type="button"
            className="absolute right-2 top-1/2 -translate-y-1/2 rounded-md px-2 py-1 text-xs font-medium text-slate-700 
hover:bg-slate-100 focus:outline-none focus:ring-2 focus:ring-slate-300"
            onClick={() => setShowPassword((v) => !v)}
            aria-label={showPassword ? 'Ocultar senha' : 'Mostrar senha'}
          >
            {showPassword ? 'Ocultar' : 'Ver'}
          </button>
        ) : null}
      </div>
    </label>
  )
}
```

smagenda/src/components/ui/TutorialOverlay.tsx

```
import { useEffect, useMemo, useState } from 'react'
import type { ReactNode } from 'react'
import { Button } from './Button'

type TutorialStep<TTarget extends string = string> = {
  title: string
  body: string
  target?: TTarget
}

export function PageTutorial(args: {
  usuarioId: string | null | undefined
  page: string
  children: (state: {
    tutorialOpen: boolean
    tutorialStep: number
    setTutorialStep: (next: number) => void
    openTutorial: () => void
    resetTutorial: () => void
    closeTutorial: () => void
  }) => ReactNode
}) {
  const tutorialKey = useMemo(() => {
    const usuarioId = (args.usuarioId ?? '').trim()
    if (!usuarioId) return null
    const page = String(args.page ?? '').trim().toLowerCase() || 'page'
    return `smagenda:tutorial:${page}:${usuarioId}`
  }, [args.page, args.usuarioId])

  const [tutorialOpen, setTutorialOpen] = useState(false)
  const [tutorialStep, setTutorialStep] = useState(0)

  useEffect(() => {
    if (!tutorialKey) return
    let active = true
    const timeoutId = setTimeout(() => {
      if (!active) return
      try {
        const done = localStorage.getItem(tutorialKey)
        if (done !== 'done') {
          setTutorialOpen(true)
          setTutorialStep(0)
        }
      } catch {
        setTutorialOpen(true)
        setTutorialStep(0)
      }
    }, 0)

    return () => {
      active = false
      clearTimeout(timeoutId)
    }
  }, [tutorialKey])

  const openTutorial = () => {
    setTutorialOpen(true)
    setTutorialStep(0)
  }

  const resetTutorial = () => {
    if (tutorialKey) {
      try {
        localStorage.removeItem(tutorialKey)
      } catch {
        void 0
      }
    }
    setTutorialOpen(true)
    setTutorialStep(0)
```

```
  }

  const closeTutorial = () => {
    if (tutorialKey) {
      try {
        localStorage.setItem(tutorialKey, 'done')
      } catch {
        void 0
      }
    }
    setTutorialOpen(false)
  }

  return args.children({ tutorialOpen, tutorialStep, setTutorialStep, openTutorial, resetTutorial, closeTutorial })
}

export function TutorialOverlay<TTarget extends string>(args: {
  open: boolean
  steps: ReadonlyArray<TutorialStep<TTarget>>
  step: number
  onStepChange: (next: number) => void
  onClose: () => void
  titleFallback?: string
}) {
  if (!args.open) return null

  const max = Math.max(0, args.steps.length - 1)
  const step = Math.min(Math.max(0, args.step), max)
  const current = args.steps[step]
  const title = current?.title ?? args.titleFallback ?? 'Tutorial'
  const body = current?.body ?? ''

  return (
    <div className="fixed inset-0 z-50">
      <div className="absolute inset-0 bg-slate-950/50" />
      <div className="absolute inset-0 p-4 flex items-end sm:items-center sm:justify-center">
        <div className="w-full max-w-md rounded-2xl border border-slate-200 bg-white shadow-xl">
          <div className="p-5 space-y-3">
            <div className="text-sm font-semibold text-slate-900">{title}</div>
            <div className="text-sm text-slate-600">{body}</div>

            <div className="flex items-center justify-between gap-2 pt-2">
              <Button variant="secondary" onClick={args.onClose}>
                Pular
              </Button>
              <div className="flex items-center gap-2">
                <Button variant="secondary" onClick={() => args.onStepChange(Math.max(0, step - 1))} disabled={step ===
0}>
                  Voltar
                </Button>
                <Button
                  onClick={() => {
                    if (step >= max) {
                      args.onClose()
                      return
                    }
                    args.onStepChange(Math.min(max, step + 1))
                  }}
                >
                  {step >= max ? 'Concluir' : 'Próximo'}
                </Button>
              </div>
            </div>

            <div className="text-xs text-slate-500">
              {args.steps.length ? step + 1 : 0} de {args.steps.length}
            </div>
          </div>
        </div>
      </div>
    </div>
```

```
  )
}
```

smagenda/src/index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

html,
body,
#root {
  height: 100%;
}

:root {
  --sm-bg: #f8fafc;
  --sm-surface: #ffffff;
  --sm-surface2: #f1f5f9;
  --sm-border: rgba(15, 23, 42, 0.12);
  --sm-text: rgba(15, 23, 42, 0.95);
  --sm-muted: rgba(15, 23, 42, 0.7);
  --sm-muted2: rgba(15, 23, 42, 0.55);
  --sm-shadow: 0 8px 20px rgba(2, 6, 23, 0.06);
  --sm-ring: rgba(148, 163, 184, 0.55);
  --sm-primary: #0f172a;
  --sm-primary-hover: #111827;
  --sm-danger: #e11d48;
  --sm-success: #059669;
  --sm-link: #2563eb;
}

.theme-prospector {
  --sm-bg: #0b1220;
  --sm-surface: #0f1b31;
  --sm-surface2: #0c172b;
  --sm-border: rgba(255, 255, 255, 0.08);
  --sm-text: rgba(255, 255, 255, 0.92);
  --sm-muted: rgba(255, 255, 255, 0.68);
  --sm-muted2: rgba(255, 255, 255, 0.52);
  --sm-shadow: 0 18px 50px rgba(0, 0, 0, 0.35);
  --sm-ring: rgba(148, 163, 184, 0.25);
  --sm-primary: rgba(99, 102, 241, 0.95);
  --sm-primary-hover: rgba(79, 70, 229, 0.95);
  --sm-danger: #f43f5e;
  --sm-success: #22c55e;
  --sm-link: rgba(147, 197, 253, 0.95);
  background: radial-gradient(1200px 600px at 15% 10%, rgba(99, 102, 241, 0.25), transparent 60%),
    radial-gradient(900px 500px at 90% 0%, rgba(34, 197, 94, 0.18), transparent 60%), var(--sm-bg);
  color: var(--sm-text);
}

.theme-prospector a {
  color: var(--sm-link);
}

.theme-prospector .bg-slate-50 {
  background-color: transparent !important;
}

.theme-prospector .bg-white {
  background-color: var(--sm-surface) !important;
}

.theme-prospector .bg-slate-100 {
  background-color: rgba(255, 255, 255, 0.04) !important;
}

.theme-prospector .border-slate-200,
.theme-prospector .divide-slate-100 {
  border-color: var(--sm-border) !important;
}

.theme-prospector .text-slate-900 {
  color: var(--sm-text) !important;
```

```
}

.theme-prospector .text-slate-800,
.theme-prospector .text-slate-700 {
  color: var(--sm-muted) !important;
}

.theme-prospector .text-slate-600,
.theme-prospector .text-slate-500 {
  color: var(--sm-muted2) !important;
}

.theme-prospector .shadow-sm {
  box-shadow: var(--sm-shadow) !important;
}

.theme-prospector .bg-slate-900 {
  background-color: var(--sm-primary) !important;
}

.theme-prospector .hover\:bg-slate-800:hover {
  background-color: var(--sm-primary-hover) !important;
}

.theme-prospector .hover\:bg-slate-50:hover {
  background-color: rgba(255, 255, 255, 0.04) !important;
}

.theme-prospector .hover\:bg-slate-100:hover {
  background-color: rgba(255, 255, 255, 0.06) !important;
}

.theme-prospector .focus\:ring-slate-300:focus {
  --tw-ring-color: var(--sm-ring) !important;
}

.theme-prospector .bg-emerald-600 {
  background-color: var(--sm-success) !important;
}

.theme-prospector .hover\:bg-emerald-500:hover {
  background-color: rgba(34, 197, 94, 0.85) !important;
}

.theme-prospector .bg-rose-600 {
  background-color: var(--sm-danger) !important;
}

.theme-prospector .hover\\:bg-rose-500:hover {
  background-color: rgba(244, 63, 94, 0.9) !important;
}

.theme-prospector select,
.theme-prospector textarea,
.theme-prospector input {
  background-color: var(--sm-surface) !important;
  color: var(--sm-text) !important;
  border-color: var(--sm-border) !important;
}

.theme-prospector select option {
  background-color: #0b1220;
  color: rgba(255, 255, 255, 0.92);
}
```

smagenda/src/lib/dates.ts

```
export function toISODate(value: Date): string {
  const year = value.getFullYear()
  const month = String(value.getMonth() + 1).padStart(2, '0')
  const day = String(value.getDate()).padStart(2, '0')
  return `${year}-${month}-${day}`
}

export function formatBRDate(value: Date): string {
  return value.toLocaleDateString('pt-BR')
}

export function formatBRMoney(value: number): string {
  return value.toLocaleString('pt-BR', { style: 'currency', currency: 'BRL' })
}

export function parseTimeToMinutes(value: string): number {
  const [hh, mm] = value.split(':').map((v) => Number(v))
  return hh * 60 + mm
}

export function minutesToTime(value: number): string {
  const hh = Math.floor(value / 60)
  const mm = value % 60
  return `${String(hh).padStart(2, '0')}:${String(mm).padStart(2, '0')}`
}

export function normalizeTimeHHMM(value: string): string {
  const raw = String(value ?? '').trim()
  if (!raw) return ''
  const parts = raw.split(':')
  if (parts.length < 2) return raw
  const hh = String(parts[0] ?? '').padStart(2, '0')
  const mm = String(parts[1] ?? '').padStart(2, '0')
  return `${hh}:${mm}`
}
```

smagenda/src/lib/env.ts

```
export function getRequiredEnv(name: string): string {
  const value = import.meta.env[name] as string | undefined
  if (!value) throw new Error(`Missing env var: ${name}`)
  return value
}

export function getOptionalEnv(name: string): string | null {
  const raw = import.meta.env[name] as string | undefined
  const value = String(raw ?? '').trim()
  return value ? value : null
}

export type EnvCheck = { ok: true; values: Record<string, string> } | { ok: false; missing: string[] }

export function checkRequiredEnvs(names: string[]): EnvCheck {
  const values: Record<string, string> = {}
  const missing: string[] = []

  for (const name of names) {
    const value = import.meta.env[name] as string | undefined
    if (!value) {
      missing.push(name)
      continue
    }
    values[name] = value
  }

  if (missing.length > 0) return { ok: false, missing }
  return { ok: true, values }
}

export type PublicBrandConfig = {
  productName: string
  companyName: string
  companyLogoUrl: string | null
}

export function getPublicBrandConfig(): PublicBrandConfig {
  const productName = getOptionalEnv('VITE_PUBLIC_PRODUCT_NAME') ?? getOptionalEnv('VITE_PUBLIC_BRAND_NAME') ??
'SMagenda'
  const companyName = getOptionalEnv('VITE_PUBLIC_COMPANY_NAME') ?? 'SingleMotion'
  const companyLogoUrl = getOptionalEnv('VITE_PUBLIC_COMPANY_LOGO_URL') ?? getOptionalEnv('VITE_PUBLIC_BRAND_LOGO_URL')
  return { productName, companyName, companyLogoUrl }
}
```

smagenda/src/lib/slug.ts

```
export function slugify(input: string): string {
  return input
    .trim()
    .toLowerCase()
    .normalize('NFD')
    .replace(/[\u0300-\u036f]/g, '')
    .replace(/[^a-z0-9]+/g, '-')
    .replace(/(^-|-$)/g, '')
}
```

smagenda/src/lib/supabase.ts

```
import { createClient } from '@supabase/supabase-js'
import { checkRequiredEnvs } from './env'

export const supabaseEnv = checkRequiredEnvs(['VITE_SUPABASE_URL', 'VITE_SUPABASE_ANON_KEY'])

function cleanEnvString(value: string) {
  return String(value ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
}

const supabaseUrl = supabaseEnv.ok ? cleanEnvString(supabaseEnv.values.VITE_SUPABASE_URL).replace(/\/+$/g, '') :
'http://localhost'
const supabaseAnonKey = supabaseEnv.ok ? cleanEnvString(supabaseEnv.values.VITE_SUPABASE_ANON_KEY) : 'missing'

const fetchWithApiKey=[REDACTED]
  const headers = new Headers(input instanceof Request ? input.headers : undefined)
  const initHeaders = new Headers(init?.headers ?? undefined)
  initHeaders.forEach((value, key) => headers.set(key, value))
  if (!headers.has('apikey')) headers.set('apikey', supabaseAnonKey)
  return fetch(input, { ...(init ?? {}), headers })
}

export const supabase = createClient(supabaseUrl, supabaseAnonKey, {
  global: {
    fetch: fetchWithApiKey,
  },
})

function decodeJwtPayload(jwt: string): Record<string, unknown> | null {
  const parts = jwt.split('.')
  if (parts.length !== 3) return null
  const payload = parts[1] ?? ''
  if (!payload) return null
  const b64 = payload.replace(/-/g, '+').replace(/_/g, '/')
  const padded = b64 + '='.repeat((4 - (b64.length % 4)) % 4)
  try {
    const json = atob(padded)
    const parsed = JSON.parse(json) as unknown
    if (!parsed || typeof parsed !== 'object') return null
    return parsed as Record<string, unknown>
  } catch {
    return null
  }
}

export function checkJwtProject(jwt: string, supabaseUrlValue: string) {
  const expectedPrefix = `${supabaseUrlValue.replace(/\/+$/, '')}/auth/v1`
  const payload = decodeJwtPayload(jwt)
  const iss = typeof payload?.iss === 'string' ? payload.iss : null
  const ok = Boolean(iss && iss.startsWith(expectedPrefix))
  if (ok) return { ok: true as const }
  return { ok: false as const, iss, expectedPrefix }
}
```

smagenda/src/main.tsx

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import { BrowserRouter } from 'react-router-dom'
import './index.css'
import App from './App.tsx'
import { AuthProvider } from './state/auth/AuthProvider'
import { supabaseEnv } from './lib/supabase'

if (import.meta.env.PROD && 'serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('/sw.js').catch(() => null)
  })
}

createRoot(document.getElementById('root')!).render(
  <StrictMode>
    {supabaseEnv.ok ? (
      <BrowserRouter>
        <AuthProvider>
          <App />
        </AuthProvider>
      </BrowserRouter>
    ) : (
      <div className="min-h-screen bg-slate-50 flex items-center justify-center px-4">
        <div className="w-full max-w-lg rounded-2xl border border-slate-200 bg-white p-6 shadow-sm space-y-4">
          <div>
            <div className="text-xs font-semibold tracking-wide text-slate-500">SMagenda</div>
            <div className="text-2xl font-semibold text-slate-900">Configuração necessária</div>
          </div>

          <div className="text-sm text-slate-700">Defina as variáveis de ambiente do Supabase para usar dados reais.
</div>

          <div className="rounded-xl border border-rose-200 bg-rose-50 p-4">
            <div className="text-sm font-semibold text-rose-800">Faltando</div>
            <div className="mt-2 space-y-1">
              {supabaseEnv.missing.map((k) => (
                <div key={k} className="font-mono text-xs text-rose-800">
                  {k}
                </div>
              ))}
            </div>
          </div>

          <div className="text-sm text-slate-700">
            Crie um arquivo <span className="font-mono text-xs">.env</span> ou <span className="font-mono text-
xs">.env.local</span> em{' '}
            <span className="font-mono text-xs">smagenda/</span> e reinicie o dev server.
          </div>
        </div>
      </div>
    )}
  </StrictMode>,
)
```

smagenda/src/state/auth/AuthContext.ts

```
import { createContext } from 'react'
import type { Session } from '@supabase/supabase-js'
import type { Principal, UsuarioProfile } from './types'

export type Impersonation = {
  usuarioId: string
}

export type AuthState = {
  session: Session | null
  principal: Principal | null
  appPrincipal: Principal | null
  masterUsuario: UsuarioProfile | null
  masterUsuarioLoading: boolean
  loading: boolean
  refresh: () => Promise<Principal | null>
  signOut: () => Promise<void>
  impersonation: Impersonation | null
  startImpersonation: (usuarioId: string) => Promise<{ ok: true } | { ok: false; message: string }>
  stopImpersonation: () => void
}

export const AuthContext = createContext<AuthState | null>(null)
```

smagenda/src/state/auth/AuthProvider.tsx

```
import { useCallback, useEffect, useMemo, useState } from 'react'
import { supabase } from '../../lib/supabase'
import type { FuncionarioProfile, Principal, SuperAdminProfile, UsuarioProfile } from './types'
import { AuthContext, type AuthState, type Impersonation } from './AuthContext'

function deriveEmailPrefix(email: string | null) {
  if (!email) return null
  const prefix = email.split('@')[0]?.trim()
  return prefix ? prefix : null
}

function deriveSlug(base: string) {
  const slug = base
    .trim()
    .toLowerCase()
    .replace(/[^a-z0-9]+/g, '-')
    .replace(/(^-+)|(-+$)/g, '')
  return slug ? slug : null
}

function toUsuarioProfile(row: Record<string, unknown>, userId: string): UsuarioProfile {
  const rawEmail = typeof row.email === 'string' ? row.email : null
  const emailPrefix = deriveEmailPrefix(rawEmail)
  const nomeCompleto = typeof row.nome_completo === 'string' && row.nome_completo.trim() ? row.nome_completo :
(emailPrefix ?? 'Usuário')
  const nomeNegocio = typeof row.nome_negocio === 'string' && row.nome_negocio.trim() ? row.nome_negocio : nomeCompleto

  const rawSlug = typeof row.slug === 'string' && row.slug.trim() ? row.slug : null
  const derivedFromEmail = emailPrefix ? deriveSlug(emailPrefix) : null
  const fallbackSlug = `u-${userId.replace(/-/g, '').slice(0, 8)}`
  const slug = rawSlug ?? derivedFromEmail ?? fallbackSlug

  const plano = (typeof row.plano === 'string' ? row.plano : 'free') as UsuarioProfile['plano']
  const tipoConta = (typeof row.tipo_conta === 'string' ? row.tipo_conta : 'master') as UsuarioProfile['tipo_conta']
  const statusPagamento = (typeof row.status_pagamento === 'string' ? row.status_pagamento : 'trial') as
UsuarioProfile['status_pagamento']
  const freeTrialConsumido = row.free_trial_consumido === true
  const tipoNegocio = typeof row.tipo_negocio === 'string' ? row.tipo_negocio : null
  const temaProspectorHabilitado = row.tema_prospector_habilitado === true

  return {
    id: userId,
    nome_completo: nomeCompleto,
    nome_negocio: nomeNegocio,
    slug,
    tipo_negocio: tipoNegocio,
    logo_url: typeof row.logo_url === 'string' ? row.logo_url : null,
    telefone: typeof row.telefone === 'string' ? row.telefone : null,
    email: rawEmail ?? '',
    endereco: typeof row.endereco === 'string' ? row.endereco : null,
    horario_inicio: typeof row.horario_inicio === 'string' ? row.horario_inicio : null,
    horario_fim: typeof row.horario_fim === 'string' ? row.horario_fim : null,
    dias_trabalho: Array.isArray(row.dias_trabalho) ? (row.dias_trabalho as number[]) : null,
    intervalo_inicio: typeof row.intervalo_inicio === 'string' ? row.intervalo_inicio : null,
    intervalo_fim: typeof row.intervalo_fim === 'string' ? row.intervalo_fim : null,
    whatsapp_api_url: typeof row.whatsapp_api_url === 'string' ? row.whatsapp_api_url : null,
    stripe_customer_id: typeof row.stripe_customer_id === 'string' ? row.stripe_customer_id : null,
    plano,
    tipo_conta: tipoConta,
    limite_funcionarios: typeof row.limite_funcionarios === 'number' ? row.limite_funcionarios : null,
    status_pagamento: statusPagamento,
    data_vencimento: typeof row.data_vencimento === 'string' ? row.data_vencimento : null,
    free_trial_consumido: freeTrialConsumido,
    tema_prospector_habilitado: temaProspectorHabilitado,
    ativo: typeof row.ativo === 'boolean' ? row.ativo : true,
  }
}

function toFuncionarioProfile(row: Record<string, unknown>, userId: string): FuncionarioProfile {
  const permissao = row.permissao === 'admin' ? 'admin' : row.permissao === 'atendente' ? 'atendente' : 'funcionario'
```

```
  return {
    id: userId,
    usuario_master_id: String(row.usuario_master_id ?? ''),
    nome_completo: String(row.nome_completo ?? ''),
    email: String(row.email ?? ''),
    telefone: typeof row.telefone === 'string' ? row.telefone : null,
    permissao,
    pode_ver_agenda: row.pode_ver_agenda !== false,
    pode_criar_agendamentos: row.pode_criar_agendamentos !== false,
    pode_cancelar_agendamentos: row.pode_cancelar_agendamentos !== false,
    pode_bloquear_horarios: row.pode_bloquear_horarios !== false,
    pode_ver_financeiro: row.pode_ver_financeiro === true,
    pode_gerenciar_servicos: row.pode_gerenciar_servicos === true,
    pode_ver_clientes_de_outros: row.pode_ver_clientes_de_outros === true,
    horario_inicio: typeof row.horario_inicio === 'string' ? row.horario_inicio : null,
    horario_fim: typeof row.horario_fim === 'string' ? row.horario_fim : null,
    dias_trabalho: Array.isArray(row.dias_trabalho) ? (row.dias_trabalho as number[]) : null,
    intervalo_inicio: typeof row.intervalo_inicio === 'string' ? row.intervalo_inicio : null,
    intervalo_fim: typeof row.intervalo_fim === 'string' ? row.intervalo_fim : null,
    capacidade_dia_inteiro: typeof row.capacidade_dia_inteiro === 'number' &&
Number.isFinite(row.capacidade_dia_inteiro) ? Math.max(1, Math.min(2, Math.floor(row.capacidade_dia_inteiro))) : 1,
    ativo: row.ativo !== false,
  }
}

async function resolvePrincipal(userId: string): Promise<Principal | null> {
  const isMissingTable = (msg: string) => msg.includes('schema cache') || msg.includes("Could not find the table
'public.")

  const { data: superAdmin, error: superAdminErr } = await
supabase.from('super_admin').select('id,nome,email,nivel').eq('id', userId).maybeSingle()
  if (superAdminErr) {
    if (!isMissingTable(superAdminErr.message)) return null
  }
  if (superAdmin) return { kind: 'super_admin', profile: superAdmin as unknown as SuperAdminProfile }

  const { data: funcionario, error: funcionarioErr } = await supabase
    .from('funcionarios')
    .select('*')
    .eq('id', userId)
    .maybeSingle()
  if (funcionarioErr) {
    if (!isMissingTable(funcionarioErr.message)) return null
  }
  if (funcionario) return { kind: 'funcionario', profile: toFuncionarioProfile(funcionario as unknown as Record<string,
unknown>, userId) }

  const { data: usuario, error: usuarioErr } = await supabase
    .from('usuarios')
    .select('*')
    .eq('id', userId)
    .maybeSingle()
  if (usuarioErr) {
    if (!isMissingTable(usuarioErr.message)) return null
  }
  if (usuario) return { kind: 'usuario', profile: toUsuarioProfile(usuario as unknown as Record<string, unknown>,
userId) }

  const { error: ensureErr } = await supabase.rpc('ensure_usuario_profile')
  if (!ensureErr) {
    const { data: usuario2 } = await supabase
      .from('usuarios')
      .select('*')
      .eq('id', userId)
      .maybeSingle()
    if (usuario2) return { kind: 'usuario', profile: toUsuarioProfile(usuario2 as unknown as Record<string, unknown>,
userId) }
  }

  return null
}
```

```
export function AuthProvider({ children }: { children: React.ReactNode }) {
  const [session, setSession] = useState<import('@supabase/supabase-js').Session | null>(null)
  const [principal, setPrincipal] = useState<Principal | null>(null)
  const [masterUsuario, setMasterUsuario] = useState<UsuarioProfile | null>(null)
  const [masterUsuarioLoading, setMasterUsuarioLoading] = useState(false)
  const [loading, setLoading] = useState(true)

  const [impersonation, setImpersonation] = useState<Impersonation | null>(() => {
    try {
      const raw = window.localStorage.getItem('smagenda:impersonation')
      if (!raw) return null
      const parsed = JSON.parse(raw) as unknown
      if (!parsed || typeof parsed !== 'object') return null
      const p = parsed as Record<string, unknown>
      const usuarioId = typeof p.usuarioId === 'string' ? p.usuarioId : null
      return usuarioId ? { usuarioId } : null
    } catch {
      return null
    }
  })

  const [impersonatedUsuario, setImpersonatedUsuario] = useState<UsuarioProfile | null>(null)

  const refresh = useCallback(async () => {
    const { data } = await supabase.auth.getSession()
    setSession(data.session)
    if (data.session?.user?.id) {
      const next = await resolvePrincipal(data.session.user.id)
      setPrincipal(next)
      return next
    } else {
      setPrincipal(null)
      return null
    }
  }, [])

  const signOut = useCallback(async () => {
    await supabase.auth.signOut()
    setSession(null)
    setPrincipal(null)
    setImpersonation(null)
    setImpersonatedUsuario(null)
    try {
      window.localStorage.removeItem('smagenda:impersonation')
    } catch {
      return
    }
  }, [])

  const stopImpersonation = useCallback(() => {
    setImpersonation(null)
    setImpersonatedUsuario(null)
    try {
      window.localStorage.removeItem('smagenda:impersonation')
    } catch {
      return
    }
  }, [])

  const startImpersonation = useCallback(
    async (usuarioId: string) => {
      if (principal?.kind !== 'super_admin') return { ok: false as const, message: 'Sem permissão.' }
      const id = usuarioId.trim()
      if (!id) return { ok: false as const, message: 'Cliente inválido.' }

      const { data, error } = await supabase.from('usuarios').select('*').eq('id', id).maybeSingle()
      if (error) return { ok: false as const, message: error.message }
      if (!data) return { ok: false as const, message: 'Cliente não encontrado na tabela usuarios.' }

      const profile = toUsuarioProfile(data as unknown as Record<string, unknown>, id)
      setImpersonation({ usuarioId: id })
      setImpersonatedUsuario(profile)
```

```
      try {
        window.localStorage.setItem('smagenda:impersonation', JSON.stringify({ usuarioId: id }))
      } catch {
        return { ok: true as const }
      }
      return { ok: true as const }
    },
    [principal?.kind]
  )

  useEffect(() => {
    const run = async () => {
      if (!impersonation) {
        setImpersonatedUsuario(null)
        return
      }
      if (principal?.kind !== 'super_admin') {
        stopImpersonation()
        return
      }

      const { data, error } = await supabase.from('usuarios').select('*').eq('id',
impersonation.usuarioId).maybeSingle()
      if (error || !data) {
        stopImpersonation()
        return
      }
      setImpersonatedUsuario(toUsuarioProfile(data as unknown as Record<string, unknown>, impersonation.usuarioId))
    }

    run().catch(() => {
      stopImpersonation()
    })
  }, [impersonation, principal?.kind, stopImpersonation])

  useEffect(() => {
    void (async () => {
      await Promise.resolve()
      await refresh()
    })()
      .catch(() => undefined)
      .finally(() => setLoading(false))

    const { data: subscription } = supabase.auth.onAuthStateChange((_event, nextSession) => {
      setSession(nextSession)
      if (!nextSession?.user?.id) {
        setPrincipal(null)
        return
      }
      resolvePrincipal(nextSession.user.id)
        .then((p) => setPrincipal(p))
        .catch(() => setPrincipal(null))
    })

    return () => {
      subscription.subscription.unsubscribe()
    }
  }, [refresh])

  useEffect(() => {
    const onFocus = () => {
      refresh().catch(() => undefined)
    }
    window.addEventListener('focus', onFocus)

    const onVisibility = () => {
      if (document.visibilityState === 'visible') refresh().catch(() => undefined)
    }
    document.addEventListener('visibilitychange', onVisibility)

    return () => {
      window.removeEventListener('focus', onFocus)
```

```
      document.removeEventListener('visibilitychange', onVisibility)
    }
  }, [refresh])

  useEffect(() => {
    const run = async () => {
      if (principal?.kind !== 'funcionario') {
        setMasterUsuario(null)
        setMasterUsuarioLoading(false)
        return
      }

      const masterIdRaw = principal.profile.usuario_master_id
      const masterId = typeof masterIdRaw === 'string' ? masterIdRaw.trim() : ''
      if (!masterId) {
        setMasterUsuario(null)
        setMasterUsuarioLoading(false)
        return
      }

      setMasterUsuarioLoading(true)
      const { data, error } = await supabase.from('usuarios').select('*').eq('id', masterId).maybeSingle()
      if (error || !data) {
        setMasterUsuario(null)
        setMasterUsuarioLoading(false)
        return
      }

      setMasterUsuario(toUsuarioProfile(data as unknown as Record<string, unknown>, masterId))
      setMasterUsuarioLoading(false)
    }

    run().catch(() => {
      setMasterUsuario(null)
      setMasterUsuarioLoading(false)
    })
  }, [principal])

  const appPrincipal = useMemo<Principal | null>(() => {
    if (principal?.kind === 'super_admin' && impersonation && impersonatedUsuario) {
      return { kind: 'usuario', profile: impersonatedUsuario }
    }
    return principal
  }, [impersonatedUsuario, impersonation, principal])

  const value = useMemo<AuthState>(
    () => ({
      session,
      principal,
      appPrincipal,
      masterUsuario,
      masterUsuarioLoading,
      loading,
      refresh,
      signOut,
      impersonation,
      startImpersonation,
      stopImpersonation,
    }),
    [session, principal, appPrincipal, masterUsuario, masterUsuarioLoading, loading, refresh, signOut, impersonation, startImpersonation, stopImpersonation]
  )

  return <AuthContext.Provider value={value}>{children}</AuthContext.Provider>
}
```

smagenda/src/state/auth/RequireAuth.tsx

```
import { useEffect, useState } from 'react'
import { Navigate, useLocation } from 'react-router-dom'
import { useAuth } from './useAuth'
import type { Principal } from './types'
import { supabase } from '../../lib/supabase'

type Props = {
  children: React.ReactNode
  requiredKind?: Principal['kind']
  allowFuncionarioAdmin?: boolean
}

export function RequireAuth({ children, requiredKind, allowFuncionarioAdmin }: Props) {
  const { loading, session, principal, appPrincipal, impersonation, signOut } = useAuth()
  const location = useLocation()
  const [masterBlocked, setMasterBlocked] = useState(false)
  const [checkingMaster, setCheckingMaster] = useState(false)
  const masterId = principal?.kind === 'funcionario' ? principal.profile.usuario_master_id : null

  useEffect(() => {
    const run = async () => {
      if (!session || !masterId) {
        setMasterBlocked(false)
        setCheckingMaster(false)
        return
      }

      setCheckingMaster(true)
      const { data, error } = await supabase
        .from('usuarios')
        .select('status_pagamento,data_vencimento,ativo')
        .eq('id', masterId)
        .maybeSingle()

      if (error || !data) {
        setMasterBlocked(true)
        setCheckingMaster(false)
        return
      }

      if (data.ativo === false) {
        setMasterBlocked(true)
        setCheckingMaster(false)
        return
      }

      const status = String(data.status_pagamento ?? '').trim().toLowerCase()
      if (status && status !== 'ativo' && status !== 'trial') {
        setMasterBlocked(true)
        setCheckingMaster(false)
        return
      }

      const venc = data.data_vencimento
      if (data.status_pagamento === 'trial' && venc) {
        const today = new Date()
        today.setHours(0, 0, 0, 0)
        const exp = new Date(`${venc}T00:00:00`)
        if (Number.isFinite(exp.getTime()) && exp < today) {
          setMasterBlocked(true)
          setCheckingMaster(false)
          return
        }
      }

      setMasterBlocked(false)
      setCheckingMaster(false)
    }

    run().catch(() => {
```

```
      setMasterBlocked(true)
      setCheckingMaster(false)
    })
  }, [masterId, session])

  if (loading) {
    return (
      <div className="min-h-screen bg-slate-50 flex items-center justify-center">
        <div className="text-slate-600">Carregando…</div>
      </div>
    )
  }

  if (checkingMaster) {
    return (
      <div className="min-h-screen bg-slate-50 flex items-center justify-center">
        <div className="text-slate-600">Carregando…</div>
      </div>
    )
  }

  if (!session) {
    const adminPath = location.pathname.startsWith('/admin')
    return <Navigate to={adminPath ? '/admin/login' : '/login'} replace />
  }

  if (!principal) {
    return <Navigate to="/login" replace />
  }

  const inAdmin = location.pathname.startsWith('/admin')
  const inPagamento = location.pathname === '/pagamento'
  const effective = inAdmin ? principal : (appPrincipal ?? principal)

  if (principal.kind === 'funcionario' && masterBlocked) {
    return (
      <div className="min-h-screen bg-slate-50 flex items-center justify-center p-4">
        <div className="w-full max-w-md space-y-3 rounded-xl border border-slate-200 bg-white p-6">
          <div className="text-sm font-semibold text-slate-900">Acesso indisponível</div>
          <div className="text-sm text-slate-600">O acesso do seu estabelecimento está pausado.</div>
          <button
            type="button"
            className="w-full inline-flex items-center justify-center rounded-lg px-4 py-2 text-sm font-medium
transition focus:outline-none focus:ring-2 focus:ring-slate-300 bg-slate-900 text-white hover:bg-slate-800"
            onClick={() => void signOut()}
          >
            Sair
          </button>
        </div>
      </div>
    )
  }

  if (principal.kind === 'usuario') {
    const venc = principal.profile.data_vencimento
    if (principal.profile.status_pagamento === 'trial' && venc) {
      const today = new Date()
      today.setHours(0, 0, 0, 0)
      const exp = new Date(`${venc}T00:00:00`)
      if (Number.isFinite(exp.getTime()) && exp < today) {
        if (inPagamento) return <>{children}</>
        return (
          <div className="min-h-screen bg-slate-50 flex items-center justify-center p-4">
            <div className="w-full max-w-md space-y-3 rounded-xl border border-slate-200 bg-white p-6">
              <div className="text-sm font-semibold text-slate-900">Período de teste expirado</div>
              <div className="text-sm text-slate-600">Seu acesso foi pausado. Regularize o pagamento para reativar.
</div>
              <button
                type="button"
                className="w-full inline-flex items-center justify-center rounded-lg px-4 py-2 text-sm font-medium
transition focus:outline-none focus:ring-2 focus:ring-slate-300 bg-slate-900 text-white hover:bg-slate-800"
                onClick={() => {
```

```
         window.location.href = '/pagamento'
        }}
       >
        Ir para pagamento
       </button>
       <button
        type="button"
        className="w-full inline-flex items-center justify-center rounded-lg border border-slate-200 px-4 py-2 text-sm font-medium transition focus:outline-none focus:ring-2 focus:ring-slate-300 bg-white text-slate-900 hover:bg-slate-50"
        onClick={() => void signOut()}
       >
        Sair
       </button>
      </div>
     </div>
    )
   }
  }

  if (principal.profile.status_pagamento !== 'ativo' && principal.profile.status_pagamento !== 'trial') {
   if (inPagamento) return <>{children}</>
   return (
    <div className="min-h-screen bg-slate-50 flex items-center justify-center p-4">
     <div className="w-full max-w-md space-y-3 rounded-xl border border-slate-200 bg-white p-6">
      <div className="text-sm font-semibold text-slate-900">Pagamento pendente</div>
      <div className="text-sm text-slate-600">Seu acesso está restrito até a regularização do pagamento.</div>
      <button
       type="button"
       className="w-full inline-flex items-center justify-center rounded-lg px-4 py-2 text-sm font-medium transition focus:outline-none focus:ring-2 focus:ring-slate-300 bg-slate-900 text-white hover:bg-slate-800"
       onClick={() => {
        window.location.href = '/pagamento'
       }}
      >
       Ir para pagamento
      </button>
      <button
       type="button"
       className="w-full inline-flex items-center justify-center rounded-lg border border-slate-200 px-4 py-2 text-sm font-medium transition focus:outline-none focus:ring-2 focus:ring-slate-300 bg-white text-slate-900 hover:bg-slate-50"
       onClick={() => void signOut()}
      >
       Sair
      </button>
     </div>
    </div>
   )
  }
 }

 if (principal.kind === 'super_admin' && !inAdmin && !impersonation) {
  return <Navigate to="/admin/dashboard" replace />
 }

 if (requiredKind && effective.kind !== requiredKind) {
  if (
   allowFuncionarioAdmin &&
   requiredKind === 'usuario' &&
   effective.kind === 'funcionario' &&
   effective.profile.permissao === 'admin'
  ) {
   return <>{children}</>
  }
  if (principal.kind === 'super_admin') return <Navigate to="/admin/dashboard" replace />
  if (effective.kind === 'funcionario') return <Navigate to="/funcionario/agenda" replace />
  return <Navigate to="/dashboard" replace />
 }
```

```
  return <>{children}</>
}
```

smagenda/src/state/auth/types.ts

```
export type UsuarioProfile = {
  id: string
  nome_completo: string
  nome_negocio: string
  slug: string
  tipo_negocio: string | null
  logo_url: string | null
  telefone: string | null
  email: string
  endereco: string | null
  horario_inicio: string | null
  horario_fim: string | null
  dias_trabalho: number[] | null
  intervalo_inicio: string | null
  intervalo_fim: string | null
  whatsapp_api_url: string | null
  stripe_customer_id: string | null
  plano: 'free' | 'basic' | 'pro' | 'team' | 'enterprise'
  tipo_conta: 'master' | 'individual'
  limite_funcionarios: number | null
  status_pagamento: 'ativo' | 'inadimplente' | 'cancelado' | 'trial' | 'suspenso'
  data_vencimento: string | null
  free_trial_consumido: boolean
  tema_prospector_habilitado: boolean
  ativo: boolean
}

export type FuncionarioProfile = {
  id: string
  usuario_master_id: string
  nome_completo: string
  email: string
  telefone: string | null
  permissao: 'admin' | 'funcionario' | 'atendente'
  pode_ver_agenda: boolean
  pode_criar_agendamentos: boolean
  pode_cancelar_agendamentos: boolean
  pode_bloquear_horarios: boolean
  pode_ver_financeiro: boolean
  pode_gerenciar_servicos: boolean
  pode_ver_clientes_de_outros: boolean
  horario_inicio: string | null
  horario_fim: string | null
  dias_trabalho: number[] | null
  intervalo_inicio: string | null
  intervalo_fim: string | null
  capacidade_dia_inteiro: number
  ativo: boolean
}

export type SuperAdminProfile = {
  id: string
  nome: string
  email: string
  nivel: 'super_admin' | 'suporte' | number
}

export type Principal =
  | { kind: 'usuario'; profile: UsuarioProfile }
  | { kind: 'funcionario'; profile: FuncionarioProfile }
  | { kind: 'super_admin'; profile: SuperAdminProfile }
```

smagenda/src/state/auth/useAuth.ts

```
import { useContext } from 'react'
import { AuthContext } from './AuthContext'

export function useAuth() {
  const ctx = useContext(AuthContext)
  if (!ctx) throw new Error('AuthContext not found')
  return ctx
}
```

smagenda/src/views/admin/AdminBootstrapPage.tsx

```
import { useMemo, useState } from 'react'
import { useNavigate } from 'react-router-dom'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { supabase } from '../../lib/supabase'

export function AdminBootstrapPage() {
  const navigate = useNavigate()

  const [email, setEmail] = useState('lucioianael@gmail.com')
  const [password, setPassword] = useState('')
  const [submitting, setSubmitting] = useState(false)
  const [resending, setResending] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [info, setInfo] = useState<string | null>(null)
  const [success, setSuccess] = useState(false)
  const [userId, setUserId] = useState<string | null>(null)
  const [sql, setSql] = useState<string | null>(null)

  const canSubmit = useMemo(() => email.trim() && password.trim(), [email, password])

  const ensureAuthenticated = async (): Promise<string> => {
    const { data } = await supabase.auth.getSession()
    const existingUserId = data.session?.user?.id
    if (existingUserId) return existingUserId

    const signInRes = await supabase.auth.signInWithPassword({ email: email.trim(), password })
    if (signInRes.error) {
      const signUpRes = await supabase.auth.signUp({ email: email.trim(), password })
      if (signUpRes.error) {
        throw new Error(signInRes.error.message)
      }

      const retryRes = await supabase.auth.signInWithPassword({ email: email.trim(), password })
      if (retryRes.error) throw new Error(retryRes.error.message)
      const retryUserId = retryRes.data.user?.id
      if (!retryUserId) throw new Error('Falha ao autenticar')
      return retryUserId
    }
    const userId = signInRes.data.user?.id
    if (!userId) throw new Error('Falha ao autenticar')
    return userId
  }

  const canResend = useMemo(() => email.trim().length > 0 && error?.toLowerCase().includes('confirm') === true, [email,
error])

  const resend = async () => {
    if (!email.trim()) return
    setResending(true)
    setInfo(null)
    try {
      const { error: resendErr } = await supabase.auth.resend({ type: 'signup', email: email.trim() })
      if (resendErr) {
        setError(resendErr.message)
        return
      }
      setInfo('Email de confirmação reenviado. Verifique sua caixa de entrada e spam.')
    } finally {
      setResending(false)
    }
  }

  const submit = async () => {
    setSubmitting(true)
    setError(null)
    setInfo(null)
    setSuccess(false)
    setSql(null)
```

```
    try {
      const userId = await ensureAuthenticated()
      setUserId(userId)

      const { data: existing, error: readErr } = await supabase.from('super_admin').select('id').eq('id', userId).maybeSingle()
      if (readErr) {
        const msg = readErr.message
        const missingTable = msg.includes("Could not find the table 'public.super_admin'") || msg.includes('schema cache')
        if (missingTable) {
          setError('Tabela super_admin não existe no Supabase.')
          setSql(buildCreateTableSql(email.trim()))
          return
        }
        throw new Error(msg)
      }

      if (!existing) {
        const { error: insertErr } = await supabase
          .from('super_admin')
          .insert({ id: userId, nome: 'Super Admin', email: email.trim(), nivel: 'super_admin' })
        if (insertErr) {
          setError('Usuário autenticado, mas não está cadastrado como Super Admin.')
          setSql(buildInsertSql(email.trim()))
          return
        }
      }

      setSuccess(true)
      navigate('/admin/dashboard', { replace: true })
    } catch (e: unknown) {
      const msg = e instanceof Error ? e.message : 'Erro ao criar Super Admin'
      const notConfirmed = msg.toLowerCase().includes('confirm')
      setError(notConfirmed ? 'Email não confirmado. Reenvie o email de confirmação ou confirme no Supabase.' : msg)
    } finally {
      setSubmitting(false)
    }
  }

  return (
    <div className="min-h-screen bg-slate-50 flex items-center justify-center p-4">
      <div className="w-full max-w-lg space-y-6">
        <div>
          <div className="text-xs font-semibold tracking-wide text-slate-500">SMagenda</div>
          <div className="text-2xl font-semibold text-slate-900">Bootstrap Super Admin (DEV)</div>
        </div>

        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}
        {info ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">{info}</div> : null}
        {success ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">Criado.</div> : null}
        {userId ? (
          <div className="rounded-xl border border-slate-200 bg-white p-4 text-sm text-slate-700">
            Seu <span className="font-mono text-xs">user_id</span>: <span className="font-mono text-xs">{userId}</span>
          </div>
        ) : null}

        <Card>
          <div className="p-6 space-y-4">
            <Input label="Email" value={email} onChange={(e) => setEmail(e.target.value)} type="email" />
            <Input label="Senha" value={password} onChange={(e) => setPassword(e.target.value)} type="password" autoComplete="new-password" />
            <Button fullWidth onClick={submit} disabled={!canSubmit || submitting}>
              Entrar e checar Super Admin
            </Button>
            {canResend ? (
              <Button fullWidth variant="secondary" onClick={resend} disabled={resending}>
                Reenviar email de confirmação
```

```
              </Button>
            ) : null}
          </div>
        </Card>

        {sql ? (
          <Card>
            <div className="p-6 space-y-3">
              <div className="text-sm font-semibold text-slate-900">SQL para rodar no Supabase</div>
              <pre className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-xs text-slate-800 overflow-auto">
{sql}</pre>
              <div className="text-sm text-slate-700">Depois de executar, tente novamente o login em /admin/login.</div>
            </div>
          </Card>
        ) : null}
      </div>
    </div>
  )
}

function buildCreateTableSql(email: string) {
  const safeEmail = email.replace(/'/g, "''")
  return `create table if not exists public.super_admin (
  id uuid primary key references auth.users(id) on delete cascade,
  nome text not null default 'Super Admin',
  email text not null unique,
  nivel text not null default 'super_admin',
  criado_em timestamptz not null default now()
);

alter table public.super_admin enable row level security;

drop policy if exists "super_admin_select_own" on public.super_admin;
create policy "super_admin_select_own" on public.super_admin
for select to authenticated
using (id = auth.uid());

drop policy if exists "super_admin_insert_first" on public.super_admin;
create policy "super_admin_insert_first" on public.super_admin
for insert to authenticated
with check (id = auth.uid() and not exists (select 1 from public.super_admin));

insert into public.super_admin (id, nome, email, nivel)
select u.id, 'Super Admin', u.email, 'super_admin'
from auth.users u
where u.email = '${safeEmail}'
on conflict (id) do nothing;`
}

function buildInsertSql(email: string) {
  const safeEmail = email.replace(/'/g, "''")
  return `drop policy if exists "super_admin_insert_first" on public.super_admin;
create policy "super_admin_insert_first" on public.super_admin
for insert to authenticated
with check (id = auth.uid() and not exists (select 1 from public.super_admin));

insert into public.super_admin (id, nome, email, nivel)
select u.id, 'Super Admin', u.email, 'super_admin'
from auth.users u
where u.email = '${safeEmail}'
on conflict (id) do nothing;`
}
```

smagenda/src/views/admin/AdminClienteDetalhesPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { Link, useNavigate, useParams } from 'react-router-dom'
import { AdminShell } from '../../components/layout/AdminShell'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type Cliente = {
  id: string
  nome_completo: string
  nome_negocio: string
  slug: string
  email: string
  telefone: string | null
  plano: string
  status_pagamento: string
  ativo: boolean
  limite_funcionarios: number | null
  limite_agendamentos_mes?: number | null
}

type Funcionario = {
  id: string
  nome_completo: string
  email: string
  telefone: string | null
  permissao: 'admin' | 'funcionario' | 'atendente'
  ativo: boolean
}

type FnResult = { ok: true; status: number; body: unknown } | { ok: false; status: number; body: unknown }

async function callFn(path: string, body: unknown): Promise<FnResult> {
  if (!supabaseEnv.ok) return { ok: false, status: 0, body: { error: 'missing_supabase_env', missing:
supabaseEnv.missing } }

  const { data } = await supabase.auth.getSession()
  const accessToken=[REDACTED]
  if (!accessToken) return { ok: false, status: 401, body: { error: 'missing_session' } }

  const fnUrl = `${supabaseEnv.values.VITE_SUPABASE_URL.replace(/\/$/, '')}/functions/v1/${path}`
  let res: Response
  try {
    res = await fetch(fnUrl, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        apikey=[REDACTED]
        Authorization: `Bearer ${accessToken}`,
      },
      body: JSON.stringify(body),
    })
  } catch (e: unknown) {
    const msg = e instanceof Error ? e.message : 'Falha de rede'
    return { ok: false, status: 0, body: { error: 'network_error', message: msg } }
  }

  const raw = await res.text().catch(() => null)
  const parsed = raw
    ? (() => {
        try {
          return JSON.parse(raw) as unknown
        } catch {
          return raw
        }
      })()
    : null
```

```
  if (!res.ok) return { ok: false, status: res.status, body: parsed }
  return { ok: true, status: res.status, body: parsed }
}

function normalizePlanoToTwoPlans(planoRaw: string) {
  const p = String(planoRaw ?? '').trim().toLowerCase()
  if (p === 'team') return 'pro'
  if (p === 'enterprise') return 'enterprise'
  if (p === 'basic' || p === 'pro' || p === 'free') return p
  return 'basic'
}

function normalizePlanoLabel(planoRaw: string) {
  const p = String(planoRaw ?? '').trim().toLowerCase()
  if (p === 'enterprise') return 'EMPRESA'
  if (p === 'pro' || p === 'team') return 'PRO'
  if (p === 'basic') return 'BASIC'
  if (p === 'free') return 'FREE'
  return planoRaw
}

const planoOptions = ['free', 'basic', 'pro', 'enterprise']
const statusOptions = ['ativo', 'inadimplente', 'cancelado', 'trial', 'suspenso']

const planoDefaultLimite: Record<string, number | null> = {
  free: 1,
  basic: 1,
  pro: 4,
  enterprise: null,
}

export function AdminClienteDetalhesPage() {
  const { id } = useParams()
  const navigate = useNavigate()
  const { startImpersonation } = useAuth()
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [cliente, setCliente] = useState<Cliente | null>(null)
  const [servicosCount, setServicosCount] = useState(0)
  const [funcionariosCount, setFuncionariosCount] = useState(0)
  const [funcionarios, setFuncionarios] = useState<Funcionario[]>([])
  const [saving, setSaving] = useState(false)
  const [plano, setPlano] = useState('')
  const [statusPagamento, setStatusPagamento] = useState('')
  const [ativo, setAtivo] = useState(true)
  const [limiteFuncionarios, setLimiteFuncionarios] = useState('')
  const [limiteAgendamentosMes, setLimiteAgendamentosMes] = useState('')
  const [creatingFuncionario, setCreatingFuncionario] = useState(false)
  const [funcFormOpen, setFuncFormOpen] = useState(false)
  const [funcNome, setFuncNome] = useState('')
  const [funcEmail, setFuncEmail] = useState('')
  const [funcSenha, setFuncSenha] = useState('')
  const [funcPermissao, setFuncPermissao] = useState<'admin' | 'funcionario' | 'atendente'>('funcionario')
  const [checkoutUrl, setCheckoutUrl] = useState<string | null>(null)
  const [creatingCheckout, setCreatingCheckout] = useState(false)
  const [refreshingPagamento, setRefreshingPagamento] = useState(false)
  const [checkoutFuncionariosTotal, setCheckoutFuncionariosTotal] = useState<number>(1)

  const limiteParsed = useMemo(() => {
    const n = limiteFuncionarios.trim() === '' ? null : Number(limiteFuncionarios)
    return Number.isNaN(n as number) ? null : n
  }, [limiteFuncionarios])

  const limiteAgMesParsed = useMemo(() => {
    const n = limiteAgendamentosMes.trim() === '' ? null : Number(limiteAgendamentosMes)
    return Number.isNaN(n as number) ? null : n
  }, [limiteAgendamentosMes])

  const limiteAtingido = useMemo(() => {
    if (!cliente) return false
    if (limiteParsed === null) return false
```

```
    return funcionariosCount >= limiteParsed
  }, [cliente, funcionariosCount, limiteParsed])

  useEffect(() => {
    const run = async () => {
      if (!id) {
        setError('Cliente inválido')
        setLoading(false)
        return
      }
      setLoading(true)
      setError(null)

      const { data, error: err } = await supabase
        .from('usuarios')
.select('id,nome_completo,nome_negocio,slug,email,telefone,plano,status_pagamento,ativo,limite_funcionarios,limite_agendamentos_mes')
        .eq('id', id)
        .maybeSingle()
      if (err) {
        setError(err.message)
        setLoading(false)
        return
      }
      const next = ((data ?? null) as unknown as Cliente | null) ?? null
      setCliente(next)
      if (next) {
        const normalizedPlano = normalizePlanoToTwoPlans(next.plano)
        setPlano(normalizedPlano)
        setStatusPagamento(next.status_pagamento)
        setAtivo(next.ativo)
        setLimiteFuncionarios(next.limite_funcionarios === null ? '' : String(next.limite_funcionarios))
        setLimiteAgendamentosMes(next.limite_agendamentos_mes === null || typeof next.limite_agendamentos_mes !== 'number' ? '' : String(next.limite_agendamentos_mes))

        const n = typeof next.limite_funcionarios === 'number' && Number.isFinite(next.limite_funcionarios) && next.limite_funcionarios > 0 ? Math.floor(next.limite_funcionarios) : 1
        const min = normalizedPlano === 'pro' ? 4 : 1
        const max = normalizedPlano === 'pro' ? 6 : 200
        setCheckoutFuncionariosTotal(Math.max(min, Math.min(max, n)))
      }

      const { count: sCount } = await supabase.from('servicos').select('id', { count: 'exact', head: true }).eq('usuario_id', id)
      const { count: fCount } = await supabase.from('funcionarios').select('id', { count: 'exact', head: true }).eq('usuario_master_id', id)
      setServicosCount(sCount ?? 0)
      setFuncionariosCount(fCount ?? 0)

      const { data: funcs } = await supabase
        .from('funcionarios')
        .select('id,nome_completo,email,telefone,permissao,ativo')
        .eq('usuario_master_id', id)
        .order('nome_completo', { ascending: true })
      setFuncionarios((funcs ?? []) as unknown as Funcionario[])

      setLoading(false)
    }
    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar')
      setLoading(false)
    })
  }, [id])

  const refreshPagamento = async () => {
    if (!id) return
    setRefreshingPagamento(true)
    setError(null)
    const { data, error: err } = await supabase
      .from('usuarios')
      .select('id,plano,status_pagamento,ativo,limite_funcionarios,limite_agendamentos_mes')
```

```
      .eq('id', id)
      .maybeSingle()
    if (err || !data) {
      setError(err?.message ?? 'Erro ao atualizar pagamento')
      setRefreshingPagamento(false)
      return
    }
    const next = data as unknown as Pick<Cliente, 'id' | 'plano' | 'status_pagamento' | 'ativo' | 'limite_funcionarios'
| 'limite_agendamentos_mes'>
    setCliente((prev) => (prev ? { ...prev, ...next } : (next as unknown as Cliente)))
    const normalizedPlano = normalizePlanoToTwoPlans(next.plano)
    setPlano(normalizedPlano)
    setStatusPagamento(next.status_pagamento)
    setAtivo(next.ativo)
    setLimiteFuncionarios(next.limite_funcionarios === null ? '' : String(next.limite_funcionarios))
    setLimiteAgendamentosMes(next.limite_agendamentos_mes === null || typeof next.limite_agendamentos_mes !== 'number' ?
'' : String(next.limite_agendamentos_mes))
    const n = typeof next.limite_funcionarios === 'number' && Number.isFinite(next.limite_funcionarios) &&
next.limite_funcionarios > 0 ? Math.floor(next.limite_funcionarios) : 1
    const min = normalizedPlano === 'pro' ? 4 : 1
    const max = normalizedPlano === 'pro' ? 6 : 200
    setCheckoutFuncionariosTotal(Math.max(min, Math.min(max, n)))
    setRefreshingPagamento(false)
  }

  const save = async () => {
    if (!id) return
    setSaving(true)
    setError(null)
    const limite = limiteFuncionarios.trim() === '' ? null : Number(limiteFuncionarios)
    if (limiteFuncionarios.trim() !== '' && Number.isNaN(limite)) {
      setError('Limite de funcionários inválido')
      setSaving(false)
      return
    }

    const limiteAgMes = limiteAgendamentosMes.trim() === '' ? null : Number(limiteAgendamentosMes)
    if (limiteAgendamentosMes.trim() !== '' && Number.isNaN(limiteAgMes)) {
      setError('Limite de agendamentos/mês inválido')
      setSaving(false)
      return
    }
    const { error: err } = await supabase
      .from('usuarios')
      .update({ plano, status_pagamento: statusPagamento, ativo, limite_funcionarios: limite, limite_agendamentos_mes:
limiteAgMes })
      .eq('id', id)
    if (err) {
      setError(err.message)
      setSaving(false)
      return
    }
    setCliente((prev) =>
      prev ? { ...prev, plano, status_pagamento: statusPagamento, ativo, limite_funcionarios: limite,
limite_agendamentos_mes: limiteAgMes } : prev
    )
    setSaving(false)
  }

  const logarComoCliente = async () => {
    if (!cliente) return
    setError(null)
    setCheckoutUrl(null)
    const res = await startImpersonation(cliente.id)
    if (!res.ok) {
      setError(res.message)
      return
    }
    navigate('/dashboard')
  }

  const createFuncionario = async () => {
```

```
    if (!cliente) return
    const nome = funcNome.trim()
    const email = funcEmail.trim().toLowerCase()
    const senha = funcSenha
    if (!nome || !email || !senha) {
      setError('Preencha nome, email e senha.')
      return
    }
    if (senha.trim().length < 8) {
      setError('A senha deve ter no mínimo 8 caracteres.')
      return
    }
    setCreatingFuncionario(true)
    setError(null)

    const res = await callFn('admin-create-funcionario', {
      usuario_master_id: cliente.id,
      nome_completo: nome,
      email,
      senha,
      permissao: funcPermissao,
    })

    if (!res.ok) {
      const msg = typeof res.body === 'object' && res.body !== null && typeof (res.body as Record<string, 
unknown>).message === 'string'
        ? String((res.body as Record<string, unknown>).message)
        : `Erro ao criar funcionário (HTTP ${res.status}).`
      setError(msg)
      setCreatingFuncionario(false)
      return
    }

    const body = res.body as Record<string, unknown>
    const newId = typeof body.id === 'string' ? body.id : null
    if (!newId) {
      setError('Falha ao criar funcionário.')
      setCreatingFuncionario(false)
      return
    }

    setFuncNome('')
    setFuncEmail('')
    setFuncSenha('')
    setFuncPermissao('funcionario')
    setFuncFormOpen(false)
    setCreatingFuncionario(false)
    await Promise.resolve()
    const { data: funcs } = await supabase
      .from('funcionarios')
      .select('id,nome_completo,email,telefone,permissao,ativo')
      .eq('usuario_master_id', cliente.id)
      .order('nome_completo', { ascending: true })
    setFuncionarios((funcs ?? []) as unknown as Funcionario[])
    setFuncionariosCount((funcs ?? []).length)
  }

  const gerarCheckout = async (metodo: 'card' | 'pix') => {
    if (!cliente) return
    if (!plano) {
      setError('Selecione um plano antes de gerar o link de pagamento.')
      return
    }

    const total = plano === 'pro' ? Math.max(4, Math.min(6, Math.floor(checkoutFuncionariosTotal || 4))) : 1
    setCreatingCheckout(true)
    setError(null)
    setCheckoutUrl(null)

    const res = await callFn('payments', { action: 'create_checkout', usuario_id: cliente.id, plano, metodo, 
funcionarios_total: total })
    if (!res.ok) {
```

```
      const msg = typeof res.body === 'object' && res.body !== null && typeof (res.body as Record<string,
unknown>).message === 'string'
        ? String((res.body as Record<string, unknown>).message)
        : `Erro ao gerar link (HTTP ${res.status}).`
      setError(msg)
      setCreatingCheckout(false)
      return
    }
    const body = res.body as Record<string, unknown>
    const url = typeof body.url === 'string' ? body.url : null
    if (!url) {
      setError('A função não retornou o link de checkout.')
      setCreatingCheckout(false)
      return
    }
    setCheckoutUrl(url)
    setCreatingCheckout(false)
    setTimeout(() => {
      void refreshPagamento()
    }, 4000)
  }

  const toggleFuncionario = async (f: Funcionario) => {
    setError(null)
    const { error: err } = await supabase.from('funcionarios').update({ ativo: !f.ativo }).eq('id', f.id)
    if (err) {
      setError(err.message)
      return
    }
    setFuncionarios((prev) => prev.map((x) => (x.id === f.id ? { ...x, ativo: !x.ativo } : x)))
  }

  const removeFuncionario = async (f: Funcionario) => {
    setError(null)
    const ok = window.confirm(`Excluir funcionário "${f.nome_completo}"?`)
    if (!ok) return
    const { error: err } = await supabase.from('funcionarios').delete().eq('id', f.id)
    if (err) {
      setError(err.message)
      return
    }
    setFuncionarios((prev) => prev.filter((x) => x.id !== f.id))
    setFuncionariosCount((n) => Math.max(0, n - 1))
  }

  return (
    <AdminShell>
      <div className="space-y-6">
        <div className="flex items-center justify-between">
          <div>
            <div className="text-sm font-semibold text-slate-500">Cliente</div>
            <div className="text-xl font-semibold text-slate-900">Detalhes</div>
          </div>
          <Link to="/admin/clientes" className="text-sm font-medium text-slate-900 hover:underline">
            Voltar
          </Link>
        </div>

        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> :
null}
        {loading ? <div className="text-sm text-slate-600">Carregando…</div> : null}

        {cliente ? (
          <div className="space-y-4">
            <Card>
              <div className="p-6 space-y-3">
                <div className="flex flex-wrap items-center justify-between gap-3">
                  <div>
                    <div className="text-lg font-semibold text-slate-900">{cliente.nome_negocio}</div>
                    <div className="text-sm text-slate-600">/{cliente.slug}</div>
                  </div>
                  <div className="flex items-center gap-2">
```

```
                    {cliente.ativo ? <Badge tone="green">Conta: Ativa</Badge> : <Badge tone="red">Conta: Inativa</Badge>}
                    <Badge tone="slate">{normalizePlanoLabel(cliente.plano)}</Badge>
                    <Badge tone={cliente.status_pagamento === 'inadimplente' ? 'red' : cliente.status_pagamento === 'ativo' ? 'green' : 'slate'}>
                      Pagamento: {(() => {
                        const v = String(cliente.status_pagamento ?? '').trim().toLowerCase()
                        if (!v) return '—'
                        if (v === 'ativo') return 'Ativo'
                        if (v === 'trial') return 'Trial'
                        if (v === 'inadimplente') return 'Inadimplente'
                        if (v === 'suspenso') return 'Suspenso'
                        if (v === 'cancelado') return 'Cancelado'
                        return cliente.status_pagamento
                      })()}
                    </Badge>
                  </div>
                </div>
                <div className="text-sm text-slate-700">{cliente.nome_completo}</div>
                <div className="text-sm text-slate-700">{cliente.email}</div>
                {cliente.telefone ? <div className="text-sm text-slate-700">{cliente.telefone}</div> : null}
                <div className="text-sm text-slate-700">Serviços: {servicosCount}</div>
                <div className="text-sm text-slate-700">Funcionários: {funcionariosCount}</div>
                <div className="flex justify-end">
                  <Button onClick={logarComoCliente}>Logar como cliente</Button>
                </div>
              </div>
            </Card>

            <Card>
              <div className="p-6 space-y-4">
                <div className="text-sm font-semibold text-slate-900">Gerenciamento</div>

                <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                  <label className="block">
                    <div className="text-sm font-medium text-slate-700 mb-1">Plano</div>
                    <select
                      className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900 outline-none focus:ring-2 focus:ring-slate-300"
                      value={plano}
                      onChange={(e) => {
                        const nextPlano = e.target.value
                        setPlano(nextPlano)
                        const nextLimite = Object.prototype.hasOwnProperty.call(planoDefaultLimite, nextPlano)
                          ? planoDefaultLimite[nextPlano]
                          : null
                        setLimiteFuncionarios(nextLimite === null ? '' : String(nextLimite))
                        setCheckoutFuncionariosTotal(nextPlano === 'pro' ? 4 : 1)
                      }}
                    >
                      {planoOptions.map((p) => (
                        <option key={p} value={p}>
                          {normalizePlanoLabel(p)}
                        </option>
                      ))}
                    </select>
                  </label>

                  <label className="block">
                    <div className="text-sm font-medium text-slate-700 mb-1">Status de pagamento</div>
                    <select
                      className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900 outline-none focus:ring-2 focus:ring-slate-300"
                      value={statusPagamento}
                      onChange={(e) => setStatusPagamento(e.target.value)}
                    >
                      {statusOptions.map((s) => (
                        <option key={s} value={s}>
                          {s}
                        </option>
                      ))}
                    </select>
```

```
                  </label>
                </div>

                <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                  <label className="block">
                    <div className="text-sm font-medium text-slate-700 mb-1">Conta ativa</div>
                    <select
                      className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900 outline-none focus:ring-2 focus:ring-slate-300"
                      value={ativo ? 'true' : 'false'}
                      onChange={(e) => setAtivo(e.target.value === 'true')}
                    >
                      <option value="true">Ativa</option>
                      <option value="false">Inativa</option>
                    </select>
                  </label>
                  <Input
                    label="Limite de funcionários (vazio = sem limite)"
                    value={limiteFuncionarios}
                    onChange={(e) => setLimiteFuncionarios(e.target.value)}
                    inputMode="numeric"
                  />
                </div>

                <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                  <Input
                    label="Limite de agendamentos/mês (vazio = sem limite)"
                    value={limiteAgendamentosMes}
                    onChange={(e) => setLimiteAgendamentosMes(e.target.value)}
                    inputMode="numeric"
                  />
                  <div className="flex items-end justify-end gap-2">
                    <Button
                      variant="secondary"
                      onClick={() => {
                        const base = limiteAgMesParsed === null ? 0 : Math.max(0, Math.floor(limiteAgMesParsed))
                        setLimiteAgendamentosMes(String(base + 50))
                      }}
                    >
                      +50
                    </Button>
                    <Button
                      variant="secondary"
                      onClick={() => {
                        const base = limiteAgMesParsed === null ? 0 : Math.max(0, Math.floor(limiteAgMesParsed))
                        setLimiteAgendamentosMes(String(base + 100))
                      }}
                    >
                      +100
                    </Button>
                    <Button
                      variant="secondary"
                      onClick={() => {
                        setLimiteAgendamentosMes('')
                      }}
                    >
                      Ilimitado
                    </Button>
                  </div>
                </div>

                {limiteFuncionarios.trim() !== '' && limiteParsed === null ? (
                  <div className="text-sm text-rose-700">Limite de funcionários inválido.</div>
                ) : null}

                {limiteAgendamentosMes.trim() !== '' && limiteAgMesParsed === null ? (
                  <div className="text-sm text-rose-700">Limite de agendamentos/mês inválido.</div>
                ) : null}

                <div className="flex justify-end">
                  <Button onClick={save} disabled={saving || !plano || !statusPagamento}>
                    Salvar alterações
```

```
                  </Button>
                </div>

                <div className="border-t border-slate-100 pt-4 space-y-3">
                  <div className="text-sm font-semibold text-slate-900">Pagamento</div>
                  <div className="flex flex-wrap items-center justify-between gap-3">
                    <div className="text-sm text-slate-600">Gera um link de checkout para o plano selecionado.</div>
                    <div className="flex flex-wrap items-center gap-2">
                      <Input
                        label="Profissionais"
                        type="number"
                        min={4}
                        max={6}
                        value={String(checkoutFuncionariosTotal)}
                        disabled={plano !== 'pro'}
                        onChange={(e) => {
                          const raw = e.target.value
                          const n = raw.trim() === '' ? 4 : Number(raw)
                          if (!Number.isFinite(n)) return
                          const i = Math.floor(n)
                          if (i > 6) {
                            setPlano('enterprise')
                            setLimiteFuncionarios('')
                            setCheckoutFuncionariosTotal(1)
                            setError('Para mais de 6 profissionais, use o plano EMPRESA.')
                            return
                          }
                          const clamped = Math.max(4, Math.min(6, i))
                          setCheckoutFuncionariosTotal(clamped)
                        }}
                        className="w-28"
                      />
                      <Button variant="secondary" onClick={() => void refreshPagamento()} disabled={refreshingPagamento || !cliente}>
                        {refreshingPagamento ? 'Atualizando…' : 'Atualizar status'}
                      </Button>
                      <Button variant="secondary" onClick={() => void gerarCheckout('pix')} disabled={creatingCheckout || !plano}>
                        {creatingCheckout ? 'Gerando…' : 'PIX (30 dias)'}
                      </Button>
                      <Button variant="secondary" onClick={() => void gerarCheckout('card')} disabled={creatingCheckout || !plano}>
                        {creatingCheckout ? 'Gerando…' : 'Cartão (assinatura)'}
                      </Button>
                    </div>
                  </div>
                  {checkoutUrl ? (
                    <div className="rounded-xl border border-slate-200 bg-white p-4 space-y-3">
                      <div className="text-xs font-semibold tracking-wide text-slate-500">CHECKOUT URL</div>
                      <a className="block break-all text-sm text-slate-900 hover:underline" href={checkoutUrl} target="_blank" rel="noreferrer">
                        {checkoutUrl}
                      </a>
                      <div className="flex flex-wrap justify-end gap-2">
                        <Button
                          variant="secondary"
                          onClick={async () => {
                            try {
                              await navigator.clipboard.writeText(checkoutUrl)
                            } catch {
                              window.prompt('Copie o link:', checkoutUrl)
                            }
                          }}
                        >
                          Copiar
                        </Button>
                        <Button
                          variant="secondary"
                          onClick={() => {
                            window.open(checkoutUrl, '_blank', 'noreferrer')
                          }}
                        >
```

```
                          Abrir
                        </Button>
                      </div>
                    </div>
                  ) : null}
                </div>
              </div>
            </Card>

            <Card>
              <div className="p-6 space-y-4">
                <div className="flex flex-wrap items-center justify-between gap-3">
                  <div className="text-sm font-semibold text-slate-900">Funcionários</div>
                  <Button
                    variant="secondary"
                    onClick={() => {
                      setError(null)
                      setCheckoutUrl(null)
                      setFuncFormOpen((v) => !v)
                    }}
                    disabled={creatingFuncionario || limiteAtingido}
                  >
                    {funcFormOpen ? 'Fechar' : 'Novo funcionário'}
                  </Button>
                </div>

                {limiteAtingido ? (
                  <div className="text-sm text-rose-700">Limite de funcionários atingido para o plano.</div>
                ) : null}

                {funcFormOpen ? (
                  <div className="rounded-xl border border-slate-200 bg-slate-50 p-4 space-y-4">
                    <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                      <Input label="Nome completo" value={funcNome} onChange={(e) => setFuncNome(e.target.value)} />
                      <Input label="Email" value={funcEmail} onChange={(e) => setFuncEmail(e.target.value)} />
                      <Input label="Senha" type="password" value={funcSenha} onChange={(e) =>
setFuncSenha(e.target.value)} />
                      <label className="block">
                        <div className="text-sm font-medium text-slate-700 mb-1">Permissão</div>
                        <select
                          className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                          value={funcPermissao}
                          onChange={(e) => setFuncPermissao(e.target.value as 'admin' | 'funcionario' | 'atendente')}
                        >
                          <option value="funcionario">Funcionário</option>
                          <option value="atendente">Atendente</option>
                          <option value="admin">Gerente</option>
                        </select>
                      </label>
                    </div>

                    <div className="flex flex-wrap justify-end gap-2">
                      <Button
                        variant="secondary"
                        onClick={() => {
                          setFuncFormOpen(false)
                        }}
                        disabled={creatingFuncionario}
                      >
                        Cancelar
                      </Button>
                      <Button onClick={createFuncionario} disabled={creatingFuncionario}>
                        {creatingFuncionario ? 'Criando…' : 'Criar funcionário'}
                      </Button>
                    </div>
                  </div>
                ) : null}

                {funcionarios.length === 0 ? <div className="text-sm text-slate-600">Nenhum funcionário.</div> : null}
                <div className="divide-y divide-slate-100">
                  {funcionarios.map((f) => (
```

```
                    <div key={f.id} className="py-3 flex items-start justify-between gap-3">
                      <div>
                        <div className="text-sm font-semibold text-slate-900">{f.nome_completo}</div>
                        <div className="text-sm text-slate-600">{f.email}</div>
                        {f.telefone ? <div className="text-sm text-slate-600">{f.telefone}</div> : null}
                        <div className="text-sm text-slate-700">{f.permissao === 'admin' ? 'Gerente' : f.permissao === 'atendente' ? 'Atendente' : 'Funcionário'}</div>
                      </div>
                      <div className="flex flex-wrap items-center justify-end gap-2">
                        {f.ativo ? <Badge tone="green">Ativo</Badge> : <Badge tone="red">Inativo</Badge>}
                        <Button variant="secondary" onClick={() => toggleFuncionario(f)}>
                          {f.ativo ? 'Desativar' : 'Ativar'}
                        </Button>
                        <Button variant="danger" onClick={() => removeFuncionario(f)}>
                          Excluir
                        </Button>
                      </div>
                    </div>
                  ))}
                </div>
              </div>
            </Card>
          </div>
        ) : null}
      </div>
    </AdminShell>
  )
}
```

smagenda/src/views/admin/AdminClientesPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { Link } from 'react-router-dom'
import { AdminShell } from '../../components/layout/AdminShell'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { supabase } from '../../lib/supabase'

type Cliente = {
  id: string
  nome_negocio: string
  slug: string
  plano: string
  status_pagamento: string
  ativo: boolean
  tema_prospector_habilitado?: boolean | null
}

function normalizePlanoLabel(planoRaw: string) {
  const p = String(planoRaw ?? '').trim().toLowerCase()
  if (p === 'enterprise') return 'EMPRESA'
  if (p === 'pro' || p === 'team') return 'PRO'
  if (p === 'basic') return 'BASIC'
  if (p === 'free') return 'FREE'
  return planoRaw
}

export function AdminClientesPage() {
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [clientes, setClientes] = useState<Cliente[]>([])

  const [schemaTemaProspectorIncompleto, setSchemaTemaProspectorIncompleto] = useState(false)
  const [updatingTemaProspector, setUpdatingTemaProspector] = useState(false)
  const [selected, setSelected] = useState<Record<string, boolean>>({})

  const [query, setQuery] = useState('')
  const [plano, setPlano] = useState('')
  const [statusPagamento, setStatusPagamento] = useState('')
  const [ativo, setAtivo] = useState('')

  const canClear = useMemo(() => Boolean(query.trim() || plano || statusPagamento || ativo), [ativo, plano, query,
statusPagamento])

  const filteredClientes = useMemo(() => clientes, [clientes])
  const selectedIds = useMemo(() => Object.keys(selected).filter((id) => selected[id] === true), [selected])
  const selectedCount = selectedIds.length

  const clearSelection = () => setSelected({})
  const selectAllFiltered = () => setSelected(Object.fromEntries(filteredClientes.map((c) => [c.id, true])))

  useEffect(() => {
    const run = async () => {
      setLoading(true)
      setError(null)
      setSchemaTemaProspectorIncompleto(false)

      const q = query.trim()
      let req = supabase
        .from('usuarios')
        .select('id,nome_negocio,slug,email,telefone,plano,status_pagamento,ativo,tema_prospector_habilitado')
        .order('criado_em', { ascending: false })
        .limit(500)

      if (q) {
        const safe = q.replaceAll(',', ' ').trim()
        req = req.or(`nome_negocio.ilike.%${safe}%,slug.ilike.%${safe}%,email.ilike.%${safe}%,telefone.ilike.%${safe}%`)
      }
      if (plano) {
```

```
        if (plano === 'pro') {
          req = req.in('plano', ['pro', 'team'])
        } else {
          req = req.eq('plano', plano)
        }
      }
      if (statusPagamento) req = req.eq('status_pagamento', statusPagamento)
      if (ativo) req = req.eq('ativo', ativo === 'true')

      const { data, error: err } = await req
      if (err) {
        const msg = err.message
        if (msg.toLowerCase().includes('does not exist') && msg.toLowerCase().includes('column')) {
          setSchemaTemaProspectorIncompleto(true)
          let fallback = supabase
            .from('usuarios')
            .select('id,nome_negocio,slug,email,telefone,plano,status_pagamento,ativo')
            .order('criado_em', { ascending: false })
            .limit(500)

          if (q) {
            const safe = q.replaceAll(',', ' ').trim()
            fallback =
fallback.or(`nome_negocio.ilike.%${safe}%,slug.ilike.%${safe}%,email.ilike.%${safe}%,telefone.ilike.%${safe}%`)
          }
          if (plano) {
            if (plano === 'pro') {
              fallback = fallback.in('plano', ['pro', 'team'])
            } else {
              fallback = fallback.eq('plano', plano)
            }
          }
          if (statusPagamento) fallback = fallback.eq('status_pagamento', statusPagamento)
          if (ativo) fallback = fallback.eq('ativo', ativo === 'true')

          const { data: data2, error: err2 } = await fallback
          if (err2) {
            setError(err2.message)
            setLoading(false)
            return
          }
          setClientes((data2 ?? []) as unknown as Cliente[])
          setLoading(false)
          return
        }

        setError(msg)
        setLoading(false)
        return
      }
      setClientes((data ?? []) as unknown as Cliente[])
      setLoading(false)
    }
    const t = window.setTimeout(() => {
      run().catch((e: unknown) => {
        setError(e instanceof Error ? e.message : 'Erro ao carregar')
        setLoading(false)
      })
    }, 250)
    return () => window.clearTimeout(t)
  }, [ativo, plano, query, statusPagamento])

  const enableTemaProspectorSelected = async (enabled: boolean) => {
    if (selectedCount === 0) return
    setUpdatingTemaProspector(true)
    setError(null)
    const patch: Record<string, unknown> = enabled
      ? {
          tema_prospector_habilitado: true,
        }
      : {
          tema_prospector_habilitado: false,
```

```
        }

    const { error: err } = await supabase.from('usuarios').update(patch).in('id', selectedIds)
    if (err) {
      const msg = err.message
      if (msg.toLowerCase().includes('does not exist') && msg.toLowerCase().includes('column')) {
        setError('Seu Supabase não tem a coluna de habilitação do Tema Prospector. Em /admin/configuracoes, execute o
bloco “SQL do Tema Prospector (habilitação por cliente)”.')
      } else {
        setError(msg)
      }
      setUpdatingTemaProspector(false)
      return
    }

    setClientes((prev) => prev.map((c) => (selectedIds.includes(c.id) ? { ...c, tema_prospector_habilitado: enabled } :
c)))
    setUpdatingTemaProspector(false)
  }

  return (
    <AdminShell>
      <div className="space-y-6">
        <Card>
          <div className="p-6 space-y-4">
            <div className="text-sm font-semibold text-slate-900">Filtros</div>
            <div className="grid grid-cols-1 gap-4 sm:grid-cols-2 lg:grid-cols-4">
              <Input label="Busca" value={query} onChange={(e) => setQuery(e.target.value)} placeholder="Nome, /slug,
email, telefone" />
              <label className="block">
                <div className="text-sm font-medium text-slate-700 mb-1">Plano</div>
                <select
                  className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                  value={plano}
                  onChange={(e) => setPlano(e.target.value)}
                >
                  <option value="">Todos</option>
                  <option value="free">FREE</option>
                  <option value="basic">BASIC</option>
                  <option value="pro">PRO</option>
                  <option value="enterprise">EMPRESA</option>
                </select>
              </label>
              <label className="block">
                <div className="text-sm font-medium text-slate-700 mb-1">Status pagamento</div>
                <select
                  className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                  value={statusPagamento}
                  onChange={(e) => setStatusPagamento(e.target.value)}
                >
                  <option value="">Todos</option>
                  <option value="ativo">ativo</option>
                  <option value="trial">trial</option>
                  <option value="inadimplente">inadimplente</option>
                  <option value="suspenso">suspenso</option>
                  <option value="cancelado">cancelado</option>
                </select>
              </label>
              <label className="block">
                <div className="text-sm font-medium text-slate-700 mb-1">Ativo</div>
                <select
                  className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                  value={ativo}
                  onChange={(e) => setAtivo(e.target.value)}
                >
                  <option value="">Todos</option>
                  <option value="true">Ativo</option>
                  <option value="false">Inativo</option>
                </select>
```

```
              </label>
            </div>
            <div className="flex justify-end">
              <Button
                variant="secondary"
                onClick={() => {
                  setQuery('')
                  setPlano('')
                  setStatusPagamento('')
                  setAtivo('')
                }}
                disabled={!canClear}
              >
                Limpar
              </Button>
            </div>
          </div>
        </Card>

        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> :
null}

        {schemaTemaProspectorIncompleto ? (
          <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-800">
            Seu Supabase ainda não tem a coluna do Tema Prospector por cliente. Execute o bloco “SQL do Tema Prospector
(habilitação por cliente)” em{' '}
            <span className="font-mono text-xs">/admin/configuracoes</span>.
          </div>
        ) : null}

        {!loading && !error && clientes.length === 0 ? (
          <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-800">
            Nenhum cliente retornado.
            <div className="mt-1 text-amber-900">
              Se você já possui clientes no Supabase, normalmente é falta de políticas (RLS) na tabela
              <span className="font-mono text-xs"> public.usuarios</span> ou o cliente existe apenas no Auth e não na
tabela
              <span className="font-mono text-xs"> usuarios</span>.
            </div>
            <div className="mt-2">
              Acesse <span className="font-mono text-xs">/admin/configuracoes</span> e execute o bloco “SQL de políticas
(Super Admin)” no SQL Editor do Supabase.
            </div>
          </div>
        ) : null}
        <Card>
          <div className="divide-y divide-slate-100">
            <div className="p-4 flex flex-wrap items-center justify-between gap-3">
              <div className="text-sm text-slate-600">Selecionados: {selectedCount}</div>
              <div className="flex flex-wrap items-center gap-2">
                <Button onClick={() => void enableTemaProspectorSelected(true)} disabled={selectedCount === 0 ||
updatingTemaProspector || schemaTemaProspectorIncompleto}>
                  Habilitar Tema Prospector
                </Button>
                <Button
                  variant="secondary"
                  onClick={() => void enableTemaProspectorSelected(false)}
                  disabled={selectedCount === 0 || updatingTemaProspector || schemaTemaProspectorIncompleto}
                >
                  Desabilitar Tema Prospector
                </Button>
                <Button variant="secondary" onClick={selectAllFiltered} disabled={loading || filteredClientes.length ===
0}>
                  Selecionar filtrados
                </Button>
                <Button variant="secondary" onClick={clearSelection} disabled={selectedCount === 0}>
                  Limpar seleção
                </Button>
              </div>
            </div>
            {loading ? (
              <div className="p-6 text-sm text-slate-600">Carregando…</div>
```

```
      ) : clientes.length === 0 ? (
       <div className="p-6 text-sm text-slate-600">Nenhum cliente.</div>
      ) : (
       clientes.map((c) => (
        <div key={c.id} className="p-4 hover:bg-slate-50">
         <div className="flex items-start justify-between gap-3">
          <div className="flex items-start gap-3 min-w-0">
           <input
            type="checkbox"
            checked={selected[c.id] === true}
            onChange={(e) => setSelected((prev) => ({ ...prev, [c.id]: e.target.checked }))}
            className="mt-1"
           />
           <div className="min-w-0">
            <Link to={`/admin/clientes/${c.id}`} className="block min-w-0">
             <div className="text-sm font-semibold text-slate-900 truncate">{c.nome_negocio}</div>
             <div className="text-sm text-slate-600 truncate">/{c.slug}</div>
            </Link>
           </div>
          </div>
          <div className="flex items-center gap-2">
           {c.tema_prospector_habilitado === true ? <Badge tone="slate">Tema Prospector</Badge> : null}
           {c.ativo ? <Badge tone="green">Ativo</Badge> : <Badge tone="red">Inativo</Badge>}
           <Badge tone="slate">{normalizePlanoLabel(c.plano)}</Badge>
           <Badge tone={c.status_pagamento === 'inadimplente' ? 'red' : 'slate'}>{c.status_pagamento}</Badge>
          </div>
         </div>
        </div>
       ))
      )}
     </div>
    </Card>
   </div>
  </AdminShell>
 )
}
```

smagenda/src/views/admin/AdminConfiguracoesPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { AdminShell } from '../../components/layout/AdminShell'
import { Card } from '../../components/ui/Card'
import { Button } from '../../components/ui/Button'
import { Input } from '../../components/ui/Input'
import { supabase, supabaseEnv } from '../../lib/supabase'

function SqlCard({ title, description, sql, defaultOpen }: { title: string; description: string; sql: string;
defaultOpen?: boolean }) {
  const [open, setOpen] = useState(Boolean(defaultOpen))
  return (
    <Card>
      <div className="p-6 space-y-3">
        <div className="flex items-start justify-between gap-3">
          <div className="space-y-1">
            <div className="text-sm font-semibold text-slate-900">{title}</div>
            <div className="text-sm text-slate-600">{description}</div>
          </div>
          <Button variant="secondary" onClick={() => setOpen((v) => !v)}>
            {open ? 'Ocultar' : 'Mostrar'}
          </Button>
        </div>

        {open ? (
          <pre className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-xs text-slate-800 overflow-auto">{sql}
</pre>
        ) : (
          <div className="rounded-xl border border-slate-200 bg-white p-3 text-xs text-slate-600">
            Clique em “Mostrar” para ver o SQL.
          </div>
        )}
      </div>
    </Card>
  )
}

export function AdminConfiguracoesPage() {
  const [userId, setUserId] = useState<string | null>(null)
  const [email, setEmail] = useState<string | null>(null)

  const supabaseProjectUrl = supabaseEnv.ok
    ? supabaseEnv.values.VITE_SUPABASE_URL.replace(/\/$/, '')
    : 'https://<PROJECT_REF>.supabase.co'
  const supabaseAnonKey = supabaseEnv.ok ? supabaseEnv.values.VITE_SUPABASE_ANON_KEY : '<SUPABASE_ANON_KEY>'

  const [resendDomain, setResendDomain] = useState('')
  const [resendLoading, setResendLoading] = useState(false)
  const [resendError, setResendError] = useState<string | null>(null)
  const [resendResult, setResendResult] = useState<
    | null
    | {
        domain?: { id?: string | null; name?: string | null; status?: string | null; region?: string | null }
        dns?: {
          checked_at?: string | null
          nameservers?: string[]
          records?: Array<{
            record?: string | null
            name?: string
            type?: string
            fqdn?: string | null
            expected?: { value?: string | null; priority?: number | null }
            dns?: { found?: boolean; match?: boolean; values?: string[] }
            hint?: string | null
            resendStatus?: string | null
          }>
        }
        error?: string
        message?: string
      }
  >(null)
```

```
  const [resendTestFrom, setResendTestFrom] = useState('')
  const [resendTestTo, setResendTestTo] = useState('')
  const [resendTestSubject, setResendTestSubject] = useState('')
  const [resendTestSending, setResendTestSending] = useState(false)
  const [resendTestError, setResendTestError] = useState<string | null>(null)
  const [resendTestResult, setResendTestResult] = useState<unknown>(null)

  useEffect(() => {
    const run = async () => {
      const { data } = await supabase.auth.getSession()
      setUserId(data.session?.user?.id ?? null)
      const sessionEmail = data.session?.user?.email ?? null
      setEmail(sessionEmail)
      if (sessionEmail && !resendTestTo) setResendTestTo(sessionEmail)

      const host = window.location.hostname
      if (host && host !== 'localhost' && host !== '127.0.0.1' && host.includes('.') && !resendDomain) {
        setResendDomain(host)
      }
      if (host && host !== 'localhost' && host !== '127.0.0.1' && host.includes('.') && !resendTestFrom) {
        setResendTestFrom(`SMagenda <no-reply@${host}>`)
      }
    }
    run().catch(() => undefined)
  }, [resendDomain, resendTestFrom, resendTestTo])

  const sendResendTest = async () => {
    setResendTestSending(true)
    setResendTestError(null)
    setResendTestResult(null)

    if (!supabaseEnv.ok) {
      setResendTestError(`Supabase não configurado. Faltando: ${supabaseEnv.missing.join(', ')}`)
      setResendTestSending(false)
      return
    }

    const domain = resendDomain.trim().toLowerCase()
    if (!domain) {
      setResendTestError('Informe o domínio (ex.: smagenda.com.br).')
      setResendTestSending(false)
      return
    }

    const from = resendTestFrom.trim()
    const to = resendTestTo.trim()
    if (!from || !to) {
      setResendTestError('Informe remetente e destinatário.')
      setResendTestSending(false)
      return
    }

    const { data } = await supabase.auth.getSession()
    const accessToken=[REDACTED]
    if (!accessToken) {
      setResendTestError('Sessão expirada. Faça login novamente.')
      setResendTestSending(false)
      return
    }

    const fnUrl = `${supabaseEnv.values.VITE_SUPABASE_URL.replace(/\/$/, '')}/functions/v1/resend-domain`
    const subject = resendTestSubject.trim() || 'Teste de email (Resend) — SMagenda'
    const text = `Teste real de envio via Resend em ${new Date().toISOString()}`

    try {
      const res = await fetch(fnUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          apikey=[REDACTED]
          Authorization: `Bearer ${accessToken}`,
```

```
        },
        body: JSON.stringify({ domain, action: 'send_test', from, to, subject, text }),
      })
      const json = (await res.json().catch(() => null)) as unknown
      if (!res.ok) {
        const obj = json && typeof json === 'object' ? (json as Record<string, unknown>) : null
        const msg = (typeof obj?.message === 'string' && obj.message) || (typeof obj?.error === 'string' && obj.error)
|| `HTTP ${res.status}`
        setResendTestError(String(msg))
        setResendTestResult(json)
        setResendTestSending(false)
        return
      }

      setResendTestResult(json)
      setResendTestSending(false)
    } catch (e: unknown) {
      setResendTestError(e instanceof Error ? e.message : 'Falha de rede ao enviar email')
      setResendTestSending(false)
    }
  }

  const runResend = async (action: 'status' | 'verify') => {
    setResendLoading(true)
    setResendError(null)
    setResendResult(null)

    if (!supabaseEnv.ok) {
      setResendError(`Supabase não configurado. Faltando: ${supabaseEnv.missing.join(', ')}`)
      setResendLoading(false)
      return
    }

    const domain = resendDomain.trim().toLowerCase()
    if (!domain) {
      setResendError('Informe o domínio (ex.: smagenda.com.br).')
      setResendLoading(false)
      return
    }

    const { data } = await supabase.auth.getSession()
    const accessToken=[REDACTED]
    if (!accessToken) {
      setResendError('Sessão expirada. Faça login novamente.')
      setResendLoading(false)
      return
    }

    const fnUrl = `${supabaseEnv.values.VITE_SUPABASE_URL.replace(/\/$/, '')}/functions/v1/resend-domain`

    try {
      const res = await fetch(fnUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          apikey=[REDACTED]
          Authorization: `Bearer ${accessToken}`,
        },
        body: JSON.stringify({ domain, action }),
      })
      const json = (await res.json().catch(() => null)) as typeof resendResult
      if (!res.ok) {
        const msg = (json as { message?: string | null })?.message ?? (json as { error?: string | null })?.error ??
`HTTP ${res.status}`
        setResendError(String(msg))
        setResendResult(json)
        setResendLoading(false)
        return
      }

      setResendResult(json)
      setResendLoading(false)
```

```
      } catch (e: unknown) {
        setResendError(e instanceof Error ? e.message : 'Falha de rede ao checar DNS')
        setResendLoading(false)
      }
    }

    const sql = useMemo(() => {
      return `create or replace function public.is_super_admin() returns boolean
  language sql stable as $$
    select exists (select 1 from public.super_admin sa where sa.id = auth.uid())
  $$;

  alter table public.usuarios enable row level security;
  alter table public.funcionarios enable row level security;
  alter table public.servicos enable row level security;
  alter table public.agendamentos enable row level security;
  alter table public.bloqueios enable row level security;

  drop policy if exists "super_admin_full_usuarios" on public.usuarios;
  create policy "super_admin_full_usuarios" on public.usuarios
  for all to authenticated
  using (public.is_super_admin())
  with check (public.is_super_admin());

  drop policy if exists "super_admin_full_funcionarios" on public.funcionarios;
  create policy "super_admin_full_funcionarios" on public.funcionarios
  for all to authenticated
  using (public.is_super_admin())
  with check (public.is_super_admin());

  drop policy if exists "super_admin_full_servicos" on public.servicos;
  create policy "super_admin_full_servicos" on public.servicos
  for all to authenticated
  using (public.is_super_admin())
  with check (public.is_super_admin());

  drop policy if exists "super_admin_full_agendamentos" on public.agendamentos;
  create policy "super_admin_full_agendamentos" on public.agendamentos
  for all to authenticated
  using (public.is_super_admin())
  with check (public.is_super_admin());

  drop policy if exists "super_admin_full_bloqueios" on public.bloqueios;
  create policy "super_admin_full_bloqueios" on public.bloqueios
  for all to authenticated
  using (public.is_super_admin())
  with check (public.is_super_admin());`
    }, [])

    const sqlTrial = useMemo(() => {
      return `alter table public.usuarios add column if not exists data_vencimento date;
  alter table public.usuarios add column if not exists termos_aceitos_em timestamptz;
  alter table public.usuarios add column if not exists privacidade_aceita_em timestamptz;
  alter table public.usuarios add column if not exists termos_versao text;
  alter table public.usuarios add column if not exists privacidade_versao text;
  alter table public.usuarios add column if not exists consentimento_ip text;
  alter table public.usuarios add column if not exists consentimento_user_agent text;

  create or replace function public.ensure_usuario_profile()
  returns void
  language plpgsql
  security definer
  set search_path = public
  as $$
  declare
    uid uuid := auth.uid();
    jwt jsonb := auth.jwt();
    meta jsonb := coalesce(jwt -> 'user_metadata', '{}'::jsonb);
    v_nome text := coalesce(nullif(meta ->> 'nome_completo', ''), null);
    v_negocio text := coalesce(nullif(meta ->> 'nome_negocio', ''), null);
    v_slug text := coalesce(nullif(meta ->> 'slug', ''), null);
    v_telefone text := nullif(meta ->> 'telefone', '');
```

```
  v_email text := nullif(jwt ->> 'email', '');
  v_terms_ver text := nullif(meta ->> 'termos_versao', '');
  v_priv_ver text := nullif(meta ->> 'privacidade_versao', '');
  v_accepted_raw text := nullif(meta ->> 'legal_aceite_em', '');
  v_accepted_at timestamptz := null;
  v_ua text := nullif(meta ->> 'legal_user_agent', '');
begin
  if uid is null then
    raise exception 'not authenticated';
  end if;

  if nullif(meta ->> 'usuario_master_id', '') is not null then
    return;
  end if;

  if exists (select 1 from information_schema.tables where table_schema = 'public' and table_name = 'funcionarios') then
    if exists (select 1 from public.funcionarios f where f.id = uid) then
      return;
    end if;
  end if;

  if exists (select 1 from information_schema.tables where table_schema = 'public' and table_name = 'super_admin') then
    if exists (select 1 from public.super_admin sa where sa.id = uid) then
      return;
    end if;
  end if;

  if v_email is null then
    begin
      select u.email into v_email
      from auth.users u
      where u.id = uid;
    exception
      when others then
        v_email := null;
    end;
  end if;

  if v_email is null then
    return;
  end if;

  if v_nome is null then
    v_nome := nullif(split_part(v_email, '@', 1), '');
  end if;

  if v_negocio is null then
    v_negocio := v_nome;
  end if;

  if v_slug is null then
    v_slug := nullif(regexp_replace(lower(split_part(v_email, '@', 1)), '[^a-z0-9]+', '-', 'g'), '');
  end if;

  if v_slug is null then
    v_slug := 'u-' || substring(replace(uid::text, '-', '') from 1 for 8);
  end if;

  if exists (select 1 from public.usuarios u where u.slug = v_slug and u.id <> uid) then
    v_slug := v_slug || '-' || substring(replace(uid::text, '-', '') from 1 for 6);
  end if;

  if v_accepted_raw is not null then
    begin
      v_accepted_at := v_accepted_raw::timestamptz;
    exception
      when others then
        v_accepted_at := null;
    end;
  end if;

  insert into public.usuarios (
```

```
    id, nome_completo, nome_negocio, telefone, email, slug,
    plano, tipo_conta, limite_funcionarios, status_pagamento, data_vencimento, ativo,
    termos_aceitos_em, privacidade_aceita_em, termos_versao, privacidade_versao, consentimento_user_agent
  )
  values (
    uid, v_nome, v_negocio, v_telefone, v_email, v_slug,
    'free', 'master', 1, 'trial', (current_date + 7), true,
    v_accepted_at, v_accepted_at, v_terms_ver, v_priv_ver, v_ua
  )
  on conflict (id) do update
  set
    nome_completo = coalesce(public.usuarios.nome_completo, excluded.nome_completo),
    nome_negocio = coalesce(public.usuarios.nome_negocio, excluded.nome_negocio),
    telefone = coalesce(public.usuarios.telefone, excluded.telefone),
    email = coalesce(public.usuarios.email, excluded.email),
    slug = coalesce(public.usuarios.slug, excluded.slug),
    limite_funcionarios = coalesce(public.usuarios.limite_funcionarios, excluded.limite_funcionarios),
    data_vencimento = coalesce(public.usuarios.data_vencimento, excluded.data_vencimento),
    termos_aceitos_em = coalesce(public.usuarios.termos_aceitos_em, excluded.termos_aceitos_em),
    privacidade_aceita_em = coalesce(public.usuarios.privacidade_aceita_em, excluded.privacidade_aceita_em),
    termos_versao = coalesce(public.usuarios.termos_versao, excluded.termos_versao),
    privacidade_versao = coalesce(public.usuarios.privacidade_versao, excluded.privacidade_versao),
    consentimento_user_agent = coalesce(public.usuarios.consentimento_user_agent, excluded.consentimento_user_agent);
end;
$$;

revoke all on function public.ensure_usuario_profile() from public;
grant execute on function public.ensure_usuario_profile() to authenticated;

create or replace function public.accept_legal_terms(
  p_terms_version text,
  p_privacy_version text
) returns void
language plpgsql
security definer
set search_path = public
as $$
declare
  v_uid uuid := auth.uid();
  v_headers jsonb := coalesce(current_setting('request.headers', true)::jsonb, '{}'::jsonb);
  v_xff text := nullif(v_headers ->> 'x-forwarded-for', '');
  v_ip text := nullif(split_part(coalesce(v_xff, ''), ',', 1), '');
  v_ua text := nullif(v_headers ->> 'user-agent', '');
begin
  if v_uid is null then
    raise exception 'not authenticated';
  end if;

  perform public.ensure_usuario_profile();

  update public.usuarios u
  set
    termos_aceitos_em = now(),
    privacidade_aceita_em = now(),
    termos_versao = nullif(trim(coalesce(p_terms_version, '')), ''),
    privacidade_versao = nullif(trim(coalesce(p_privacy_version, '')), ''),
    consentimento_ip = coalesce(nullif(u.consentimento_ip, ''), v_ip),
    consentimento_user_agent = coalesce(nullif(u.consentimento_user_agent, ''), v_ua)
  where u.id = v_uid;
end;
$$;

revoke all on function public.accept_legal_terms(text, text) from public;
grant execute on function public.accept_legal_terms(text, text) to authenticated;

update public.usuarios
set limite_funcionarios = 1
where status_pagamento = 'trial'
  and (limite_funcionarios is null or limite_funcionarios < 1);

update public.usuarios
set data_vencimento = current_date + 7
```

```
where status_pagamento = 'trial'
  and data_vencimento is null;`
  }, [])

  const sqlPaymentsStripe = useMemo(() => {
    return `alter table public.usuarios add column if not exists free_trial_consumido boolean not null default false;
alter table public.usuarios add column if not exists stripe_customer_id text;
alter table public.usuarios add column if not exists stripe_subscription_id text;
alter table public.usuarios add column if not exists stripe_checkout_session_id text;
alter table public.usuarios add column if not exists stripe_last_event_id text;
alter table public.usuarios add column if not exists stripe_last_event_at timestamptz;

update public.usuarios
set free_trial_consumido = true
where plano is not null
  and lower(trim(plano)) <> 'free'
  and free_trial_consumido is distinct from true;

create index if not exists usuarios_stripe_customer_id_idx on public.usuarios (stripe_customer_id);
create index if not exists usuarios_stripe_subscription_id_idx on public.usuarios (stripe_subscription_id);`
  }, [])

  const sqlRlsApp = useMemo(() => {
    return `alter table public.usuarios enable row level security;
alter table public.funcionarios enable row level security;
alter table public.servicos enable row level security;
alter table public.agendamentos enable row level security;
alter table public.bloqueios enable row level security;

alter table public.usuarios add column if not exists termos_aceitos_em timestamptz;
alter table public.usuarios add column if not exists privacidade_aceita_em timestamptz;
alter table public.usuarios add column if not exists termos_versao text;
alter table public.usuarios add column if not exists privacidade_versao text;
alter table public.usuarios add column if not exists consentimento_ip text;
alter table public.usuarios add column if not exists consentimento_user_agent text;

alter table public.servicos add column if not exists taxa_agendamento numeric not null default 0;

alter table public.servicos add column if not exists buffer_antes_min int not null default 0;
alter table public.servicos add column if not exists buffer_depois_min int not null default 0;
alter table public.servicos add column if not exists antecedencia_minutos int not null default 120;
alter table public.servicos add column if not exists janela_max_dias int not null default 365;
alter table public.servicos add column if not exists dia_inteiro boolean not null default false;

alter table public.funcionarios add column if not exists capacidade_dia_inteiro int not null default 1;

do $$
begin
  if not exists (
    select 1
    from pg_constraint
    where conname = 'funcionarios_capacidade_dia_inteiro_chk'
      and conrelid = 'public.funcionarios'::regclass
  ) then
    alter table public.funcionarios
      add constraint funcionarios_capacidade_dia_inteiro_chk check (capacidade_dia_inteiro between 1 and 2);
  end if;
end;
$$;

alter table public.agendamentos add column if not exists extras jsonb;

create table if not exists public.super_admin (
  id uuid primary key references auth.users(id) on delete cascade,
  nome text not null default 'Super Admin',
  email text null,
  nivel text not null default 'super_admin',
  criado_em timestamptz not null default now(),
  whatsapp_api_url text,
  whatsapp_api_key text,
  whatsapp_instance_name text
);
```

```
alter table public.super_admin enable row level security;

drop policy if exists "super_admin_self_select" on public.super_admin;
create policy "super_admin_self_select" on public.super_admin
for select to authenticated
using (id = auth.uid());

drop policy if exists "super_admin_self_insert" on public.super_admin;
create policy "super_admin_self_insert" on public.super_admin
for insert to authenticated
with check (id = auth.uid());

drop policy if exists "super_admin_self_update" on public.super_admin;
create policy "super_admin_self_update" on public.super_admin
for update to authenticated
using (id = auth.uid())
with check (id = auth.uid());

grant select, insert, update on public.super_admin to authenticated;

create or replace function public.is_super_admin() returns boolean
language sql stable as $$
  select exists (select 1 from public.super_admin sa where sa.id = auth.uid())
$$;

create or replace function public.can_view_master_agenda(p_master_id uuid) returns boolean
language sql stable security definer
set search_path = public as $$
  select exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = p_master_id
      and f.ativo = true
      and f.pode_ver_agenda = true
      and f.permissao in ('admin', 'atendente')
  )
$$;

drop policy if exists "usuarios_select_self_or_master_for_funcionario" on public.usuarios;
create policy "usuarios_select_self_or_master_for_funcionario" on public.usuarios
for select to authenticated
using (
  id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = usuarios.id
      and f.ativo = true
  )
);

drop policy if exists "usuarios_update_self" on public.usuarios;
create policy "usuarios_update_self" on public.usuarios
for update to authenticated
using (
  id = auth.uid()
  or public.is_super_admin()
)
with check (
  id = auth.uid()
  or public.is_super_admin()
);

drop policy if exists "funcionarios_select_master_or_self" on public.funcionarios;
create policy "funcionarios_select_master_or_self" on public.funcionarios
for select to authenticated
using (
  usuario_master_id = auth.uid()
```

```
  or public.is_super_admin()
  or id = auth.uid()
  or public.can_view_master_agenda(funcionarios.usuario_master_id)
);

drop policy if exists "funcionarios_insert_master" on public.funcionarios;
create policy "funcionarios_insert_master" on public.funcionarios
for insert to authenticated
with check (
  usuario_master_id = auth.uid()
  or public.is_super_admin()
);

drop policy if exists "funcionarios_update_master" on public.funcionarios;
create policy "funcionarios_update_master" on public.funcionarios
for update to authenticated
using (
  usuario_master_id = auth.uid()
  or public.is_super_admin()
)
with check (
  usuario_master_id = auth.uid()
  or public.is_super_admin()
);

drop policy if exists "funcionarios_delete_master" on public.funcionarios;
create policy "funcionarios_delete_master" on public.funcionarios
for delete to authenticated
using (
  usuario_master_id = auth.uid()
  or public.is_super_admin()
);

drop policy if exists "servicos_select_master_or_funcionario" on public.servicos;
create policy "servicos_select_master_or_funcionario" on public.servicos
for select to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = servicos.usuario_id
      and f.ativo = true
  )
);

drop policy if exists "servicos_insert_master_or_funcionario" on public.servicos;
create policy "servicos_insert_master_or_funcionario" on public.servicos
for insert to authenticated
with check (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = servicos.usuario_id
      and f.ativo = true
      and f.pode_gerenciar_servicos = true
  )
);

drop policy if exists "servicos_update_master_or_funcionario" on public.servicos;
create policy "servicos_update_master_or_funcionario" on public.servicos
for update to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
```

```
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = servicos.usuario_id
      and f.ativo = true
      and f.pode_gerenciar_servicos = true
  )
)
with check (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = servicos.usuario_id
      and f.ativo = true
      and f.pode_gerenciar_servicos = true
  )
);

drop policy if exists "servicos_delete_master_or_funcionario" on public.servicos;
create policy "servicos_delete_master_or_funcionario" on public.servicos
for delete to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = servicos.usuario_id
      and f.ativo = true
      and f.pode_gerenciar_servicos = true
  )
);

drop policy if exists "agendamentos_select_master_or_funcionario" on public.agendamentos;
create policy "agendamentos_select_master_or_funcionario" on public.agendamentos
for select to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = agendamentos.usuario_id
      and f.ativo = true
      and f.pode_ver_agenda = true
      and f.permissao in ('admin', 'atendente')
  )
  or (
    funcionario_id = auth.uid()
    and exists (
      select 1
      from public.funcionarios f
      where f.id = auth.uid()
        and f.usuario_master_id = agendamentos.usuario_id
        and f.ativo = true
        and f.pode_ver_agenda = true
    )
  )
);

drop policy if exists "agendamentos_insert_master_or_funcionario" on public.agendamentos;
create policy "agendamentos_insert_master_or_funcionario" on public.agendamentos
for insert to authenticated
with check (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or (
    funcionario_id = auth.uid()
```

```
    and exists (
      select 1
      from public.funcionarios f
      where f.id = auth.uid()
        and f.usuario_master_id = agendamentos.usuario_id
        and f.ativo = true
        and f.pode_criar_agendamentos = true
    )
  )
);

drop policy if exists "agendamentos_update_master_or_funcionario" on public.agendamentos;
create policy "agendamentos_update_master_or_funcionario" on public.agendamentos
for update to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or (
    funcionario_id = auth.uid()
    and exists (
      select 1
      from public.funcionarios f
      where f.id = auth.uid()
        and f.usuario_master_id = agendamentos.usuario_id
        and f.ativo = true
        and (f.pode_cancelar_agendamentos = true or f.pode_criar_agendamentos = true)
    )
  )
)
with check (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or (
    funcionario_id = auth.uid()
    and exists (
      select 1
      from public.funcionarios f
      where f.id = auth.uid()
        and f.usuario_master_id = agendamentos.usuario_id
        and f.ativo = true
        and (f.pode_cancelar_agendamentos = true or f.pode_criar_agendamentos = true)
    )
  )
);

drop policy if exists "agendamentos_delete_master" on public.agendamentos;
create policy "agendamentos_delete_master" on public.agendamentos
for delete to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
);

drop policy if exists "bloqueios_select_master_or_funcionario" on public.bloqueios;
create policy "bloqueios_select_master_or_funcionario" on public.bloqueios
for select to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = bloqueios.usuario_id
      and f.ativo = true
      and f.pode_ver_agenda = true
      and f.permissao in ('admin', 'atendente')
  )
  or (
    (bloqueios.funcionario_id is null or bloqueios.funcionario_id = auth.uid())
    and exists (
      select 1
```

```
      from public.funcionarios f
      where f.id = auth.uid()
        and f.usuario_master_id = bloqueios.usuario_id
        and f.ativo = true
        and f.pode_ver_agenda = true
    )
  )
);

drop policy if exists "bloqueios_insert_master_or_funcionario" on public.bloqueios;
create policy "bloqueios_insert_master_or_funcionario" on public.bloqueios
for insert to authenticated
with check (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = bloqueios.usuario_id
      and f.ativo = true
      and f.pode_bloquear_horarios = true
      and (bloqueios.funcionario_id = auth.uid() or (bloqueios.funcionario_id is null and f.permissao = 'admin'))
  )
);

drop policy if exists "bloqueios_update_master_or_funcionario" on public.bloqueios;
create policy "bloqueios_update_master_or_funcionario" on public.bloqueios
for update to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = bloqueios.usuario_id
      and f.ativo = true
      and f.pode_bloquear_horarios = true
      and (bloqueios.funcionario_id = auth.uid() or (bloqueios.funcionario_id is null and f.permissao = 'admin'))
  )
)
with check (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = bloqueios.usuario_id
      and f.ativo = true
      and f.pode_bloquear_horarios = true
      and (bloqueios.funcionario_id = auth.uid() or (bloqueios.funcionario_id is null and f.permissao = 'admin'))
  )
);

drop policy if exists "bloqueios_delete_master_or_funcionario" on public.bloqueios;
create policy "bloqueios_delete_master_or_funcionario" on public.bloqueios
for delete to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = bloqueios.usuario_id
      and f.ativo = true
      and f.pode_bloquear_horarios = true
      and (bloqueios.funcionario_id = auth.uid() or (bloqueios.funcionario_id is null and f.permissao = 'admin'))
  )
);
```

```
grant select, update on public.usuarios to authenticated;
grant select, insert, update, delete on public.funcionarios to authenticated;
grant select, insert, update, delete on public.servicos to authenticated;
grant select, insert, update, delete on public.agendamentos to authenticated;
grant select, insert, update, delete on public.bloqueios to authenticated;`
  }, [])

  const sqlFuncionariosPermissaoAtendente = useMemo(() => {
    return `do $$
begin
  if exists (
    select 1
    from pg_constraint
    where conname = 'funcionarios_permissao_check'
      and conrelid = 'public.funcionarios'::regclass
  ) then
    alter table public.funcionarios drop constraint funcionarios_permissao_check;
  end if;

  alter table public.funcionarios
    add constraint funcionarios_permissao_check
    check (permissao is null or permissao in ('admin', 'funcionario', 'atendente'));
end;
$$;`
  }, [])

  const sqlPublicOcupacoes = useMemo(() => {
    return `drop function if exists public.public_get_ocupacoes(uuid, date, uuid);

alter table public.servicos add column if not exists buffer_antes_min int not null default 0;
alter table public.servicos add column if not exists buffer_depois_min int not null default 0;

create or replace function public.public_get_ocupacoes(p_usuario_id uuid, p_data date, p_funcionario_id uuid)
returns table (start_min int, end_min int)
language plpgsql
security definer
set search_path = public
as $$
begin
  return query
  select
    (floor(extract(epoch from a.hora_inicio::time) / 60)::int - coalesce(s.buffer_antes_min, 0)::int) as start_min,
    (floor(extract(epoch from coalesce(a.hora_fim, a.hora_inicio)::time) / 60)::int + coalesce(s.buffer_depois_min,
0)::int) as end_min
  from public.agendamentos a
  left join public.servicos s on s.id = a.servico_id and s.usuario_id = a.usuario_id
  where a.usuario_id = p_usuario_id
    and a.data = p_data
    and a.status <> 'cancelado'
    and (p_funcionario_id is null or a.funcionario_id is null or a.funcionario_id = p_funcionario_id)

  union all

  select
    floor(extract(epoch from b.hora_inicio::time) / 60)::int as start_min,
    floor(extract(epoch from b.hora_fim::time) / 60)::int as end_min
  from public.bloqueios b
  where b.usuario_id = p_usuario_id
    and b.data = p_data
    and (p_funcionario_id is null or b.funcionario_id is null or b.funcionario_id = p_funcionario_id);
end;
$$;

revoke all on function public.public_get_ocupacoes(uuid, date, uuid) from public;
grant execute on function public.public_get_ocupacoes(uuid, date, uuid) to anon;
grant execute on function public.public_get_ocupacoes(uuid, date, uuid) to authenticated;`
  }, [])

  const sqlTaxaAgendamento = useMemo(() => {
    return `alter table public.servicos add column if not exists taxa_agendamento numeric not null default 0;
```

```
alter table public.servicos add column if not exists buffer_antes_min int not null default 0;
alter table public.servicos add column if not exists buffer_depois_min int not null default 0;
alter table public.servicos add column if not exists antecedencia_minutos int not null default 120;
alter table public.servicos add column if not exists janela_max_dias int not null default 365;

create table if not exists public.taxa_agendamento_pagamentos (
  id uuid primary key default gen_random_uuid(),
  criado_em timestamptz not null default now(),
  usuario_id uuid not null references public.usuarios(id) on delete cascade,
  agendamento_id uuid null references public.agendamentos(id) on delete set null,
  servico_id uuid null references public.servicos(id) on delete set null,
  funcionario_id uuid null references public.funcionarios(id) on delete set null,
  cliente_nome text null,
  cliente_telefone text null,
  valor numeric not null default 0,
  moeda text not null default 'brl',
  status text not null default 'pendente',
  stripe_checkout_session_id text null,
  stripe_payment_intent_id text null,
  pago_em timestamptz null,
  credito_em timestamptz null,
  usado_em timestamptz null,
  utilizado_em_agendamento_id uuid null references public.agendamentos(id) on delete set null
);

do $$
begin
  if to_regclass('public.unidades') is null then
    alter table public.taxa_agendamento_pagamentos add column if not exists unidade_id uuid;
  else
    alter table public.taxa_agendamento_pagamentos add column if not exists unidade_id uuid references
public.unidades(id) on delete set null;
  end if;
end;
$$;

create index if not exists taxa_agendamento_pagamentos_usuario_idx on public.taxa_agendamento_pagamentos (usuario_id,
status, criado_em desc);
create index if not exists taxa_agendamento_pagamentos_cliente_idx on public.taxa_agendamento_pagamentos (usuario_id,
cliente_telefone, status, criado_em desc);
create unique index if not exists taxa_agendamento_pagamentos_stripe_session_unique on
public.taxa_agendamento_pagamentos (stripe_checkout_session_id) where stripe_checkout_session_id is not null;

alter table public.taxa_agendamento_pagamentos enable row level security;

drop policy if exists "taxa_agendamento_pagamentos_select_master_or_funcionario" on public.taxa_agendamento_pagamentos;
create policy "taxa_agendamento_pagamentos_select_master_or_funcionario" on public.taxa_agendamento_pagamentos
for select to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = taxa_agendamento_pagamentos.usuario_id
      and f.ativo = true
      and f.pode_ver_financeiro = true
  )
);

grant select on public.taxa_agendamento_pagamentos to authenticated;

create or replace function public.agendamentos_tornar_taxa_credito()
returns trigger
language plpgsql
security definer
set search_path = public
as $$
begin
  if TG_OP = 'UPDATE' then
    if lower(coalesce(NEW.status::text, '')) in ('nao_compareceu', 'não_compareceu', 'no_show')
```

```
      and lower(coalesce(OLD.status::text, '')) not in ('nao_compareceu', 'não_compareceu', 'no_show') then
      update public.taxa_agendamento_pagamentos
      set status = 'credito', credito_em = now()
      where agendamento_id = NEW.id
        and status = 'pago';
    end if;
  end if;
  return NEW;
end;
$$;

drop trigger if exists agendamentos_tornar_taxa_credito_trg on public.agendamentos;
create trigger agendamentos_tornar_taxa_credito_trg
after update of status on public.agendamentos
for each row execute function public.agendamentos_tornar_taxa_credito();

create or replace function public.consume_taxa_agendamento_credito(p_usuario_id uuid, p_cliente_telefone text)
returns uuid
language plpgsql
security definer
set search_path = public
as $$
declare
  v_id uuid;
begin
  if p_usuario_id is null or nullif(trim(coalesce(p_cliente_telefone, '')), '') is null then
    return null;
  end if;

  select t.id
  into v_id
  from public.taxa_agendamento_pagamentos t
  where t.usuario_id = p_usuario_id
    and t.cliente_telefone = p_cliente_telefone
    and t.status = 'credito'
  order by t.criado_em asc
  for update skip locked
  limit 1;

  if v_id is null then
    return null;
  end if;

  update public.taxa_agendamento_pagamentos
  set status = 'reservado'
  where id = v_id;

  return v_id;
end;
$$;

revoke all on function public.consume_taxa_agendamento_credito(uuid, text) from public;
grant execute on function public.consume_taxa_agendamento_credito(uuid, text) to service_role;

drop function if exists public.public_get_servicos_publicos(uuid);

create or replace function public.public_get_servicos_publicos(p_usuario_id uuid)
returns table (
  id uuid,
  nome text,
  descricao text,
  duracao_minutos int,
  buffer_antes_min int,
  buffer_depois_min int,
  antecedencia_minutos int,
  janela_max_dias int,
  dia_inteiro boolean,
  preco numeric,
  taxa_agendamento numeric,
  cor text,
  foto_url text
)
```

```
language plpgsql
security definer
set search_path = public
as $$
begin
  return query
  select
    s.id::uuid,
    s.nome::text,
    s.descricao::text,
    s.duracao_minutos::int,
    coalesce(s.buffer_antes_min, 0)::int,
    coalesce(s.buffer_depois_min, 0)::int,
    coalesce(s.antecedencia_minutos, 120)::int,
    365::int,
    coalesce(s.dia_inteiro, false)::boolean,
    s.preco::numeric,
    s.taxa_agendamento::numeric,
    s.cor::text,
    s.foto_url::text
  from public.servicos s
  where s.usuario_id = p_usuario_id
    and s.ativo = true
  order by s.ordem asc nulls last, s.criado_em asc;
end;
$$;

do $$
begin
  if not exists (
    select 1
    from pg_proc p
    join pg_namespace n on n.oid = p.pronamespace
    where n.nspname = 'public'
      and p.proname = 'public_create_agendamento_publico'
      and p.proargtypes = '2950 1082 25 2950 25 25 2950 3802 25'::oidvector
  ) and exists (
    select 1
    from pg_proc p
    join pg_namespace n on n.oid = p.pronamespace
    where n.nspname = 'public'
      and p.proname = 'public_create_agendamento_publico'
      and p.proargtypes = '2950 1082 25 2950 25 25 2950 3802'::oidvector
  ) then
    execute $fn$
      create function public.public_create_agendamento_publico(
        p_usuario_id uuid,
        p_data date,
        p_hora_inicio text,
        p_servico_id uuid,
        p_cliente_nome text,
        p_cliente_telefone text,
        p_funcionario_id uuid,
        p_extras jsonb default null,
        p_status text default 'pendente'
      )
      returns uuid
      language plpgsql
      security definer
      set search_path = public
      as $body$
      declare
        v_created_id uuid;
        v_status text;
      begin
        v_status := lower(coalesce(nullif(trim(coalesce(p_status, '')), ''), 'pendente'));
        if v_status not in ('confirmado', 'pendente') then
          raise exception 'status_invalido';
        end if;

        v_created_id := public.public_create_agendamento_publico(
          p_usuario_id,
```

```
          p_data,
          p_hora_inicio,
          p_servico_id,
          p_cliente_nome,
          p_cliente_telefone,
          p_funcionario_id,
          p_extras
        );

        update public.agendamentos
        set status = v_status
        where id = v_created_id
          and usuario_id = p_usuario_id;

        return v_created_id;
      end;
      $body$;
    $fn$;
  end if;

  if not exists (
    select 1
    from pg_proc p
    join pg_namespace n on n.oid = p.pronamespace
    where n.nspname = 'public'
      and p.proname = 'public_create_agendamento_publico'
      and p.proargtypes = '2950 1082 25 2950 25 25 2950 2950 3802 25'::oidvector
  ) and exists (
    select 1
    from pg_proc p
    join pg_namespace n on n.oid = p.pronamespace
    where n.nspname = 'public'
      and p.proname = 'public_create_agendamento_publico'
      and p.proargtypes = '2950 1082 25 2950 25 25 2950 2950 3802'::oidvector
  ) then
    execute $fn$
      create function public.public_create_agendamento_publico(
        p_usuario_id uuid,
        p_data date,
        p_hora_inicio text,
        p_servico_id uuid,
        p_cliente_nome text,
        p_cliente_telefone text,
        p_funcionario_id uuid,
        p_unidade_id uuid default null,
        p_extras jsonb default null,
        p_status text default 'pendente'
      )
      returns uuid
      language plpgsql
      security definer
      set search_path = public
      as $body$
      declare
        v_created_id uuid;
        v_status text;
      begin
        v_status := lower(coalesce(nullif(trim(coalesce(p_status, '')), ''), 'pendente'));
        if v_status not in ('confirmado', 'pendente') then
          raise exception 'status_invalido';
        end if;

        v_created_id := public.public_create_agendamento_publico(
          p_usuario_id,
          p_data,
          p_hora_inicio,
          p_servico_id,
          p_cliente_nome,
          p_cliente_telefone,
          p_funcionario_id,
          p_unidade_id,
          p_extras
```

```
        );

        update public.agendamentos
        set status = v_status
        where id = v_created_id
          and usuario_id = p_usuario_id;

        return v_created_id;
      end;
      $body$;
    $fn$;
  end if;

  if not exists (
    select 1
    from pg_proc p
    join pg_namespace n on n.oid = p.pronamespace
    where n.nspname = 'public'
      and p.proname = 'public_create_agendamento_publico'
      and p.proargtypes = '2950 1082 25 2950 25 25 2950 25'::oidvector
  ) and exists (
    select 1
    from pg_proc p
    join pg_namespace n on n.oid = p.pronamespace
    where n.nspname = 'public'
      and p.proname = 'public_create_agendamento_publico'
      and p.proargtypes = '2950 1082 25 2950 25 25 2950'::oidvector
  ) then
    execute $fn$
      create function public.public_create_agendamento_publico(
        p_usuario_id uuid,
        p_data date,
        p_hora_inicio text,
        p_servico_id uuid,
        p_cliente_nome text,
        p_cliente_telefone text,
        p_funcionario_id uuid,
        p_status text default 'pendente'
      )
      returns uuid
      language plpgsql
      security definer
      set search_path = public
      as $body$
      declare
        v_created_id uuid;
        v_status text;
      begin
        v_status := lower(coalesce(nullif(trim(coalesce(p_status, '')), ''), 'pendente'));
        if v_status not in ('confirmado', 'pendente') then
          raise exception 'status_invalido';
        end if;

        v_created_id := public.public_create_agendamento_publico(
          p_usuario_id,
          p_data,
          p_hora_inicio,
          p_servico_id,
          p_cliente_nome,
          p_cliente_telefone,
          p_funcionario_id
        );

        update public.agendamentos
        set status = v_status
        where id = v_created_id
          and usuario_id = p_usuario_id;

        return v_created_id;
      end;
      $body$;
    $fn$;
```

```
  end if;

  if not exists (
    select 1
    from pg_proc p
    join pg_namespace n on n.oid = p.pronamespace
    where n.nspname = 'public'
      and p.proname = 'public_create_agendamento_publico'
      and p.proargtypes = '2950 1082 25 2950 25 25 2950 2950 25'::oidvector
  ) and exists (
    select 1
    from pg_proc p
    join pg_namespace n on n.oid = p.pronamespace
    where n.nspname = 'public'
      and p.proname = 'public_create_agendamento_publico'
      and p.proargtypes = '2950 1082 25 2950 25 25 2950 2950'::oidvector
  ) then
    execute $fn$
      create function public.public_create_agendamento_publico(
        p_usuario_id uuid,
        p_data date,
        p_hora_inicio text,
        p_servico_id uuid,
        p_cliente_nome text,
        p_cliente_telefone text,
        p_funcionario_id uuid,
        p_unidade_id uuid default null,
        p_status text default 'pendente'
      )
      returns uuid
      language plpgsql
      security definer
      set search_path = public
      as $body$
      declare
        v_created_id uuid;
        v_status text;
      begin
        v_status := lower(coalesce(nullif(trim(coalesce(p_status, '')), ''), 'pendente'));
        if v_status not in ('confirmado', 'pendente') then
          raise exception 'status_invalido';
        end if;

        v_created_id := public.public_create_agendamento_publico(
          p_usuario_id,
          p_data,
          p_hora_inicio,
          p_servico_id,
          p_cliente_nome,
          p_cliente_telefone,
          p_funcionario_id,
          p_unidade_id
        );

        update public.agendamentos
        set status = v_status
        where id = v_created_id
          and usuario_id = p_usuario_id;

        return v_created_id;
      end;
      $body$;
    $fn$;
  end if;
end;
$$;`
  }, [])

  const sqlPublicBooking = useMemo(() => {
    return `create extension if not exists pgcrypto;

drop function if exists public.public_get_usuario_publico(text);
```

```
drop function if exists public.public_get_usuario_publico(text, text);
drop function if exists public.public_get_servicos_publicos(uuid);
drop function if exists public.public_get_funcionarios_publicos(uuid);
drop function if exists public.public_get_funcionarios_publicos(uuid, uuid);

drop function if exists public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid);
drop function if exists public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, uuid);
drop function if exists public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, jsonb);
drop function if exists public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, jsonb, text);
drop function if exists public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, jsonb, uuid);
drop function if exists public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, uuid, text);
drop function if exists public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, uuid, jsonb);
drop function if exists public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, uuid, jsonb,
text);

alter table public.usuarios add column if not exists slug text;
create unique index if not exists usuarios_slug_unique on public.usuarios (slug);

alter table public.usuarios add column if not exists logo_url text;
alter table public.usuarios add column if not exists instagram_url text;
alter table public.usuarios add column if not exists tipo_negocio text;
alter table public.usuarios add column if not exists public_primary_color text;
alter table public.usuarios add column if not exists public_background_color text;
alter table public.usuarios add column if not exists public_use_background_image boolean not null default false;
alter table public.usuarios add column if not exists public_background_image_url text;
alter table public.usuarios add column if not exists timezone text;
alter table public.usuarios add column if not exists limite_agendamentos_mes int;

alter table public.usuarios add column if not exists termos_aceitos_em timestamptz;
alter table public.usuarios add column if not exists privacidade_aceita_em timestamptz;
alter table public.usuarios add column if not exists termos_versao text;
alter table public.usuarios add column if not exists privacidade_versao text;
alter table public.usuarios add column if not exists consentimento_ip text;
alter table public.usuarios add column if not exists consentimento_user_agent text;

alter table public.servicos add column if not exists foto_url text;
alter table public.servicos add column if not exists taxa_agendamento numeric not null default 0;
alter table public.servicos add column if not exists buffer_antes_min int not null default 0;
alter table public.servicos add column if not exists buffer_depois_min int not null default 0;
alter table public.servicos add column if not exists antecedencia_minutos int not null default 120;
alter table public.servicos add column if not exists janela_max_dias int not null default 365;
alter table public.servicos add column if not exists dia_inteiro boolean not null default false;

alter table public.funcionarios add column if not exists capacidade_dia_inteiro int not null default 1;

do $$
begin
  if not exists (
    select 1
    from pg_constraint
    where conname = 'funcionarios_capacidade_dia_inteiro_chk'
      and conrelid = 'public.funcionarios'::regclass
  ) then
    alter table public.funcionarios
      add constraint funcionarios_capacidade_dia_inteiro_chk check (capacidade_dia_inteiro between 1 and 2);
  end if;
end;
$$;

alter table public.agendamentos add column if not exists extras jsonb;

do $$
begin
  if to_regclass('public.unidades') is null then
    alter table public.funcionarios add column if not exists unidade_id uuid;
    alter table public.agendamentos add column if not exists unidade_id uuid;
    alter table public.bloqueios add column if not exists unidade_id uuid;
  else
    alter table public.funcionarios add column if not exists unidade_id uuid references public.unidades(id) on delete
set null;
    alter table public.agendamentos add column if not exists unidade_id uuid references public.unidades(id) on delete
set null;
```

```
    alter table public.bloqueios add column if not exists unidade_id uuid references public.unidades(id) on delete set
null;
  end if;
end;
$$;

create table if not exists public.taxa_agendamento_pagamentos (
  id uuid primary key default gen_random_uuid(),
  criado_em timestamptz not null default now(),
  usuario_id uuid not null references public.usuarios(id) on delete cascade,
  agendamento_id uuid null references public.agendamentos(id) on delete set null,
  servico_id uuid null references public.servicos(id) on delete set null,
  funcionario_id uuid null references public.funcionarios(id) on delete set null,
  cliente_nome text null,
  cliente_telefone text null,
  valor numeric not null default 0,
  moeda text not null default 'brl',
  status text not null default 'pendente',
  stripe_checkout_session_id text null,
  stripe_payment_intent_id text null,
  pago_em timestamptz null,
  credito_em timestamptz null,
  usado_em timestamptz null,
  utilizado_em_agendamento_id uuid null references public.agendamentos(id) on delete set null
);

do $$
begin
  if to_regclass('public.unidades') is null then
    alter table public.taxa_agendamento_pagamentos add column if not exists unidade_id uuid;
  else
    alter table public.taxa_agendamento_pagamentos add column if not exists unidade_id uuid references
public.unidades(id) on delete set null;
  end if;
end;
$$;

create index if not exists taxa_agendamento_pagamentos_usuario_idx on public.taxa_agendamento_pagamentos (usuario_id,
status, criado_em desc);
create index if not exists taxa_agendamento_pagamentos_cliente_idx on public.taxa_agendamento_pagamentos (usuario_id,
cliente_telefone, status, criado_em desc);
create unique index if not exists taxa_agendamento_pagamentos_stripe_session_unique on
public.taxa_agendamento_pagamentos (stripe_checkout_session_id) where stripe_checkout_session_id is not null;

alter table public.taxa_agendamento_pagamentos enable row level security;

drop policy if exists "taxa_agendamento_pagamentos_select_master_or_funcionario" on public.taxa_agendamento_pagamentos;
create policy "taxa_agendamento_pagamentos_select_master_or_funcionario" on public.taxa_agendamento_pagamentos
for select to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = taxa_agendamento_pagamentos.usuario_id
      and f.ativo = true
      and f.pode_ver_financeiro = true
  )
);

grant select on public.taxa_agendamento_pagamentos to authenticated;

create or replace function public.agendamentos_tornar_taxa_credito()
returns trigger
language plpgsql
security definer
set search_path = public
as $$
begin
  if TG_OP = 'UPDATE' then
```

```
    if lower(coalesce(NEW.status::text, '')) in ('nao_compareceu', 'não_compareceu', 'no_show')
      and lower(coalesce(OLD.status::text, '')) not in ('nao_compareceu', 'não_compareceu', 'no_show') then
      update public.taxa_agendamento_pagamentos
      set status = 'credito', credito_em = now()
      where agendamento_id = NEW.id
        and status = 'pago';
    end if;
  end if;
  return NEW;
end;
$$;

drop trigger if exists agendamentos_tornar_taxa_credito_trg on public.agendamentos;
create trigger agendamentos_tornar_taxa_credito_trg
after update of status on public.agendamentos
for each row execute function public.agendamentos_tornar_taxa_credito();

create or replace function public.consume_taxa_agendamento_credito(p_usuario_id uuid, p_cliente_telefone text)
returns uuid
language plpgsql
security definer
set search_path = public
as $$
declare
  v_id uuid;
begin
  if p_usuario_id is null or nullif(trim(coalesce(p_cliente_telefone, '')), '') is null then
    return null;
  end if;

  select t.id
  into v_id
  from public.taxa_agendamento_pagamentos t
  where t.usuario_id = p_usuario_id
    and t.cliente_telefone = p_cliente_telefone
    and t.status = 'credito'
  order by t.criado_em asc
  for update skip locked
  limit 1;

  if v_id is null then
    return null;
  end if;

  update public.taxa_agendamento_pagamentos
  set status = 'reservado'
  where id = v_id;

  return v_id;
end;
$$;

revoke all on function public.consume_taxa_agendamento_credito(uuid, text) from public;
grant execute on function public.consume_taxa_agendamento_credito(uuid, text) to service_role;

create or replace function public.public_get_usuario_publico(p_slug text, p_unidade_slug text default null)
returns table (
  id uuid,
  nome_negocio text,
  logo_url text,
  endereco text,
  telefone text,
  instagram_url text,
  horario_inicio text,
  horario_fim text,
  dias_trabalho int[],
  intervalo_inicio text,
  intervalo_fim text,
  ativo boolean,
  tipo_conta text,
  plano text,
  tipo_negocio text,
```

```
  public_primary_color text,
  public_background_color text,
  public_use_background_image boolean,
  public_background_image_url text,
  unidade_id uuid,
  unidade_nome text,
  unidade_slug text
)
language plpgsql
security definer
set search_path = public
as $$
begin
  if to_regclass('public.unidades') is null then
    return query
    select
      u.id::uuid,
      u.nome_negocio::text,
      u.logo_url::text,
      u.endereco::text,
      u.telefone::text,
      u.instagram_url::text,
      u.horario_inicio::text,
      u.horario_fim::text,
      u.dias_trabalho::int[],
      u.intervalo_inicio::text,
      u.intervalo_fim::text,
      u.ativo::boolean,
      u.tipo_conta::text,
      u.plano::text,
      u.tipo_negocio::text,
      u.public_primary_color::text,
      u.public_background_color::text,
      u.public_use_background_image::boolean,
      u.public_background_image_url::text,
      null::uuid as unidade_id,
      null::text as unidade_nome,
      null::text as unidade_slug
    from public.usuarios u
    where u.slug = p_slug
      and u.ativo = true
    limit 1;
  else
    return query
    with base as (
      select u.*
      from public.usuarios u
      where u.slug = p_slug
        and u.ativo = true
      limit 1
    ), uni as (
      select un.*
      from public.unidades un
      join base b on b.id = un.usuario_id
      where nullif(trim(coalesce(p_unidade_slug, '')), '') is not null
        and un.slug = p_unidade_slug
        and un.ativo = true
      limit 1
    )
    select
      b.id::uuid,
      (case when u.id is not null then u.nome::text else b.nome_negocio::text end) as nome_negocio,
      b.logo_url::text,
      coalesce(u.endereco::text, b.endereco::text) as endereco,
      coalesce(u.telefone::text, b.telefone::text) as telefone,
      b.instagram_url::text,
      coalesce(u.horario_inicio::text, b.horario_inicio::text) as horario_inicio,
      coalesce(u.horario_fim::text, b.horario_fim::text) as horario_fim,
      coalesce(u.dias_trabalho::int[], b.dias_trabalho::int[]) as dias_trabalho,
      coalesce(u.intervalo_inicio::text, b.intervalo_inicio::text) as intervalo_inicio,
      coalesce(u.intervalo_fim::text, b.intervalo_fim::text) as intervalo_fim,
      b.ativo::boolean,
```

```
      b.tipo_conta::text,
      b.plano::text,
      b.tipo_negocio::text,
      coalesce(u.public_primary_color::text, b.public_primary_color::text) as public_primary_color,
      coalesce(u.public_background_color::text, b.public_background_color::text) as public_background_color,
      coalesce(u.public_use_background_image::boolean, b.public_use_background_image::boolean) as
public_use_background_image,
      coalesce(u.public_background_image_url::text, b.public_background_image_url::text) as public_background_image_url,
      u.id::uuid as unidade_id,
      u.nome::text as unidade_nome,
      u.slug::text as unidade_slug
    from base b
    left join uni u on true;
  end if;
end;
$$;

create or replace function public.public_get_servicos_publicos(p_usuario_id uuid)
returns table (
  id uuid,
  nome text,
  descricao text,
  duracao_minutos int,
  buffer_antes_min int,
  buffer_depois_min int,
  antecedencia_minutos int,
  janela_max_dias int,
  dia_inteiro boolean,
  preco numeric,
  taxa_agendamento numeric,
  cor text,
  foto_url text
)
language plpgsql
security definer
set search_path = public
as $$
begin
  return query
  select
    s.id::uuid,
    s.nome::text,
    s.descricao::text,
    s.duracao_minutos::int,
    coalesce(s.buffer_antes_min, 0)::int,
    coalesce(s.buffer_depois_min, 0)::int,
    coalesce(s.antecedencia_minutos, 120)::int,
    365::int,
    coalesce(s.dia_inteiro, false)::boolean,
    s.preco::numeric,
    s.taxa_agendamento::numeric,
    s.cor::text,
    s.foto_url::text
  from public.servicos s
  where s.usuario_id = p_usuario_id
    and s.ativo = true
  order by s.ordem asc nulls last, s.criado_em asc;
end;
$$;

create or replace function public.public_get_funcionarios_publicos(p_usuario_master_id uuid, p_unidade_id uuid default
null)
returns table (
  id uuid,
  nome_completo text,
  horario_inicio text,
  horario_fim text,
  dias_trabalho int[],
  intervalo_inicio text,
  intervalo_fim text
)
language plpgsql
```

```
security definer
set search_path = public
as $$
begin
  return query
  select f.id::uuid, f.nome_completo::text, f.horario_inicio::text, f.horario_fim::text, f.dias_trabalho::int[],
f.intervalo_inicio::text, f.intervalo_fim::text
  from public.funcionarios f
  where f.usuario_master_id = p_usuario_master_id
    and f.ativo = true
    and lower(trim(f.permissao)) = 'funcionario'
    and (p_unidade_id is null or f.unidade_id = p_unidade_id)
  order by f.criado_em asc;
end;
$$;

drop function if exists public.public_get_slots_publicos(uuid, date, uuid, uuid);
drop function if exists public.public_get_slots_publicos(uuid, date, uuid, uuid, uuid);

create or replace function public.public_get_slots_publicos(
  p_usuario_id uuid,
  p_data date,
  p_servico_id uuid,
  p_funcionario_id uuid,
  p_unidade_id uuid default null
)
returns table (hora_inicio text)
language plpgsql
security definer
set search_path = public
as $$
declare
  v_uid uuid;
  v_base record;
  v_staff_exists boolean;
  v_staff_horario_inicio time;
  v_staff_horario_fim time;
  v_staff_dias_trabalho int[];
  v_staff_intervalo_inicio time;
  v_staff_intervalo_fim time;
  v_duracao int;
  v_dia_inteiro boolean;
  v_weekday int;
  v_sched_inicio time;
  v_sched_fim time;
  v_iv_inicio time;
  v_iv_fim time;
  v_sched_inicio_min int;
  v_sched_fim_min int;
  v_iv_inicio_min int;
  v_iv_fim_min int;
  v_start_min int;
  v_end_min int;
  v_now_min int;
  v_step_min int := 30;
  v_min_lead_min int;
  v_max_days int;
  v_buffer_antes int;
  v_buffer_depois int;
  v_tz text;
  v_today date;
  v_cap int;
  v_diaria_count int;
begin
  if p_usuario_id is null or p_data is null or p_servico_id is null then
    raise exception 'invalid_payload';
  end if;

  v_uid := p_usuario_id;
  select u.horario_inicio, u.horario_fim, u.dias_trabalho, u.intervalo_inicio, u.intervalo_fim, u.timezone
  into v_base
  from public.usuarios u
```

```
  where u.id = v_uid
    and u.ativo = true;

  if v_base is null then
    raise exception 'usuario_invalido';
  end if;

  v_tz := coalesce(nullif(v_base.timezone::text, ''), 'America/Sao_Paulo');
  v_today := (now() at time zone v_tz)::date;

  if p_data < v_today then
    raise exception 'data_passada';
  end if;

  select
    s.duracao_minutos,
    coalesce(s.buffer_antes_min, 0),
    coalesce(s.buffer_depois_min, 0),
    coalesce(s.antecedencia_minutos, 120),
    365,
    coalesce(s.dia_inteiro, false)::boolean
  into v_duracao, v_buffer_antes, v_buffer_depois, v_min_lead_min, v_max_days, v_dia_inteiro
  from public.servicos s
  where s.id = p_servico_id
    and s.usuario_id = v_uid
    and s.ativo = true;

  if v_duracao is null or v_duracao <= 0 then
    raise exception 'servico_invalido';
  end if;

  if p_data > (v_today + v_max_days) then
    raise exception 'data_muito_futura';
  end if;

  if p_funcionario_id is not null then
    select true, f.horario_inicio, f.horario_fim, f.dias_trabalho, f.intervalo_inicio, f.intervalo_fim
    into v_staff_exists, v_staff_horario_inicio, v_staff_horario_fim, v_staff_dias_trabalho, v_staff_intervalo_inicio,
v_staff_intervalo_fim
    from public.funcionarios f
    where f.id = p_funcionario_id
      and f.usuario_master_id = v_uid
      and f.ativo = true;

    if v_staff_exists is distinct from true then
      raise exception 'funcionario_invalido';
    end if;
  end if;

  v_sched_inicio := coalesce(v_staff_horario_inicio, v_base.horario_inicio::time);
  v_sched_fim := coalesce(v_staff_horario_fim, v_base.horario_fim::time);
  v_iv_inicio := coalesce(v_staff_intervalo_inicio, v_base.intervalo_inicio::time);
  v_iv_fim := coalesce(v_staff_intervalo_fim, v_base.intervalo_fim::time);

  if v_sched_inicio is null or v_sched_fim is null then
    raise exception 'horarios_nao_configurados';
  end if;

  v_weekday := extract(dow from p_data)::int;

  if v_staff_dias_trabalho is not null and cardinality(v_staff_dias_trabalho) > 0 then
    if not (v_weekday = any(v_staff_dias_trabalho)) then
      raise exception 'fora_do_dia_de_trabalho';
    end if;
  elsif v_base.dias_trabalho is not null and cardinality(v_base.dias_trabalho) > 0 then
    if not (v_weekday = any(v_base.dias_trabalho)) then
      raise exception 'fora_do_dia_de_trabalho';
    end if;
  end if;

  v_sched_inicio_min := floor(extract(epoch from v_sched_inicio) / 60)::int;
  v_sched_fim_min := floor(extract(epoch from v_sched_fim) / 60)::int;
```

```
  v_start_min := v_sched_inicio_min + v_buffer_antes;
  v_end_min := v_sched_fim_min - (v_duracao + v_buffer_depois);

  if v_end_min < v_start_min then
    return;
  end if;

  if v_iv_inicio is not null and v_iv_fim is not null then
    v_iv_inicio_min := floor(extract(epoch from v_iv_inicio) / 60)::int;
    v_iv_fim_min := floor(extract(epoch from v_iv_fim) / 60)::int;
  else
    v_iv_inicio_min := null;
    v_iv_fim_min := null;
  end if;

  if p_data = v_today then
    v_now_min := floor(extract(epoch from ((now() at time zone v_tz)::time)) / 60)::int;
    v_start_min := greatest(v_start_min, v_now_min + v_min_lead_min);
  end if;

  if v_dia_inteiro then
    if v_sched_fim_min <= v_sched_inicio_min then
      return;
    end if;

    if p_data = v_today then
      if v_sched_inicio_min < (v_now_min + v_min_lead_min) then
        return;
      end if;
    end if;

    v_cap := 1;
    if p_funcionario_id is not null then
      select coalesce(f.capacidade_dia_inteiro, 1)::int
      into v_cap
      from public.funcionarios f
      where f.id = p_funcionario_id
        and f.usuario_master_id = v_uid
        and f.ativo = true;
    end if;
    if v_cap < 1 then v_cap := 1; end if;
    if v_cap > 2 then v_cap := 2; end if;

    if exists (
      select 1
      from public.bloqueios b
      where b.usuario_id = v_uid
        and b.data = p_data
        and (p_funcionario_id is null or b.funcionario_id is null or b.funcionario_id = p_funcionario_id)
        and floor(extract(epoch from b.hora_inicio::time) / 60)::int < v_sched_fim_min
        and v_sched_inicio_min < floor(extract(epoch from b.hora_fim::time) / 60)::int
      limit 1
    ) then
      return;
    end if;

    if exists (
      select 1
      from public.agendamentos a
      join public.servicos s2 on s2.id = a.servico_id and s2.usuario_id = a.usuario_id
      where a.usuario_id = v_uid
        and a.data = p_data
        and a.status <> 'cancelado'
        and (p_funcionario_id is null or a.funcionario_id is null or a.funcionario_id = p_funcionario_id)
        and coalesce(s2.dia_inteiro, false) = false
      limit 1
    ) then
      return;
    end if;

    select count(*)::int
    into v_diaria_count
```

```
      from public.agendamentos a
      join public.servicos s2 on s2.id = a.servico_id and s2.usuario_id = a.usuario_id
      where a.usuario_id = v_uid
        and a.data = p_data
        and a.status <> 'cancelado'
        and (p_funcionario_id is null or a.funcionario_id is null or a.funcionario_id = p_funcionario_id)
        and coalesce(s2.dia_inteiro, false) = true;

      if coalesce(v_diaria_count, 0) >= v_cap then
        return;
      end if;

      return query
      select to_char(v_sched_inicio, 'HH24:MI') as hora_inicio;
      return;
    end if;

    v_start_min := ((v_start_min + v_step_min - 1) / v_step_min) * v_step_min;

    if p_unidade_id is not null and to_regprocedure('public.public_get_ocupacoes(uuid,date,uuid,uuid)') is not null then
      return query
      with candidates as (
        select gs as start_min
        from generate_series(v_start_min, v_end_min, v_step_min) gs
      ),
      without_interval as (
        select c.start_min
        from candidates c
        where not (
          v_iv_inicio_min is not null
          and v_iv_fim_min is not null
          and (c.start_min - v_buffer_antes) < v_iv_fim_min
          and v_iv_inicio_min < (c.start_min + v_duracao + v_buffer_depois)
        )
      ),
      free as (
        select w.start_min
        from without_interval w
        where not exists (
          select 1
          from public.public_get_ocupacoes(v_uid, p_data, p_funcionario_id, p_unidade_id) r
          where (w.start_min - v_buffer_antes) < r.end_min and r.start_min < (w.start_min + v_duracao + v_buffer_depois)
          limit 1
        )
      )
      select to_char((time '00:00' + (free.start_min::text || ' minutes')::interval)::time, 'HH24:MI') as hora_inicio
      from free
      order by free.start_min asc;
    else
      return query
      with candidates as (
        select gs as start_min
        from generate_series(v_start_min, v_end_min, v_step_min) gs
      ),
      without_interval as (
        select c.start_min
        from candidates c
        where not (
          v_iv_inicio_min is not null
          and v_iv_fim_min is not null
          and (c.start_min - v_buffer_antes) < v_iv_fim_min
          and v_iv_inicio_min < (c.start_min + v_duracao + v_buffer_depois)
        )
      ),
      free as (
        select w.start_min
        from without_interval w
        where not exists (
          select 1
          from public.public_get_ocupacoes(v_uid, p_data, p_funcionario_id) r
          where (w.start_min - v_buffer_antes) < r.end_min and r.start_min < (w.start_min + v_duracao + v_buffer_depois)
          limit 1
```

```
      )
    )
    select to_char((time '00:00' + (free.start_min::text || ' minutes')::interval)::time, 'HH24:MI') as hora_inicio
    from free
    order by free.start_min asc;
  end if;
end;
$$;

create or replace function public.public_create_agendamento_publico(
  p_usuario_id uuid,
  p_data date,
  p_hora_inicio text,
  p_servico_id uuid,
  p_cliente_nome text,
  p_cliente_telefone text,
  p_funcionario_id uuid,
  p_unidade_id uuid default null,
  p_extras jsonb default null,
  p_status text default 'pendente'
)
returns uuid
language plpgsql
security definer
set search_path = public
as $$
declare
  v_uid uuid;
  v_base record;
  v_staff_exists boolean;
  v_staff_horario_inicio time;
  v_staff_horario_fim time;
  v_staff_dias_trabalho int[];
  v_staff_intervalo_inicio time;
  v_staff_intervalo_fim time;
  v_duracao int;
  v_dia_inteiro boolean;
  v_start_min int;
  v_end_min int;
  v_weekday int;
  v_sched_inicio time;
  v_sched_fim time;
  v_iv_inicio time;
  v_iv_fim time;
  v_horario_fim text;
  v_created_id uuid;
  v_now_min int;
  v_min_lead_min int;
  v_max_days int;
  v_buffer_antes int;
  v_buffer_depois int;
  v_tz text;
  v_today date;
  v_limite_mes int;
  v_month_start date;
  v_month_end date;
  v_month_count int;
  v_status text;
  v_cap int;
  v_diaria_count int;
begin
  if p_usuario_id is null or p_data is null or nullif(p_hora_inicio, '') is null or p_servico_id is null then
    raise exception 'invalid_payload';
  end if;

  v_status := lower(coalesce(nullif(trim(coalesce(p_status, '')), ''), 'pendente'));
  if v_status not in ('confirmado', 'pendente') then
    raise exception 'status_invalido';
  end if;

  v_uid := p_usuario_id;
```

```
  select u.horario_inicio, u.horario_fim, u.dias_trabalho, u.intervalo_inicio, u.intervalo_fim, u.timezone, 
u.limite_agendamentos_mes
  into v_base
  from public.usuarios u
  where u.id = v_uid
    and u.ativo = true
  for update;

  if v_base is null then
    raise exception 'usuario_invalido';
  end if;

  v_tz := coalesce(nullif(v_base.timezone::text, ''), 'America/Sao_Paulo');
  v_today := (now() at time zone v_tz)::date;

  if p_data < v_today then
    raise exception 'data_passada';
  end if;

  v_limite_mes := v_base.limite_agendamentos_mes;
  if v_limite_mes is not null then
    if v_limite_mes <= 0 then
      raise exception 'limite_mensal_atingido';
    end if;

    v_month_start := date_trunc('month', p_data)::date;
    v_month_end := (v_month_start + interval '1 month')::date;

    select count(*)
    into v_month_count
    from public.agendamentos a
    where a.usuario_id = v_uid
      and a.data >= v_month_start
      and a.data < v_month_end
      and a.status <> 'cancelado';

    if v_month_count >= v_limite_mes then
      raise exception 'limite_mensal_atingido';
    end if;
  end if;

  select
    s.duracao_minutos,
    coalesce(s.buffer_antes_min, 0),
    coalesce(s.buffer_depois_min, 0),
    coalesce(s.antecedencia_minutos, 120),
    365,
    coalesce(s.dia_inteiro, false)::boolean
  into v_duracao, v_buffer_antes, v_buffer_depois, v_min_lead_min, v_max_days, v_dia_inteiro
  from public.servicos s
  where s.id = p_servico_id
    and s.usuario_id = v_uid
    and s.ativo = true;

  if v_duracao is null or v_duracao <= 0 then
    raise exception 'servico_invalido';
  end if;

  if p_data > (v_today + v_max_days) then
    raise exception 'data_muito_futura';
  end if;

  if p_funcionario_id is not null then
    select true, f.horario_inicio, f.horario_fim, f.dias_trabalho, f.intervalo_inicio, f.intervalo_fim
    into v_staff_exists, v_staff_horario_inicio, v_staff_horario_fim, v_staff_dias_trabalho, v_staff_intervalo_inicio, 
v_staff_intervalo_fim
    from public.funcionarios f
    where f.id = p_funcionario_id
      and f.usuario_master_id = v_uid
      and f.ativo = true;

    if v_staff_exists is distinct from true then
```

```
      raise exception 'funcionario_invalido';
    end if;
  end if;

  v_sched_inicio := coalesce(v_staff_horario_inicio, v_base.horario_inicio::time);
  v_sched_fim := coalesce(v_staff_horario_fim, v_base.horario_fim::time);
  v_iv_inicio := coalesce(v_staff_intervalo_inicio, v_base.intervalo_inicio::time);
  v_iv_fim := coalesce(v_staff_intervalo_fim, v_base.intervalo_fim::time);

  if v_sched_inicio is null or v_sched_fim is null then
    raise exception 'horarios_nao_configurados';
  end if;

  v_weekday := extract(dow from p_data)::int;

  if v_staff_dias_trabalho is not null and cardinality(v_staff_dias_trabalho) > 0 then
    if not (v_weekday = any(v_staff_dias_trabalho)) then
      raise exception 'fora_do_dia_de_trabalho';
    end if;
  elsif v_base.dias_trabalho is not null and cardinality(v_base.dias_trabalho) > 0 then
    if not (v_weekday = any(v_base.dias_trabalho)) then
      raise exception 'fora_do_dia_de_trabalho';
    end if;
  end if;

  if v_dia_inteiro then
    v_start_min := floor(extract(epoch from v_sched_inicio) / 60)::int;
    v_end_min := floor(extract(epoch from v_sched_fim) / 60)::int;
    v_horario_fim := to_char(v_sched_fim, 'HH24:MI');

    if v_end_min <= v_start_min then
      raise exception 'fora_do_horario';
    end if;

    if p_data = v_today then
      v_now_min := floor(extract(epoch from ((now() at time zone v_tz)::time)) / 60)::int;
      if v_start_min < v_now_min + v_min_lead_min then
        raise exception 'antecedencia_minima';
      end if;
    end if;

    v_cap := 1;
    if p_funcionario_id is not null then
      select coalesce(f.capacidade_dia_inteiro, 1)::int
      into v_cap
      from public.funcionarios f
      where f.id = p_funcionario_id
        and f.usuario_master_id = v_uid
        and f.ativo = true;
    end if;
    if v_cap < 1 then v_cap := 1; end if;
    if v_cap > 2 then v_cap := 2; end if;

    if exists (
      select 1
      from public.bloqueios b
      where b.usuario_id = v_uid
        and b.data = p_data
        and (p_funcionario_id is null or b.funcionario_id is null or b.funcionario_id = p_funcionario_id)
        and floor(extract(epoch from b.hora_inicio::time) / 60)::int < v_end_min
        and v_start_min < floor(extract(epoch from b.hora_fim::time) / 60)::int
      limit 1
    ) then
      raise exception 'ocupado';
    end if;

    if exists (
      select 1
      from public.agendamentos a
      join public.servicos s2 on s2.id = a.servico_id and s2.usuario_id = a.usuario_id
      where a.usuario_id = v_uid
        and a.data = p_data
```

```
        and a.status <> 'cancelado'
        and (p_funcionario_id is null or a.funcionario_id is null or a.funcionario_id = p_funcionario_id)
        and coalesce(s2.dia_inteiro, false) = false
      limit 1
    ) then
      raise exception 'ocupado';
    end if;

    select count(*)::int
    into v_diaria_count
    from public.agendamentos a
    join public.servicos s2 on s2.id = a.servico_id and s2.usuario_id = a.usuario_id
    where a.usuario_id = v_uid
      and a.data = p_data
      and a.status <> 'cancelado'
      and (p_funcionario_id is null or a.funcionario_id is null or a.funcionario_id = p_funcionario_id)
      and coalesce(s2.dia_inteiro, false) = true;

    if coalesce(v_diaria_count, 0) >= v_cap then
      raise exception 'ocupado';
    end if;
  else
    v_start_min := floor(extract(epoch from (p_hora_inicio::time)) / 60)::int;
    v_end_min := v_start_min + v_duracao;
    v_horario_fim := to_char(((p_hora_inicio::time) + (v_duracao::text || ' minutes')::interval)::time, 'HH24:MI');

    if p_data = v_today then
      v_now_min := floor(extract(epoch from ((now() at time zone v_tz)::time)) / 60)::int;
      if v_start_min < v_now_min + v_min_lead_min then
        raise exception 'antecedencia_minima';
      end if;
    end if;

    if (v_start_min - v_buffer_antes) < floor(extract(epoch from v_sched_inicio) / 60)::int then
      raise exception 'fora_do_horario';
    end if;
    if (v_end_min + v_buffer_depois) > floor(extract(epoch from v_sched_fim) / 60)::int then
      raise exception 'fora_do_horario';
    end if;

    if v_iv_inicio is not null and v_iv_fim is not null then
      if (v_start_min - v_buffer_antes) < floor(extract(epoch from v_iv_fim) / 60)::int
        and floor(extract(epoch from v_iv_inicio) / 60)::int < (v_end_min + v_buffer_depois) then
        raise exception 'intervalo';
      end if;
    end if;

    if exists (
      select 1
      from public.public_get_ocupacoes(v_uid, p_data, p_funcionario_id) r
      where (v_start_min - v_buffer_antes) < r.end_min and r.start_min < (v_end_min + v_buffer_depois)
      limit 1
    ) then
      raise exception 'ocupado';
    end if;
  end if;

  insert into public.agendamentos (
    usuario_id,
    funcionario_id,
    unidade_id,
    servico_id,
    cliente_nome,
    cliente_telefone,
    extras,
    data,
    hora_inicio,
    hora_fim,
    status
  )
  values (
    v_uid,
```

```
    p_funcionario_id,
    p_unidade_id,
    p_servico_id,
    nullif(p_cliente_nome, ''),
    nullif(p_cliente_telefone, ''),
    p_extras,
    p_data,
    case when v_dia_inteiro then v_sched_inicio else p_hora_inicio::time end,
    case when v_dia_inteiro then v_sched_fim else v_horario_fim::time end,
    v_status
  )
  returning id into v_created_id;

  return v_created_id;
end;
$$;

revoke all on function public.public_get_usuario_publico(text, text) from public;
revoke all on function public.public_get_servicos_publicos(uuid) from public;
revoke all on function public.public_get_funcionarios_publicos(uuid, uuid) from public;
revoke all on function public.public_get_slots_publicos(uuid, date, uuid, uuid, uuid) from public;
revoke all on function public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, uuid, jsonb,
text) from public;

grant execute on function public.public_get_usuario_publico(text, text) to anon;
grant execute on function public.public_get_servicos_publicos(uuid) to anon;
grant execute on function public.public_get_funcionarios_publicos(uuid, uuid) to anon;
grant execute on function public.public_get_slots_publicos(uuid, date, uuid, uuid, uuid) to anon;
grant execute on function public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, uuid,
jsonb, text) to anon;

grant execute on function public.public_get_usuario_publico(text, text) to authenticated;
grant execute on function public.public_get_servicos_publicos(uuid) to authenticated;
grant execute on function public.public_get_funcionarios_publicos(uuid, uuid) to authenticated;
grant execute on function public.public_get_slots_publicos(uuid, date, uuid, uuid, uuid) to authenticated;
grant execute on function public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, uuid,
jsonb, text) to authenticated;

notify pgrst, 'reload schema';`
  }, [])

  const sqlStorageLogos = useMemo(() => {
    return `insert into storage.buckets (id, name, public)
values ('logos', 'logos', true)
on conflict (id) do update set public = true;

alter table storage.objects enable row level security;

drop policy if exists "public_read_logos" on storage.objects;
create policy "public_read_logos" on storage.objects
for select to public
using (bucket_id = 'logos');

drop policy if exists "auth_write_own_logos" on storage.objects;
create policy "auth_write_own_logos" on storage.objects
for insert to authenticated
with check (bucket_id = 'logos' and name like auth.uid()::text || '/%');

drop policy if exists "auth_update_own_logos" on storage.objects;
create policy "auth_update_own_logos" on storage.objects
for update to authenticated
using (bucket_id = 'logos' and name like auth.uid()::text || '/%')
with check (bucket_id = 'logos' and name like auth.uid()::text || '/%');

drop policy if exists "auth_delete_own_logos" on storage.objects;
create policy "auth_delete_own_logos" on storage.objects
for delete to authenticated
using (bucket_id = 'logos' and name like auth.uid()::text || '/%');`
  }, [])

  const sqlStorageServicosFotos = useMemo(() => {
    return `insert into storage.buckets (id, name, public)
```

```
values ('servicos', 'servicos', true)
on conflict (id) do update set public = true;

alter table storage.objects enable row level security;

drop policy if exists "public_read_servicos" on storage.objects;
create policy "public_read_servicos" on storage.objects
for select to public
using (bucket_id = 'servicos');

drop policy if exists "auth_write_own_servicos" on storage.objects;
create policy "auth_write_own_servicos" on storage.objects
for insert to authenticated
with check (bucket_id = 'servicos' and name like auth.uid()::text || '/%');

drop policy if exists "auth_update_own_servicos" on storage.objects;
create policy "auth_update_own_servicos" on storage.objects
for update to authenticated
using (bucket_id = 'servicos' and name like auth.uid()::text || '/%')
with check (bucket_id = 'servicos' and name like auth.uid()::text || '/%');

drop policy if exists "auth_delete_own_servicos" on storage.objects;
create policy "auth_delete_own_servicos" on storage.objects
for delete to authenticated
using (bucket_id = 'servicos' and name like auth.uid()::text || '/%');`
  }, [])

  const sqlServicosFotosPro = useMemo(() => {
    return `alter table public.servicos add column if not exists foto_url text;

create or replace function public.servicos_enforce_foto_by_plano()
returns trigger
language plpgsql
security definer
set search_path = public
as $$
declare
  v_plano text;
begin
  if public.is_super_admin() then
    return NEW;
  end if;

  if nullif(coalesce(NEW.foto_url, ''), '') is null then
    return NEW;
  end if;

  select u.plano::text into v_plano
  from public.usuarios u
  where u.id = NEW.usuario_id;

  v_plano := lower(coalesce(nullif(v_plano, ''), 'free'));
  if v_plano not in ('pro', 'team', 'enterprise') then
    raise exception 'plano_sem_foto_servico';
  end if;

  return NEW;
end;
$$;

drop trigger if exists servicos_enforce_foto_by_plano on public.servicos;
create trigger servicos_enforce_foto_by_plano
before insert or update of foto_url on public.servicos
for each row execute function public.servicos_enforce_foto_by_plano();`
  }, [])

  const sqlServicosLimiteBasic = useMemo(() => {
    return `create or replace function public.servicos_enforce_limite_by_plano()
returns trigger
language plpgsql
security definer
set search_path = public
```

```
as $$
declare
  v_plano text;
  v_count int;
begin
  if public.is_super_admin() then
    return NEW;
  end if;

  select u.plano::text into v_plano
  from public.usuarios u
  where u.id = NEW.usuario_id;

  v_plano := lower(coalesce(nullif(v_plano, ''), 'free'));
  if v_plano in ('pro', 'team', 'enterprise') then
    return NEW;
  end if;

  select count(*) into v_count
  from public.servicos s
  where s.usuario_id = NEW.usuario_id;

  if coalesce(v_count, 0) >= 3 then
    raise exception 'limite_servicos_atingido';
  end if;

  return NEW;
end;
$$;

drop trigger if exists servicos_enforce_limite_by_plano on public.servicos;
create trigger servicos_enforce_limite_by_plano
before insert on public.servicos
for each row execute function public.servicos_enforce_limite_by_plano();`
  }, [])

  const sqlMultiUnidadesEnterprise = useMemo(() => {
    return `alter table public.usuarios add column if not exists tipo_negocio text;

alter table public.servicos add column if not exists taxa_agendamento numeric not null default 0;

create table if not exists public.taxa_agendamento_pagamentos (
  id uuid primary key default gen_random_uuid(),
  criado_em timestamptz not null default now(),
  usuario_id uuid not null references public.usuarios(id) on delete cascade,
  agendamento_id uuid null references public.agendamentos(id) on delete set null,
  servico_id uuid null references public.servicos(id) on delete set null,
  funcionario_id uuid null references public.funcionarios(id) on delete set null,
  cliente_nome text null,
  cliente_telefone text null,
  valor numeric not null default 0,
  moeda text not null default 'brl',
  status text not null default 'pendente',
  stripe_checkout_session_id text null,
  stripe_payment_intent_id text null,
  pago_em timestamptz null,
  credito_em timestamptz null,
  usado_em timestamptz null,
  utilizado_em_agendamento_id uuid null references public.agendamentos(id) on delete set null
);

alter table public.taxa_agendamento_pagamentos add column if not exists unidade_id uuid references public.unidades(id)
on delete set null;

create index if not exists taxa_agendamento_pagamentos_usuario_idx on public.taxa_agendamento_pagamentos (usuario_id,
status, criado_em desc);
create index if not exists taxa_agendamento_pagamentos_cliente_idx on public.taxa_agendamento_pagamentos (usuario_id,
cliente_telefone, status, criado_em desc);
create unique index if not exists taxa_agendamento_pagamentos_stripe_session_unique on
public.taxa_agendamento_pagamentos (stripe_checkout_session_id) where stripe_checkout_session_id is not null;

alter table public.taxa_agendamento_pagamentos enable row level security;
```

```
drop policy if exists "taxa_agendamento_pagamentos_select_master_or_funcionario" on public.taxa_agendamento_pagamentos;
create policy "taxa_agendamento_pagamentos_select_master_or_funcionario" on public.taxa_agendamento_pagamentos
for select to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
  or exists (
    select 1
    from public.funcionarios f
    where f.id = auth.uid()
      and f.usuario_master_id = taxa_agendamento_pagamentos.usuario_id
      and f.ativo = true
      and f.pode_ver_financeiro = true
  )
);

grant select on public.taxa_agendamento_pagamentos to authenticated;

create or replace function public.agendamentos_tornar_taxa_credito()
returns trigger
language plpgsql
security definer
set search_path = public
as $$
begin
  if TG_OP = 'UPDATE' then
    if lower(coalesce(NEW.status::text, '')) in ('nao_compareceu', 'não_compareceu', 'no_show')
      and lower(coalesce(OLD.status::text, '')) not in ('nao_compareceu', 'não_compareceu', 'no_show') then
      update public.taxa_agendamento_pagamentos
      set status = 'credito', credito_em = now()
      where agendamento_id = NEW.id
        and status = 'pago';
    end if;
  end if;
  return NEW;
end;
$$;

drop trigger if exists agendamentos_tornar_taxa_credito_trg on public.agendamentos;
create trigger agendamentos_tornar_taxa_credito_trg
after update of status on public.agendamentos
for each row execute function public.agendamentos_tornar_taxa_credito();

create or replace function public.consume_taxa_agendamento_credito(p_usuario_id uuid, p_cliente_telefone text)
returns uuid
language plpgsql
security definer
set search_path = public
as $$
declare
  v_id uuid;
begin
  if p_usuario_id is null or nullif(trim(coalesce(p_cliente_telefone, '')), '') is null then
    return null;
  end if;

  select t.id
  into v_id
  from public.taxa_agendamento_pagamentos t
  where t.usuario_id = p_usuario_id
    and t.cliente_telefone = p_cliente_telefone
    and t.status = 'credito'
  order by t.criado_em asc
  for update skip locked
  limit 1;

  if v_id is null then
    return null;
  end if;

  update public.taxa_agendamento_pagamentos
```

```
  set status = 'reservado'
  where id = v_id;

  return v_id;
end;
$$;

revoke all on function public.consume_taxa_agendamento_credito(uuid, text) from public;
grant execute on function public.consume_taxa_agendamento_credito(uuid, text) to service_role;

drop function if exists public.public_get_servicos_publicos(uuid);

create or replace function public.public_get_servicos_publicos(p_usuario_id uuid)
returns table (
  id uuid,
  nome text,
  descricao text,
  duracao_minutos int,
  buffer_antes_min int,
  buffer_depois_min int,
  antecedencia_minutos int,
  janela_max_dias int,
  preco numeric,
  taxa_agendamento numeric,
  cor text,
  foto_url text
)
language plpgsql
security definer
set search_path = public
as $$
begin
  return query
  select
    s.id::uuid,
    s.nome::text,
    s.descricao::text,
    s.duracao_minutos::int,
    coalesce(s.buffer_antes_min, 0)::int,
    coalesce(s.buffer_depois_min, 0)::int,
    coalesce(s.antecedencia_minutos, 120)::int,
    365::int,
    s.preco::numeric,
    s.taxa_agendamento::numeric,
    s.cor::text,
    s.foto_url::text
  from public.servicos s
  where s.usuario_id = p_usuario_id
    and s.ativo = true
  order by s.ordem asc nulls last, s.criado_em asc;
end;
$$;

create table if not exists public.unidades (
  id uuid primary key default gen_random_uuid(),
  usuario_id uuid not null references public.usuarios(id) on delete cascade,
  nome text not null,
  slug text not null,
  endereco text null,
  telefone text null,
  horario_inicio text null,
  horario_fim text null,
  dias_trabalho int[] null,
  intervalo_inicio text null,
  intervalo_fim text null,
  timezone text null,
  public_primary_color text null,
  public_background_color text null,
  public_use_background_image boolean not null default false,
  public_background_image_url text null,
  ativo boolean not null default true,
  criado_em timestamptz not null default now()
```

```
);

create unique index if not exists unidades_usuario_slug_unique on public.unidades (usuario_id, slug);
create index if not exists unidades_usuario_idx on public.unidades (usuario_id);

alter table public.unidades enable row level security;

drop policy if exists "unidades_select_own" on public.unidades;
create policy "unidades_select_own" on public.unidades
for select to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
);

drop policy if exists "unidades_insert_own" on public.unidades;
create policy "unidades_insert_own" on public.unidades
for insert to authenticated
with check (
  usuario_id = auth.uid()
  or public.is_super_admin()
);

drop policy if exists "unidades_update_own" on public.unidades;
create policy "unidades_update_own" on public.unidades
for update to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
)
with check (
  usuario_id = auth.uid()
  or public.is_super_admin()
);

drop policy if exists "unidades_delete_own" on public.unidades;
create policy "unidades_delete_own" on public.unidades
for delete to authenticated
using (
  usuario_id = auth.uid()
  or public.is_super_admin()
);

grant select, insert, update, delete on public.unidades to authenticated;

alter table public.funcionarios add column if not exists unidade_id uuid references public.unidades(id) on delete set null;
alter table public.agendamentos add column if not exists unidade_id uuid references public.unidades(id) on delete set null;
alter table public.bloqueios add column if not exists unidade_id uuid references public.unidades(id) on delete set null;

create index if not exists funcionarios_unidade_idx on public.funcionarios (usuario_master_id, unidade_id);
create index if not exists agendamentos_unidade_idx on public.agendamentos (usuario_id, unidade_id, data);
create index if not exists bloqueios_unidade_idx on public.bloqueios (usuario_id, unidade_id, data);

drop function if exists public.public_get_usuario_publico(text);
drop function if exists public.public_get_usuario_publico(text, text);

create or replace function public.public_get_usuario_publico(p_slug text, p_unidade_slug text default null)
returns table (
  id uuid,
  nome_negocio text,
  logo_url text,
  endereco text,
  telefone text,
  instagram_url text,
  horario_inicio text,
  horario_fim text,
  dias_trabalho int[],
  intervalo_inicio text,
  intervalo_fim text,
  ativo boolean,
```

```
  tipo_conta text,
  plano text,
  tipo_negocio text,
  public_primary_color text,
  public_background_color text,
  public_use_background_image boolean,
  public_background_image_url text,
  unidade_id uuid,
  unidade_nome text,
  unidade_slug text
)
language plpgsql
security definer
set search_path = public
as $$
begin
  return query
  with base as (
    select u.*
    from public.usuarios u
    where u.slug = p_slug
      and u.ativo = true
    limit 1
  ), uni as (
    select un.*
    from public.unidades un
    join base b on b.id = un.usuario_id
    where p_unidade_slug is not null
      and un.slug = p_unidade_slug
      and un.ativo = true
    limit 1
  )
  select
    b.id::uuid,
    (case when u.id is not null then u.nome::text else b.nome_negocio::text end) as nome_negocio,
    b.logo_url::text,
    coalesce(u.endereco::text, b.endereco::text) as endereco,
    coalesce(u.telefone::text, b.telefone::text) as telefone,
    b.instagram_url::text,
    coalesce(u.horario_inicio::text, b.horario_inicio::text) as horario_inicio,
    coalesce(u.horario_fim::text, b.horario_fim::text) as horario_fim,
    coalesce(u.dias_trabalho::int[], b.dias_trabalho::int[]) as dias_trabalho,
    coalesce(u.intervalo_inicio::text, b.intervalo_inicio::text) as intervalo_inicio,
    coalesce(u.intervalo_fim::text, b.intervalo_fim::text) as intervalo_fim,
    b.ativo::boolean,
    b.tipo_conta::text,
    b.plano::text,
    b.tipo_negocio::text,
    coalesce(u.public_primary_color::text, b.public_primary_color::text) as public_primary_color,
    coalesce(u.public_background_color::text, b.public_background_color::text) as public_background_color,
    coalesce(u.public_use_background_image::boolean, b.public_use_background_image::boolean) as
public_use_background_image,
    coalesce(u.public_background_image_url::text, b.public_background_image_url::text) as public_background_image_url,
    u.id::uuid as unidade_id,
    u.nome::text as unidade_nome,
    u.slug::text as unidade_slug
  from base b
  left join uni u on true;
end;
$$;

drop function if exists public.public_get_funcionarios_publicos(uuid);
drop function if exists public.public_get_funcionarios_publicos(uuid, uuid);

create or replace function public.public_get_funcionarios_publicos(p_usuario_master_id uuid, p_unidade_id uuid default
null)
returns table (
  id uuid,
  nome_completo text,
  horario_inicio text,
  horario_fim text,
  dias_trabalho int[],
```

```
  intervalo_inicio text,
  intervalo_fim text
)
language plpgsql
security definer
set search_path = public
as $$
begin
  return query
  select f.id::uuid, f.nome_completo::text, f.horario_inicio::text, f.horario_fim::text, f.dias_trabalho::int[],
f.intervalo_inicio::text, f.intervalo_fim::text
  from public.funcionarios f
  where f.usuario_master_id = p_usuario_master_id
    and f.ativo = true
    and lower(trim(f.permissao)) = 'funcionario'
    and (p_unidade_id is null or f.unidade_id = p_unidade_id)
  order by f.criado_em asc;
end;
$$;

drop function if exists public.public_get_ocupacoes(uuid, date, uuid);
drop function if exists public.public_get_ocupacoes(uuid, date, uuid, uuid);

create or replace function public.public_get_ocupacoes(p_usuario_id uuid, p_data date, p_funcionario_id uuid,
p_unidade_id uuid default null)
returns table (start_min int, end_min int)
language plpgsql
security definer
set search_path = public
as $$
begin
  return query
  select
    (floor(extract(epoch from a.hora_inicio::time) / 60)::int - coalesce(s.buffer_antes_min, 0)::int) as start_min,
    (floor(extract(epoch from coalesce(a.hora_fim, a.hora_inicio)::time) / 60)::int + coalesce(s.buffer_depois_min,
0)::int) as end_min
  from public.agendamentos a
  left join public.servicos s on s.id = a.servico_id and s.usuario_id = a.usuario_id
  where a.usuario_id = p_usuario_id
    and a.data = p_data
    and a.status <> 'cancelado'
    and (p_unidade_id is null and a.unidade_id is null or p_unidade_id is not null and a.unidade_id = p_unidade_id)
    and (p_funcionario_id is null or a.funcionario_id is null or a.funcionario_id = p_funcionario_id)

  union all

  select
    floor(extract(epoch from b.hora_inicio::time) / 60)::int as start_min,
    floor(extract(epoch from b.hora_fim::time) / 60)::int as end_min
  from public.bloqueios b
  where b.usuario_id = p_usuario_id
    and b.data = p_data
    and (p_unidade_id is null and b.unidade_id is null or p_unidade_id is not null and b.unidade_id = p_unidade_id)
    and (p_funcionario_id is null or b.funcionario_id is null or b.funcionario_id = p_funcionario_id);
end;
$$;

drop function if exists public.public_get_slots_publicos(uuid, date, uuid, uuid);
drop function if exists public.public_get_slots_publicos(uuid, date, uuid, uuid, uuid);

create or replace function public.public_get_slots_publicos(
  p_usuario_id uuid,
  p_data date,
  p_servico_id uuid,
  p_funcionario_id uuid,
  p_unidade_id uuid default null
)
returns table (hora_inicio text)
language plpgsql
security definer
set search_path = public
as $$
```

```
declare
  v_uid uuid;
  v_base record;
  v_staff_exists boolean;
  v_staff_horario_inicio time;
  v_staff_horario_fim time;
  v_staff_dias_trabalho int[];
  v_staff_intervalo_inicio time;
  v_staff_intervalo_fim time;
  v_duracao int;
  v_dia_inteiro boolean;
  v_weekday int;
  v_sched_inicio time;
  v_sched_fim time;
  v_iv_inicio time;
  v_iv_fim time;
  v_sched_inicio_min int;
  v_sched_fim_min int;
  v_iv_inicio_min int;
  v_iv_fim_min int;
  v_start_min int;
  v_end_min int;
  v_now_min int;
  v_step_min int := 30;
  v_min_lead_min int;
  v_max_days int;
  v_buffer_antes int;
  v_buffer_depois int;
  v_tz text;
  v_today date;
  v_unidade_ok boolean;
  v_cap int;
  v_diaria_count int;
begin
  if p_usuario_id is null or p_data is null or p_servico_id is null then
    raise exception 'invalid_payload';
  end if;

  v_uid := p_usuario_id;

  if p_unidade_id is not null then
    select true into v_unidade_ok
    from public.unidades un
    where un.id = p_unidade_id
      and un.usuario_id = v_uid
      and un.ativo = true;
    if v_unidade_ok is distinct from true then
      raise exception 'unidade_invalida';
    end if;
  end if;

  select
    coalesce(un.horario_inicio::text, u.horario_inicio::text) as horario_inicio,
    coalesce(un.horario_fim::text, u.horario_fim::text) as horario_fim,
    coalesce(un.dias_trabalho, u.dias_trabalho) as dias_trabalho,
    coalesce(un.intervalo_inicio::text, u.intervalo_inicio::text) as intervalo_inicio,
    coalesce(un.intervalo_fim::text, u.intervalo_fim::text) as intervalo_fim,
    coalesce(un.timezone::text, u.timezone::text) as timezone
  into v_base
  from public.usuarios u
  left join public.unidades un
    on un.id = p_unidade_id
    and un.usuario_id = u.id
    and un.ativo = true
  where u.id = v_uid
    and u.ativo = true;

  if v_base is null then
    raise exception 'usuario_invalido';
  end if;

  v_tz := coalesce(nullif(v_base.timezone::text, ''), 'America/Sao_Paulo');
```

```
  v_today := (now() at time zone v_tz)::date;

  if p_data < v_today then
    raise exception 'data_passada';
  end if;

  select
    s.duracao_minutos,
    coalesce(s.buffer_antes_min, 0)::int,
    coalesce(s.buffer_depois_min, 0)::int,
    coalesce(s.antecedencia_minutos, 120)::int,
    365::int,
    coalesce(s.dia_inteiro, false)::boolean
  into v_duracao, v_buffer_antes, v_buffer_depois, v_min_lead_min, v_max_days, v_dia_inteiro
  from public.servicos s
  where s.id = p_servico_id
    and s.usuario_id = v_uid
    and s.ativo = true;

  if v_duracao is null or v_duracao <= 0 then
    raise exception 'servico_invalido';
  end if;

  if p_data > (v_today + v_max_days) then
    raise exception 'data_muito_futura';
  end if;

  if p_funcionario_id is not null then
    select true, f.horario_inicio, f.horario_fim, f.dias_trabalho, f.intervalo_inicio, f.intervalo_fim
    into v_staff_exists, v_staff_horario_inicio, v_staff_horario_fim, v_staff_dias_trabalho, v_staff_intervalo_inicio,
v_staff_intervalo_fim
    from public.funcionarios f
    where f.id = p_funcionario_id
      and f.usuario_master_id = v_uid
      and f.ativo = true
      and (p_unidade_id is null or f.unidade_id = p_unidade_id);

    if v_staff_exists is distinct from true then
      raise exception 'funcionario_invalido';
    end if;
  end if;

  v_sched_inicio := coalesce(v_staff_horario_inicio, v_base.horario_inicio::time);
  v_sched_fim := coalesce(v_staff_horario_fim, v_base.horario_fim::time);
  v_iv_inicio := coalesce(v_staff_intervalo_inicio, v_base.intervalo_inicio::time);
  v_iv_fim := coalesce(v_staff_intervalo_fim, v_base.intervalo_fim::time);

  if v_sched_inicio is null or v_sched_fim is null then
    raise exception 'horarios_nao_configurados';
  end if;

  v_weekday := extract(dow from p_data)::int;
  if p_funcionario_id is not null then
    if v_staff_dias_trabalho is not null and array_length(v_staff_dias_trabalho, 1) > 0 then
      if not (v_weekday = any(v_staff_dias_trabalho)) then
        raise exception 'fora_do_dia_de_trabalho';
      end if;
    end if;
  else
    if v_base.dias_trabalho is not null and array_length(v_base.dias_trabalho, 1) > 0 then
      if not (v_weekday = any(v_base.dias_trabalho)) then
        raise exception 'fora_do_dia_de_trabalho';
      end if;
    end if;
  end if;

  v_sched_inicio_min := floor(extract(epoch from v_sched_inicio) / 60)::int;
  v_sched_fim_min := floor(extract(epoch from v_sched_fim) / 60)::int;
  v_iv_inicio_min := case when v_iv_inicio is null then null else floor(extract(epoch from v_iv_inicio) / 60)::int end;
  v_iv_fim_min := case when v_iv_fim is null then null else floor(extract(epoch from v_iv_fim) / 60)::int end;

  if v_iv_inicio_min is not null and v_iv_fim_min is not null and v_iv_inicio_min >= v_iv_fim_min then
```

```
      v_iv_inicio_min := null;
      v_iv_fim_min := null;
    end if;

    v_start_min := v_sched_inicio_min + v_buffer_antes;
    v_end_min := v_sched_fim_min - v_duracao - v_buffer_depois;

    if v_dia_inteiro then
      if v_sched_fim_min <= v_sched_inicio_min then
        return;
      end if;

      if p_data = v_today then
        v_now_min := floor(extract(epoch from (now() at time zone v_tz)::time) / 60)::int;
        if v_sched_inicio_min < (v_now_min + v_min_lead_min) then
          return;
        end if;
      end if;

      v_cap := 1;
      if p_funcionario_id is not null then
        select coalesce(f.capacidade_dia_inteiro, 1)::int
        into v_cap
        from public.funcionarios f
        where f.id = p_funcionario_id
          and f.usuario_master_id = v_uid
          and f.ativo = true;
      end if;
      if v_cap < 1 then v_cap := 1; end if;
      if v_cap > 2 then v_cap := 2; end if;

      if exists (
        select 1
        from public.bloqueios b
        where b.usuario_id = v_uid
          and b.data = p_data
          and (p_unidade_id is null and b.unidade_id is null or p_unidade_id is not null and b.unidade_id = p_unidade_id)
          and (p_funcionario_id is null or b.funcionario_id is null or b.funcionario_id = p_funcionario_id)
          and floor(extract(epoch from b.hora_inicio::time) / 60)::int < v_sched_fim_min
          and v_sched_inicio_min < floor(extract(epoch from b.hora_fim::time) / 60)::int
        limit 1
      ) then
        return;
      end if;

      if exists (
        select 1
        from public.agendamentos a
        join public.servicos s2 on s2.id = a.servico_id and s2.usuario_id = a.usuario_id
        where a.usuario_id = v_uid
          and a.data = p_data
          and a.status <> 'cancelado'
          and (p_unidade_id is null and a.unidade_id is null or p_unidade_id is not null and a.unidade_id = p_unidade_id)
          and (p_funcionario_id is null or a.funcionario_id is null or a.funcionario_id = p_funcionario_id)
          and coalesce(s2.dia_inteiro, false) = false
        limit 1
      ) then
        return;
      end if;

      select count(*)::int
      into v_diaria_count
      from public.agendamentos a
      join public.servicos s2 on s2.id = a.servico_id and s2.usuario_id = a.usuario_id
      where a.usuario_id = v_uid
        and a.data = p_data
        and a.status <> 'cancelado'
        and (p_unidade_id is null and a.unidade_id is null or p_unidade_id is not null and a.unidade_id = p_unidade_id)
        and (p_funcionario_id is null or a.funcionario_id is null or a.funcionario_id = p_funcionario_id)
        and coalesce(s2.dia_inteiro, false) = true;

      if coalesce(v_diaria_count, 0) >= v_cap then
```

```
      return;
    end if;

    return query
    select to_char(v_sched_inicio, 'HH24:MI') as hora_inicio;
    return;
  end if;

  if p_data = v_today then
    v_now_min := floor(extract(epoch from (now() at time zone v_tz)::time) / 60)::int;
    v_start_min := greatest(v_start_min, v_now_min + v_min_lead_min);
  end if;

  v_start_min := ((v_start_min + v_step_min - 1) / v_step_min) * v_step_min;

  return query
  with candidates as (
    select gs as start_min
    from generate_series(v_start_min, v_end_min, v_step_min) gs
  ),
  without_interval as (
    select c.start_min
    from candidates c
    where not (
      v_iv_inicio_min is not null
      and v_iv_fim_min is not null
      and (c.start_min - v_buffer_antes) < v_iv_fim_min
      and v_iv_inicio_min < (c.start_min + v_duracao + v_buffer_depois)
    )
  ),
  free as (
    select w.start_min
    from without_interval w
    where not exists (
      select 1
      from public.public_get_ocupacoes(v_uid, p_data, p_funcionario_id, p_unidade_id) r
      where (w.start_min - v_buffer_antes) < r.end_min and r.start_min < (w.start_min + v_duracao + v_buffer_depois)
      limit 1
    )
  )
  select to_char((time '00:00' + (free.start_min::text || ' minutes')::interval)::time, 'HH24:MI') as hora_inicio
  from free
  order by free.start_min asc;
end;
$$;

drop function if exists public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid);
drop function if exists public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, uuid);

create or replace function public.public_create_agendamento_publico(
  p_usuario_id uuid,
  p_data date,
  p_hora_inicio text,
  p_servico_id uuid,
  p_cliente_nome text,
  p_cliente_telefone text,
  p_funcionario_id uuid,
  p_unidade_id uuid default null,
  p_extras jsonb default null,
  p_status text default 'pendente'
)
returns uuid
language plpgsql
security definer
set search_path = public
as $$
declare
  v_uid uuid;
  v_base record;
  v_staff_exists boolean;
  v_staff_horario_inicio time;
  v_staff_horario_fim time;
```

```
    v_staff_dias_trabalho int[];
    v_staff_intervalo_inicio time;
    v_staff_intervalo_fim time;
    v_duracao int;
    v_dia_inteiro boolean;
    v_start_min int;
    v_end_min int;
    v_weekday int;
    v_sched_inicio time;
    v_sched_fim time;
    v_iv_inicio time;
    v_iv_fim time;
    v_horario_fim text;
    v_created_id uuid;
    v_now_min int;
    v_min_lead_min int;
    v_max_days int;
    v_buffer_antes int;
    v_buffer_depois int;
    v_tz text;
    v_today date;
    v_limite_mes int;
    v_month_start date;
    v_month_end date;
    v_month_count int;
    v_unidade_ok boolean;
    v_status text;
    v_cap int;
    v_diaria_count int;
  begin
    if p_usuario_id is null or p_data is null or nullif(p_hora_inicio, '') is null or p_servico_id is null then
      raise exception 'invalid_payload';
    end if;

    v_status := lower(coalesce(nullif(trim(coalesce(p_status, '')), ''), 'pendente'));
    if v_status not in ('confirmado', 'pendente') then
      raise exception 'status_invalido';
    end if;

    v_uid := p_usuario_id;

    select
      coalesce(un.horario_inicio::text, u.horario_inicio::text) as horario_inicio,
      coalesce(un.horario_fim::text, u.horario_fim::text) as horario_fim,
      coalesce(un.dias_trabalho, u.dias_trabalho) as dias_trabalho,
      coalesce(un.intervalo_inicio::text, u.intervalo_inicio::text) as intervalo_inicio,
      coalesce(un.intervalo_fim::text, u.intervalo_fim::text) as intervalo_fim,
      coalesce(un.timezone::text, u.timezone::text) as timezone,
      u.limite_agendamentos_mes as limite_agendamentos_mes
    into v_base
    from public.usuarios u
    left join public.unidades un
      on un.id = p_unidade_id
      and un.usuario_id = u.id
      and un.ativo = true
    where u.id = v_uid
      and u.ativo = true;

    if v_base is null then
      raise exception 'usuario_invalido';
    end if;

    if p_unidade_id is not null then
      select true into v_unidade_ok
      from public.unidades un
      where un.id = p_unidade_id
        and un.usuario_id = v_uid
        and un.ativo = true;
      if v_unidade_ok is distinct from true then
        raise exception 'unidade_invalida';
      end if;
    end if;
```

```
  v_tz := coalesce(nullif(v_base.timezone::text, ''), 'America/Sao_Paulo');
  v_today := (now() at time zone v_tz)::date;

  if p_data < v_today then
    raise exception 'data_passada';
  end if;

  v_weekday := extract(dow from p_data)::int;

  select
    s.duracao_minutos,
    coalesce(s.buffer_antes_min, 0)::int,
    coalesce(s.buffer_depois_min, 0)::int,
    coalesce(s.antecedencia_minutos, 120)::int,
    365::int,
    coalesce(s.dia_inteiro, false)::boolean
  into v_duracao, v_buffer_antes, v_buffer_depois, v_min_lead_min, v_max_days, v_dia_inteiro
  from public.servicos s
  where s.id = p_servico_id
    and s.usuario_id = v_uid
    and s.ativo = true;

  if v_duracao is null or v_duracao <= 0 then
    raise exception 'servico_invalido';
  end if;

  if p_data > (v_today + v_max_days) then
    raise exception 'data_muito_futura';
  end if;

  if p_funcionario_id is not null then
    select true, f.horario_inicio, f.horario_fim, f.dias_trabalho, f.intervalo_inicio, f.intervalo_fim
    into v_staff_exists, v_staff_horario_inicio, v_staff_horario_fim, v_staff_dias_trabalho, v_staff_intervalo_inicio,
v_staff_intervalo_fim
    from public.funcionarios f
    where f.id = p_funcionario_id
      and f.usuario_master_id = v_uid
      and f.ativo = true
      and (p_unidade_id is null or f.unidade_id = p_unidade_id);

    if v_staff_exists is distinct from true then
      raise exception 'funcionario_invalido';
    end if;
  end if;

  if p_funcionario_id is not null then
    if v_staff_dias_trabalho is not null and array_length(v_staff_dias_trabalho, 1) > 0 then
      if not (v_weekday = any(v_staff_dias_trabalho)) then
        raise exception 'fora_do_dia_de_trabalho';
      end if;
    end if;
  else
    if v_base.dias_trabalho is not null and array_length(v_base.dias_trabalho, 1) > 0 then
      if not (v_weekday = any(v_base.dias_trabalho)) then
        raise exception 'fora_do_dia_de_trabalho';
      end if;
    end if;
  end if;

  v_sched_inicio := coalesce(v_staff_horario_inicio, v_base.horario_inicio::time);
  v_sched_fim := coalesce(v_staff_horario_fim, v_base.horario_fim::time);
  v_iv_inicio := coalesce(v_staff_intervalo_inicio, v_base.intervalo_inicio::time);
  v_iv_fim := coalesce(v_staff_intervalo_fim, v_base.intervalo_fim::time);

  if v_sched_inicio is null or v_sched_fim is null then
    raise exception 'horarios_nao_configurados';
  end if;

  if v_dia_inteiro then
    v_start_min := floor(extract(epoch from v_sched_inicio) / 60)::int;
    v_end_min := floor(extract(epoch from v_sched_fim) / 60)::int;
```

```
    v_horario_fim := to_char(v_sched_fim, 'HH24:MI');

    if v_end_min <= v_start_min then
      raise exception 'fora_do_horario';
    end if;

    if p_data = v_today then
      v_now_min := floor(extract(epoch from (now() at time zone v_tz)::time) / 60)::int;
      if v_start_min < (v_now_min + v_min_lead_min) then
        raise exception 'antecedencia_minima';
      end if;
    end if;

    v_cap := 1;
    if p_funcionario_id is not null then
      select coalesce(f.capacidade_dia_inteiro, 1)::int
      into v_cap
      from public.funcionarios f
      where f.id = p_funcionario_id
        and f.usuario_master_id = v_uid
        and f.ativo = true;
    end if;
    if v_cap < 1 then v_cap := 1; end if;
    if v_cap > 2 then v_cap := 2; end if;

    if exists (
      select 1
      from public.bloqueios b
      where b.usuario_id = v_uid
        and b.data = p_data
        and (p_unidade_id is null and b.unidade_id is null or p_unidade_id is not null and b.unidade_id = p_unidade_id)
        and (p_funcionario_id is null or b.funcionario_id is null or b.funcionario_id = p_funcionario_id)
        and floor(extract(epoch from b.hora_inicio::time) / 60)::int < v_end_min
        and v_start_min < floor(extract(epoch from b.hora_fim::time) / 60)::int
      limit 1
    ) then
      raise exception 'ocupado';
    end if;

    if exists (
      select 1
      from public.agendamentos a
      join public.servicos s2 on s2.id = a.servico_id and s2.usuario_id = a.usuario_id
      where a.usuario_id = v_uid
        and a.data = p_data
        and a.status <> 'cancelado'
        and (p_unidade_id is null and a.unidade_id is null or p_unidade_id is not null and a.unidade_id = p_unidade_id)
        and (p_funcionario_id is null or a.funcionario_id is null or a.funcionario_id = p_funcionario_id)
        and coalesce(s2.dia_inteiro, false) = false
      limit 1
    ) then
      raise exception 'ocupado';
    end if;

    select count(*)::int
    into v_diaria_count
    from public.agendamentos a
    join public.servicos s2 on s2.id = a.servico_id and s2.usuario_id = a.usuario_id
    where a.usuario_id = v_uid
      and a.data = p_data
      and a.status <> 'cancelado'
      and (p_unidade_id is null and a.unidade_id is null or p_unidade_id is not null and a.unidade_id = p_unidade_id)
      and (p_funcionario_id is null or a.funcionario_id is null or a.funcionario_id = p_funcionario_id)
      and coalesce(s2.dia_inteiro, false) = true;

    if coalesce(v_diaria_count, 0) >= v_cap then
      raise exception 'ocupado';
    end if;
  else
    v_start_min := floor(extract(epoch from p_hora_inicio::time) / 60)::int;
    v_end_min := v_start_min + v_duracao;
    v_horario_fim := to_char((time '00:00' + (v_end_min::text || ' minutes')::interval)::time, 'HH24:MI');
```

```
    if p_data = v_today then
      v_now_min := floor(extract(epoch from (now() at time zone v_tz)::time) / 60)::int;
      if v_start_min < (v_now_min + v_min_lead_min) then
        raise exception 'antecedencia_minima';
      end if;
    end if;

    if (v_start_min - v_buffer_antes) < floor(extract(epoch from v_sched_inicio) / 60)::int or (v_end_min +
v_buffer_depois) > floor(extract(epoch from v_sched_fim) / 60)::int then
      raise exception 'fora_do_horario';
    end if;

    if v_iv_inicio is not null and v_iv_fim is not null then
      if (v_start_min - v_buffer_antes) < floor(extract(epoch from v_iv_fim) / 60)::int and floor(extract(epoch from
v_iv_inicio) / 60)::int < (v_end_min + v_buffer_depois) then
        raise exception 'em_intervalo';
      end if;
    end if;

    if exists (
      select 1
      from public.public_get_ocupacoes(v_uid, p_data, p_funcionario_id, p_unidade_id) r
      where (v_start_min - v_buffer_antes) < r.end_min and r.start_min < (v_end_min + v_buffer_depois)
      limit 1
    ) then
      raise exception 'ocupado';
    end if;
  end if;

  v_limite_mes := v_base.limite_agendamentos_mes;
  if v_limite_mes is not null then
    v_month_start := date_trunc('month', p_data)::date;
    v_month_end := (date_trunc('month', p_data) + interval '1 month - 1 day')::date;
    select count(*) into v_month_count
    from public.agendamentos a
    where a.usuario_id = v_uid
      and a.status <> 'cancelado'
      and a.data >= v_month_start
      and a.data <= v_month_end;
    if v_month_count >= v_limite_mes then
      raise exception 'limite_mensal_atingido';
    end if;
  end if;

  insert into public.agendamentos (
    usuario_id,
    funcionario_id,
    unidade_id,
    servico_id,
    cliente_nome,
    cliente_telefone,
    extras,
    data,
    hora_inicio,
    hora_fim,
    status
  )
  values (
    v_uid,
    p_funcionario_id,
    p_unidade_id,
    p_servico_id,
    nullif(p_cliente_nome, ''),
    nullif(p_cliente_telefone, ''),
    p_extras,
    p_data,
    case when v_dia_inteiro then v_sched_inicio else p_hora_inicio::time end,
    case when v_dia_inteiro then v_sched_fim else v_horario_fim::time end,
    v_status
  )
  returning id into v_created_id;
```

```
  return v_created_id;
end;
$$;

revoke all on function public.public_get_usuario_publico(text, text) from public;
revoke all on function public.public_get_funcionarios_publicos(uuid, uuid) from public;
revoke all on function public.public_get_ocupacoes(uuid, date, uuid, uuid) from public;
revoke all on function public.public_get_slots_publicos(uuid, date, uuid, uuid, uuid) from public;
revoke all on function public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, uuid, jsonb,
text) from public;

grant execute on function public.public_get_usuario_publico(text, text) to anon;
grant execute on function public.public_get_funcionarios_publicos(uuid, uuid) to anon;
grant execute on function public.public_get_ocupacoes(uuid, date, uuid, uuid) to anon;
grant execute on function public.public_get_slots_publicos(uuid, date, uuid, uuid, uuid) to anon;
grant execute on function public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, uuid,
jsonb, text) to anon;

grant execute on function public.public_get_usuario_publico(text, text) to authenticated;
grant execute on function public.public_get_funcionarios_publicos(uuid, uuid) to authenticated;
grant execute on function public.public_get_ocupacoes(uuid, date, uuid, uuid) to authenticated;
grant execute on function public.public_get_slots_publicos(uuid, date, uuid, uuid, uuid) to authenticated;
grant execute on function public.public_create_agendamento_publico(uuid, date, text, uuid, text, text, uuid, uuid,
jsonb, text) to authenticated;`
  }, [])

  const sqlWhatsappAutomacao = useMemo(() => {
    return `alter table public.usuarios add column if not exists whatsapp_instance_name text;
alter table public.usuarios add column if not exists enviar_confirmacao boolean not null default true;
alter table public.usuarios add column if not exists enviar_lembrete boolean not null default false;
alter table public.usuarios add column if not exists enviar_cancelamento boolean not null default true;
alter table public.usuarios add column if not exists lembrete_horas_antes int not null default 24;
alter table public.usuarios add column if not exists mensagem_confirmacao text;
alter table public.usuarios add column if not exists mensagem_lembrete text;
alter table public.usuarios add column if not exists mensagem_cancelamento text;

alter table public.agendamentos add column if not exists confirmacao_enviada boolean not null default false;
alter table public.agendamentos add column if not exists confirmacao_enviada_em timestamptz;
alter table public.agendamentos add column if not exists lembrete_enviado boolean not null default false;
alter table public.agendamentos add column if not exists lembrete_enviado_em timestamptz;

create index if not exists agendamentos_confirmacao_idx on public.agendamentos (usuario_id, status,
confirmacao_enviada);
create index if not exists agendamentos_lembrete_idx on public.agendamentos (usuario_id, status, lembrete_enviado,
data);

update public.usuarios
set whatsapp_instance_name = coalesce(whatsapp_instance_name, slug)
where whatsapp_instance_name is null and slug is not null;`
  }, [])

  const sqlWhatsappHabilitacao = useMemo(() => {
    return `alter table public.usuarios add column if not exists whatsapp_habilitado boolean not null default false;

create index if not exists usuarios_whatsapp_habilitado_idx on public.usuarios (whatsapp_habilitado);

create or replace function public.usuarios_block_whatsapp_update()
returns trigger
language plpgsql
security definer
set search_path = public
as $$
begin
  if public.is_super_admin() then
    return NEW;
  end if;

  if NEW.whatsapp_habilitado is distinct from OLD.whatsapp_habilitado
    or NEW.whatsapp_api_url is distinct from OLD.whatsapp_api_url
    or NEW.whatsapp_api_key is distinct from OLD.whatsapp_api_key then
    raise exception 'not_allowed';
```

```
  end if;

  return NEW;
end;
$$;

drop trigger if exists usuarios_block_whatsapp_update on public.usuarios;
create trigger usuarios_block_whatsapp_update
before update on public.usuarios
for each row execute function public.usuarios_block_whatsapp_update();`
  }, [])

  const sqlTemaProspectorHabilitacao = useMemo(() => {
    return `alter table public.usuarios add column if not exists tema_prospector_habilitado boolean not null default
false;

create index if not exists usuarios_tema_prospector_habilitado_idx on public.usuarios (tema_prospector_habilitado);

create or replace function public.usuarios_block_tema_prospector_update()
returns trigger
language plpgsql
security definer
set search_path = public
as $$
begin
  if public.is_super_admin() then
    return NEW;
  end if;

  if NEW.tema_prospector_habilitado is distinct from OLD.tema_prospector_habilitado then
    raise exception 'not_allowed';
  end if;

  return NEW;
end;
$$;

drop trigger if exists usuarios_block_tema_prospector_update on public.usuarios;
create trigger usuarios_block_tema_prospector_update
before update on public.usuarios
for each row execute function public.usuarios_block_tema_prospector_update();`
  }, [])

  const sqlWhatsappSuperAdmin = useMemo(() => {
    return `create table if not exists public.super_admin (
  id uuid primary key references auth.users(id) on delete cascade,
  nome text not null default 'Super Admin',
  email text null,
  nivel text not null default 'super_admin',
  criado_em timestamptz not null default now(),
  whatsapp_api_url text,
  whatsapp_api_key text,
  whatsapp_instance_name text
);

alter table public.super_admin enable row level security;

drop policy if exists "super_admin_self_select" on public.super_admin;
create policy "super_admin_self_select" on public.super_admin
for select to authenticated
using (id = auth.uid());

drop policy if exists "super_admin_self_insert" on public.super_admin;
create policy "super_admin_self_insert" on public.super_admin
for insert to authenticated
with check (id = auth.uid());

drop policy if exists "super_admin_self_update" on public.super_admin;
create policy "super_admin_self_update" on public.super_admin
for update to authenticated
using (id = auth.uid())
with check (id = auth.uid());
```

```
grant select, insert, update on public.super_admin to authenticated;

create or replace function public.is_super_admin() returns boolean
language sql stable as $$
  select exists (select 1 from public.super_admin sa where sa.id = auth.uid())
$$;

alter table public.usuarios enable row level security;

drop policy if exists "super_admin_select_usuarios" on public.usuarios;
create policy "super_admin_select_usuarios" on public.usuarios
for select to authenticated
using (public.is_super_admin());

grant select on public.usuarios to authenticated;`
  }, [])

  const sqlWhatsappGlobalConfig = useMemo(() => {
    return `create table if not exists public.super_admin (
  id uuid primary key references auth.users(id) on delete cascade,
  nome text not null default 'Super Admin',
  email text null,
  nivel text not null default 'super_admin',
  criado_em timestamptz not null default now(),
  whatsapp_api_url text,
  whatsapp_api_key text,
  whatsapp_instance_name text
);

alter table public.super_admin enable row level security;

drop policy if exists "super_admin_self_select" on public.super_admin;
create policy "super_admin_self_select" on public.super_admin
for select to authenticated
using (id = auth.uid());

drop policy if exists "super_admin_self_insert" on public.super_admin;
create policy "super_admin_self_insert" on public.super_admin
for insert to authenticated
with check (id = auth.uid());

drop policy if exists "super_admin_self_update" on public.super_admin;
create policy "super_admin_self_update" on public.super_admin
for update to authenticated
using (id = auth.uid())
with check (id = auth.uid());

grant select, insert, update on public.super_admin to authenticated;

create or replace function public.is_super_admin() returns boolean
language sql stable as $$
  select exists (select 1 from public.super_admin sa where sa.id = auth.uid())
$$;

alter table public.usuarios add column if not exists whatsapp_habilitado boolean not null default false;
alter table public.usuarios add column if not exists whatsapp_instance_name text;
alter table public.usuarios add column if not exists enviar_confirmacao boolean not null default true;
alter table public.usuarios add column if not exists enviar_lembrete boolean not null default false;
alter table public.usuarios add column if not exists enviar_cancelamento boolean not null default true;
alter table public.usuarios add column if not exists lembrete_horas_antes int not null default 24;
alter table public.usuarios add column if not exists mensagem_confirmacao text;
alter table public.usuarios add column if not exists mensagem_lembrete text;
alter table public.usuarios add column if not exists mensagem_cancelamento text;

create index if not exists usuarios_whatsapp_habilitado_idx on public.usuarios (whatsapp_habilitado);

do $$
begin
  if exists (
    select 1
    from information_schema.columns
```

```
    where table_schema = 'public'
      and table_name = 'usuarios'
      and column_name = 'whatsapp_api_url'
  ) and exists (
    select 1
    from information_schema.columns
    where table_schema = 'public'
      and table_name = 'usuarios'
      and column_name = 'whatsapp_api_key'
  ) then
    execute 'drop trigger if exists usuarios_block_whatsapp_update on public.usuarios';
    execute 'update public.super_admin sa set whatsapp_api_url = coalesce(nullif(sa.whatsapp_api_url, ''''),
u.whatsapp_api_url), whatsapp_api_key=[REDACTED]''''), u.whatsapp_api_key) from (select whatsapp_api_url,
whatsapp_api_key from public.usuarios where whatsapp_api_url is not null and whatsapp_api_key is not null and
nullif(whatsapp_api_url, '''') is not null and nullif(whatsapp_api_key, '''') is not null limit 1) u where
(sa.whatsapp_api_url is null or sa.whatsapp_api_url = '''') and (sa.whatsapp_api_key is null or sa.whatsapp_api_key=
[REDACTED]'''')';
    execute 'update public.usuarios set whatsapp_api_url = null, whatsapp_api_key=[REDACTED]';

    execute $sql$
      create or replace function public.usuarios_block_whatsapp_update()
      returns trigger
      language plpgsql
      security definer
      set search_path = public
      as $func$
      begin
        if public.is_super_admin() then
          return NEW;
        end if;

        if NEW.whatsapp_habilitado is distinct from OLD.whatsapp_habilitado
          or NEW.whatsapp_api_url is distinct from OLD.whatsapp_api_url
          or NEW.whatsapp_api_key is distinct from OLD.whatsapp_api_key then
          raise exception 'not_allowed';
        end if;

        return NEW;
      end;
      $func$;
    $sql$;
    execute 'create trigger usuarios_block_whatsapp_update before update on public.usuarios for each row execute
function public.usuarios_block_whatsapp_update()';
  else
    execute 'drop trigger if exists usuarios_block_whatsapp_update on public.usuarios';
    execute $sql$
      create or replace function public.usuarios_block_whatsapp_update()
      returns trigger
      language plpgsql
      security definer
      set search_path = public
      as $func$
      begin
        if public.is_super_admin() then
          return NEW;
        end if;

        if NEW.whatsapp_habilitado is distinct from OLD.whatsapp_habilitado then
          raise exception 'not_allowed';
        end if;

        return NEW;
      end;
      $func$;
    $sql$;
    execute 'create trigger usuarios_block_whatsapp_update before update on public.usuarios for each row execute
function public.usuarios_block_whatsapp_update()';
  end if;
end;
$$;`
  }, [])
```

```
  const sqlFuncionarioHorarios = useMemo(() => {
    return `drop function if exists public.funcionario_update_horarios(text, text, int[], text, text);
drop function if exists public.funcionario_update_horarios(text, text, int[], text, text, int);

create or replace function public.funcionario_update_horarios(
  p_horario_inicio text,
  p_horario_fim text,
  p_dias_trabalho int[],
  p_intervalo_inicio text default null,
  p_intervalo_fim text default null,
  p_capacidade_dia_inteiro int default null
) returns void
language plpgsql
security definer
set search_path = public
as $$
begin
  raise exception 'not_allowed';
end;
$$;

revoke all on function public.funcionario_update_horarios(text, text, int[], text, text, int) from public;
revoke all on function public.funcionario_update_horarios(text, text, int[], text, text, int) from authenticated;`
  }, [])

  const sqlWhatsappLembretesCron = useMemo(() => {
    return `create extension if not exists pg_net;
create extension if not exists pg_cron;

do $$
declare
  v_jobid int;
begin
  select jobid
  into v_jobid
  from cron.job
  where command like '%whatsapp-lembretes%'
  limit 1;

  if v_jobid is not null then
    perform cron.unschedule(v_jobid);
  end if;

  perform cron.schedule(
    '*/5 * * * *',
    $cron$
    select net.http_post(
      url:='${supabaseProjectUrl}/functions/v1/whatsapp-lembretes',
      headers:='{ "Content-Type": "application/json", "apikey": "${supabaseAnonKey}", "Authorization": "Bearer
${supabaseAnonKey}", "x-cron-secret": "<CRON_SECRET>" }'::jsonb,
      body:='{}'::jsonb
    );
    $cron$
  );
end;
$$;`
  }, [supabaseAnonKey, supabaseProjectUrl])

  const sqlBillingDailyCron = useMemo(() => {
    return `create extension if not exists pg_net;
create extension if not exists pg_cron;

do $$
declare
  v_jobid int;
begin
  select jobid
  into v_jobid
  from cron.job
  where command like '%whatsapp-lembretes%'
    and command like '%billing_daily%'
  limit 1;
```

```
  if v_jobid is not null then
    perform cron.unschedule(v_jobid);
  end if;

  perform cron.schedule(
    '0 9 * * *',
    $cron$
    select net.http_post(
      url:='${supabaseProjectUrl}/functions/v1/whatsapp-lembretes',
      headers:='{ "Content-Type": "application/json", "apikey": "${supabaseAnonKey}", "Authorization": "Bearer
${supabaseAnonKey}", "x-cron-secret": "<CRON_SECRET>" }'::jsonb,
      body:='{ "action": "billing_daily" }'::jsonb
    );
    $cron$
  );
end;
$$;`
  }, [supabaseAnonKey, supabaseProjectUrl])

  const sqlBillingStatusTrigger = useMemo(() => {
    return `create extension if not exists pg_net;

create or replace function public.usuarios_notify_whatsapp_billing_status()
returns trigger
language plpgsql
security definer
set search_path = public
as $$
begin
  begin
    if TG_OP = 'UPDATE' then
      if NEW.status_pagamento is distinct from OLD.status_pagamento then
        begin
          perform net.http_post(
            url:='${supabaseProjectUrl}/functions/v1/whatsapp-lembretes',
            headers:='{ "Content-Type": "application/json", "apikey": "${supabaseAnonKey}", "Authorization": "Bearer
${supabaseAnonKey}", "x-cron-secret": "<CRON_SECRET>" }'::jsonb,
            body:=jsonb_build_object(
              'action', 'billing_status_changed',
              'usuario_id', NEW.id::text,
              'status_pagamento', NEW.status_pagamento
            )
          );
        exception
          when others then
            null;
        end;
      end if;
    end if;
  exception
    when others then
      null;
  end;

  return NEW;
end;
$$;

drop trigger if exists usuarios_whatsapp_billing_status_trg on public.usuarios;
create trigger usuarios_whatsapp_billing_status_trg
after update of status_pagamento on public.usuarios
for each row execute function public.usuarios_notify_whatsapp_billing_status();`
  }, [supabaseAnonKey, supabaseProjectUrl])

  const sqlWhatsappConfirmacaoTrigger = useMemo(() => {
    return `create extension if not exists pg_net;

create or replace function public.agendamentos_notify_whatsapp_confirmacao()
returns trigger
language plpgsql
security definer
```

```
set search_path = public
as $$
begin
  begin
    if TG_OP = 'INSERT' then
      if NEW.status = 'confirmado' then
        begin
          perform net.http_post(
            url:='${supabaseProjectUrl}/functions/v1/whatsapp-lembretes',
            headers:='{ "Content-Type": "application/json", "apikey": "${supabaseAnonKey}", "Authorization": "Bearer
${supabaseAnonKey}", "x-cron-secret": "<CRON_SECRET>" }'::jsonb,
            body:=jsonb_build_object('agendamento_id', NEW.id::text)
          );
        exception
          when others then
            null;
        end;
      end if;
    elsif TG_OP = 'UPDATE' then
      if NEW.status = 'confirmado' and (OLD.status is distinct from NEW.status) then
        begin
          perform net.http_post(
            url:='${supabaseProjectUrl}/functions/v1/whatsapp-lembretes',
            headers:='{ "Content-Type": "application/json", "apikey": "${supabaseAnonKey}", "Authorization": "Bearer
${supabaseAnonKey}", "x-cron-secret": "<CRON_SECRET>" }'::jsonb,
            body:=jsonb_build_object('agendamento_id', NEW.id::text)
          );
        exception
          when others then
            null;
        end;
      end if;
    end if;
  exception
    when others then
      null;
  end;
  return NEW;
end;
$$;

drop trigger if exists agendamentos_whatsapp_confirmacao_trg on public.agendamentos;
create trigger agendamentos_whatsapp_confirmacao_trg
after insert or update of status on public.agendamentos
for each row execute function public.agendamentos_notify_whatsapp_confirmacao();

create or replace function public.agendamentos_notify_whatsapp_funcionario_novo_agendamento()
returns trigger
language plpgsql
security definer
set search_path = public
as $$
begin
  begin
    if TG_OP = 'INSERT' then
      if NEW.funcionario_id is not null and coalesce(lower(NEW.status), '') <> 'cancelado' then
        begin
          perform net.http_post(
            url:='${supabaseProjectUrl}/functions/v1/whatsapp-lembretes',
            headers:='{ "Content-Type": "application/json", "apikey": "${supabaseAnonKey}", "Authorization": "Bearer
${supabaseAnonKey}", "x-cron-secret": "<CRON_SECRET>" }'::jsonb,
            body:=jsonb_build_object('action', 'funcionario_novo_agendamento', 'agendamento_id', NEW.id::text)
          );
        exception
          when others then
            null;
        end;
      end if;
    elsif TG_OP = 'UPDATE' then
      if NEW.funcionario_id is not null and (OLD.funcionario_id is distinct from NEW.funcionario_id) and
coalesce(lower(NEW.status), '') <> 'cancelado' then
        begin
```

```
          perform net.http_post(
            url:='${supabaseProjectUrl}/functions/v1/whatsapp-lembretes',
            headers:='{ "Content-Type": "application/json", "apikey": "${supabaseAnonKey}", "Authorization": "Bearer
${supabaseAnonKey}", "x-cron-secret": "<CRON_SECRET>" }'::jsonb,
            body:=jsonb_build_object('action', 'funcionario_novo_agendamento', 'agendamento_id', NEW.id::text)
          );
        exception
          when others then
            null;
        end;
      end if;
    end if;
  exception
    when others then
      null;
  end;
  return NEW;
end;
$$;

drop trigger if exists agendamentos_whatsapp_funcionario_novo_agendamento_trg on public.agendamentos;
create trigger agendamentos_whatsapp_funcionario_novo_agendamento_trg
after insert or update of funcionario_id on public.agendamentos
for each row execute function public.agendamentos_notify_whatsapp_funcionario_novo_agendamento();`
  }, [supabaseAnonKey, supabaseProjectUrl])

  const sqlAuditLogs = useMemo(() => {
    return `create extension if not exists pgcrypto;

create table if not exists public.audit_logs (
  id uuid primary key default gen_random_uuid(),
  criado_em timestamptz not null default now(),
  usuario_id uuid null,
  tabela text not null,
  acao text not null,
  registro_id text null,
  ator_email text null
);

create index if not exists audit_logs_criado_em_idx on public.audit_logs (criado_em desc);
create index if not exists audit_logs_usuario_id_idx on public.audit_logs (usuario_id);

alter table public.audit_logs enable row level security;

drop policy if exists "super_admin_select_audit_logs" on public.audit_logs;
create policy "super_admin_select_audit_logs" on public.audit_logs
for select to authenticated
using (public.is_super_admin());

create or replace function public.audit_logs_capture()
returns trigger
language plpgsql
security definer
set search_path = public
as $$
declare
  v_row jsonb := coalesce(to_jsonb(NEW), to_jsonb(OLD));
  v_usuario_id uuid := null;
  v_registro_id text := null;
  v_actor_email text := nullif(auth.jwt() ->> 'email', '');
begin
  begin
    if TG_TABLE_SCHEMA <> 'public' then
      return null;
    end if;

    if TG_TABLE_NAME = 'audit_logs' then
      return null;
    end if;

    v_registro_id := coalesce(v_row ->> 'id', null);
```

```
    if TG_TABLE_NAME = 'usuarios' then
      if (v_row ? 'id') then
        v_usuario_id := nullif(v_row ->> 'id', '')::uuid;
      end if;
    else
      if (v_row ? 'usuario_id') then
        v_usuario_id := nullif(v_row ->> 'usuario_id', '')::uuid;
      elsif (v_row ? 'usuario_master_id') then
        v_usuario_id := nullif(v_row ->> 'usuario_master_id', '')::uuid;
      end if;
    end if;

    insert into public.audit_logs (usuario_id, tabela, acao, registro_id, ator_email)
    values (v_usuario_id, TG_TABLE_NAME, lower(TG_OP), v_registro_id, v_actor_email);
  exception
    when others then
      return null;
  end;

  return null;
end;
$$;

do $$
begin
  if exists (select 1 from information_schema.tables where table_schema = 'public' and table_name = 'usuarios') then
    drop trigger if exists audit_usuarios on public.usuarios;
    create trigger audit_usuarios
    after insert or update or delete on public.usuarios
    for each row execute function public.audit_logs_capture();
  end if;

  if exists (select 1 from information_schema.tables where table_schema = 'public' and table_name = 'funcionarios') then
    drop trigger if exists audit_funcionarios on public.funcionarios;
    create trigger audit_funcionarios
    after insert or update or delete on public.funcionarios
    for each row execute function public.audit_logs_capture();
  end if;

  if exists (select 1 from information_schema.tables where table_schema = 'public' and table_name = 'servicos') then
    drop trigger if exists audit_servicos on public.servicos;
    create trigger audit_servicos
    after insert or update or delete on public.servicos
    for each row execute function public.audit_logs_capture();
  end if;

  if exists (select 1 from information_schema.tables where table_schema = 'public' and table_name = 'agendamentos') then
    drop trigger if exists audit_agendamentos on public.agendamentos;
    create trigger audit_agendamentos
    after insert or update or delete on public.agendamentos
    for each row execute function public.audit_logs_capture();
  end if;

  if exists (select 1 from information_schema.tables where table_schema = 'public' and table_name = 'bloqueios') then
    drop trigger if exists audit_bloqueios on public.bloqueios;
    create trigger audit_bloqueios
    after insert or update or delete on public.bloqueios
    for each row execute function public.audit_logs_capture();
  end if;
end;
$$;`
  }, [])

  return (
    <AdminShell>
      <div className="space-y-6">
        <Card>
          <div className="p-6 space-y-2">
            <div className="text-sm font-semibold text-slate-900">Sessão</div>
            <div className="text-sm text-slate-700">{email ?? '—'}</div>
            <div className="text-xs text-slate-600 font-mono">{userId ?? '—'}</div>
          </div>
```

```
        </Card>

        <Card>
          <div className="p-6 space-y-3">
            <div className="text-sm font-semibold text-slate-900">Email (Resend) — DNS</div>
            <div className="text-sm text-slate-600">
              Esse checker não altera DNS. Ele só confirma se os registros exigidos pelo Resend já estão públicos.
            </div>

            <div className="flex flex-col gap-3 sm:flex-row sm:items-end">
              <div className="flex-1">
                <div className="text-xs text-slate-600 mb-1">Domínio</div>
                <Input value={resendDomain} onChange={(e) => setResendDomain(e.target.value)}
placeholder="smagenda.com.br" />
              </div>

              <div className="flex gap-2">
                <Button onClick={() => runResend('status')} disabled={resendLoading}>
                  Checar
                </Button>
                <Button variant="secondary" onClick={() => runResend('verify')} disabled={resendLoading}>
                  Revalidar no Resend
                </Button>
              </div>
            </div>

            {resendError ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">
{resendError}</div> : null}

            {resendResult?.domain ? (
              <div className="space-y-3">
                <div className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-sm text-slate-800">
                  <div className="flex flex-wrap gap-x-4 gap-y-1">
                    <div>
                      <span className="text-slate-600">Domínio:</span> <span className="font-mono">
{resendResult.domain.name ?? '—'}</span>
                    </div>
                    <div>
                      <span className="text-slate-600">Status:</span> <span className="font-mono">
{resendResult.domain.status ?? '—'}</span>
                    </div>
                    <div>
                      <span className="text-slate-600">Região:</span> <span className="font-mono">
{resendResult.domain.region ?? '—'}</span>
                    </div>
                  </div>
                  {resendResult.dns?.nameservers && resendResult.dns.nameservers.length > 0 ? (
                    <div className="mt-2 text-xs text-slate-600">
                      NS: <span className="font-mono">{resendResult.dns.nameservers.join(' • ')}</span>
                    </div>
                  ) : null}
                </div>

                {resendResult.dns?.records && resendResult.dns.records.length > 0 ? (
                  <div className="space-y-2">
                    {resendResult.dns.records.map((r, idx) => {
                      const match = r.dns?.match === true
                      const found = r.dns?.found === true
                      const cls = match
                        ? 'border-emerald-200 bg-emerald-50 text-emerald-800'
                        : found
                          ? 'border-amber-200 bg-amber-50 text-amber-900'
                          : 'border-rose-200 bg-rose-50 text-rose-700'
                      return (
                        <div key={`${r.fqdn ?? idx}`} className={`rounded-xl border p-3 ${cls}`}>
                          <div className="flex flex-wrap gap-x-3 gap-y-1 text-xs">
                            <span className="font-mono">
                              {r.type ?? '—'} {r.fqdn ?? r.name ?? '—'}
                            </span>
                            <span className="text-slate-600">Resend:</span>
                            <span className="font-mono">{r.resendStatus ?? '—'}</span>
                            <span className="text-slate-600">DNS:</span>
```

```
                            <span className="font-mono">{match ? 'ok' : found ? 'encontrado (difere)' : 'não
encontrado'}</span>
                          </div>

                          <div className="mt-2 text-xs">
                            <div>
                              <span className="text-slate-600">Esperado:</span>{' '}
                              <span className="font-mono break-all">
                                {r.expected?.value ?? '—'}
                                {typeof r.expected?.priority === 'number' ? ` (prio ${r.expected.priority})` : ''}
                              </span>
                            </div>

                            {r.dns?.values && r.dns.values.length > 0 ? (
                              <div className="mt-1">
                                <span className="text-slate-600">Retorno:</span>{' '}
                                <span className="font-mono break-all">{r.dns.values.join(' | ')}</span>
                              </div>
                            ) : null}

                            {r.hint ? <div className="mt-1 text-xs">{r.hint}</div> : null}
                          </div>
                        </div>
                      )
                    })}
                  </div>
                ) : null}
              </div>
            ) : null}
          </div>
        </Card>

        <Card>
          <div className="p-6 space-y-3">
            <div className="text-sm font-semibold text-slate-900">Email (Resend) — Teste de envio</div>
            <div className="text-sm text-slate-600">
              Envia um email real via API do Resend. Se falhar, normalmente é DNS/domínio não verificado ou falta da
secret `RESEND_API_KEY`.
            </div>

            <div className="flex flex-col gap-3">
              <Input label="Remetente (From)" value={resendTestFrom} onChange={(e) => setResendTestFrom(e.target.value)}
placeholder="SMagenda <no-reply@smagenda.com.br>" />
              <Input
                label="Destinatário (To)"
                value={resendTestTo}
                onChange={(e) => setResendTestTo(e.target.value)}
                placeholder="seuemail@dominio.com"
              />
              <Input
                label="Assunto"
                value={resendTestSubject}
                onChange={(e) => setResendTestSubject(e.target.value)}
                placeholder="Teste de email"
              />
              <div className="flex justify-end">
                <Button onClick={sendResendTest} disabled={resendTestSending || resendLoading}>
                  {resendTestSending ? 'Enviando…' : 'Enviar email teste'}
                </Button>
              </div>
            </div>

            {resendTestError ? (
              <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{resendTestError}
</div>
            ) : null}
            {resendTestResult ? (
              <pre className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-xs text-slate-800 overflow-auto
whitespace-pre-wrap">
                {JSON.stringify(resendTestResult, null, 2)}
              </pre>
            ) : null}
```

```
            <div className="rounded-xl border border-slate-200 bg-white p-3 text-sm text-slate-700">
              <div className="font-semibold text-slate-900">SMTP (Supabase Auth)</div>
              <div className="text-slate-600">
                Para os emails de confirmação e recuperação de senha do Supabase, configure em Authentication → SMTP
Settings.
              </div>
              <div className="mt-2 text-xs text-slate-600 font-mono">
                Host=smtp.resend.com • Username=resend • Password=RESEND_API_KEY • Port=465 (SMTPS) ou 587 (STARTTLS)
              </div>
            </div>
          </div>
        </Card>

        <SqlCard
          title="SQL de políticas (Super Admin)"
          description="Use se as telas do admin retornarem erro de RLS/permissão."
          sql={sql}
        />

        <SqlCard
          title="SQL de Trial (7 dias / 1 funcionário)"
          description="Use para habilitar trial automático após confirmação de email."
          sql={sqlTrial}
        />

        <SqlCard
          title="SQL de Pagamento (Stripe)"
          description="Use para permitir que o webhook salve o status de pagamento real."
          sql={sqlPaymentsStripe}
        />

        <SqlCard
          title="SQL de políticas (Usuário / Funcionário)"
          description="Use para liberar o login e o acesso do funcionário às telas."
          sql={sqlRlsApp}
        />

        <SqlCard
          title="SQL de permissões de funcionário (Atendente)"
          description="Use se aparecer erro ao criar funcionário com permissao='atendente'."
          sql={sqlFuncionariosPermissaoAtendente}
        />

        <SqlCard
          title="SQL de bloqueio de horário do funcionário"
          description="Use para impedir que funcionário salve o próprio horário via RPC (apenas o gerente ajusta no
painel)."
          sql={sqlFuncionarioHorarios}
        />

        <SqlCard
          title="SQL de horários públicos (ocupações + bloqueios)"
          description="Use para o link público respeitar agendamentos e bloqueios."
          sql={sqlPublicOcupacoes}
        />

        <SqlCard
          title="SQL do link público (listar + agendar)"
          description="Use para o link público funcionar com RLS habilitado, sem expor dados sensíveis."
          sql={sqlPublicBooking}
          defaultOpen
        />

        <SqlCard
          title="SQL da taxa de agendamento (créditos)"
          description="Use para habilitar taxa por serviço, cobrança no agendamento público e crédito por falta."
          sql={sqlTaxaAgendamento}
        />

        <SqlCard
          title="SQL de Multi-unidades (EMPRESA)"
```

```
          description="Use para criar filiais e habilitar links do tipo /agendar/:slug/:unidade."
          sql={sqlMultiUnidadesEnterprise}
        />

        <SqlCard title="SQL do Storage (logos)" description="Use para upload de logo no onboarding e exibição no link
público." sql={sqlStorageLogos} />

        <SqlCard
          title="SQL do Storage (fotos de serviços)"
          description="Use para upload de fotos dos serviços (bucket público + políticas por usuário)."
          sql={sqlStorageServicosFotos}
        />

        <SqlCard title="SQL de fotos nos serviços (PRO+)" description="Use para liberar fotos de serviços apenas no
plano PRO+." sql={sqlServicosFotosPro} />

        <SqlCard
          title="SQL de limite de serviços (BASIC)"
          description="Use para limitar a criação de serviços a 3 no plano BASIC/FREE."
          sql={sqlServicosLimiteBasic}
        />

        <SqlCard title="SQL do WhatsApp (automação)" description="Use para habilitar QR Code, confirmação automática e
lembretes." sql={sqlWhatsappAutomacao} />

        <SqlCard
          title="SQL do WhatsApp (habilitação por cliente)"
          description="Use para liberar o WhatsApp por cliente via painel do Super Admin."
          sql={sqlWhatsappHabilitacao}
        />

        <SqlCard
          title="SQL do Tema Prospector (habilitação por cliente)"
          description="Use para liberar o tema alternativo por cliente via painel do Super Admin."
          sql={sqlTemaProspectorHabilitacao}
        />

        <SqlCard
          title="SQL do WhatsApp (Super Admin)"
          description="Use para salvar a Evolution API e enviar avisos na tela do Super Admin."
          sql={sqlWhatsappSuperAdmin}
        />

        <SqlCard
          title="SQL do WhatsApp (migração p/ config global)"
          description="Use para aplicar o modelo novo (URL/key no Super Admin) e limpar URL/key dos usuários."
          sql={sqlWhatsappGlobalConfig}
        />

        <SqlCard
          title="SQL do WhatsApp (cron de lembretes)"
          description="Use para executar lembretes automaticamente a cada 5 minutos."
          sql={sqlWhatsappLembretesCron}
        />

        <SqlCard
          title="SQL de Cobrança (cron diário)"
          description="Use para enviar avisos e suspender/cancelar automaticamente por atraso."
          sql={sqlBillingDailyCron}
        />

        <SqlCard
          title="SQL de Cobrança (trigger de status)"
          description="Use para avisar no WhatsApp quando o status do pagamento mudar."
          sql={sqlBillingStatusTrigger}
        />

        <SqlCard
          title="SQL do WhatsApp (trigger confirmação imediata)"
          description="Use para enviar confirmação automaticamente ao criar/confirmar um agendamento."
          sql={sqlWhatsappConfirmacaoTrigger}
        />
```

```
        <SqlCard title="SQL de Logs de Auditoria" description="Use para registrar ações nas tabelas e exibir em
/admin/logs." sql={sqlAuditLogs} />
      </div>
    </AdminShell>
  )
}
```

smagenda/src/views/admin/AdminDashboardPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { AdminShell } from '../../components/layout/AdminShell'
import { Card } from '../../components/ui/Card'
import { supabase } from '../../lib/supabase'

type UsuarioRow = { id: string; ativo: boolean; plano: string; status_pagamento: string }

export function AdminDashboardPage() {
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [usuarios, setUsuarios] = useState<UsuarioRow[]>([])

  const resumo = useMemo(() => {
    const total = usuarios.length
    const ativos = usuarios.filter((u) => u.ativo).length
    const inadimplente = usuarios.filter((u) => u.status_pagamento === 'inadimplente').length
    return { total, ativos, inadimplente }
  }, [usuarios])

  useEffect(() => {
    const run = async () => {
      setLoading(true)
      setError(null)
      const { data, error: err } = await
supabase.from('usuarios').select('id,ativo,plano,status_pagamento').order('criado_em', { ascending: false }).limit(500)
      if (err) {
        setError(err.message)
        setLoading(false)
        return
      }
      setUsuarios((data ?? []) as unknown as UsuarioRow[])
      setLoading(false)
    }
    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar')
      setLoading(false)
    })
  }, [])

  return (
    <AdminShell>
      <div className="space-y-6">
        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> :
null}

        {!loading && !error && usuarios.length === 0 ? (
          <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-800">
            Nenhum cliente retornado do banco.
            <div className="mt-1 text-amber-900">
              Se você já tem clientes no Supabase, isso geralmente indica políticas (RLS) faltando na tabela
              <span className="font-mono text-xs"> public.usuarios</span>.
            </div>
            <div className="mt-2">
              Abra <span className="font-mono text-xs">/admin/configuracoes</span> e execute o bloco “SQL de políticas
(Super Admin)” no SQL Editor do Supabase.
            </div>
          </div>
        ) : null}
        <div className="grid grid-cols-1 gap-4 sm:grid-cols-3">
          <Card>
            <div className="p-6">
              <div className="text-sm text-slate-600">Total de clientes</div>
              <div className="text-2xl font-semibold text-slate-900">{loading ? '—' : resumo.total}</div>
            </div>
          </Card>
          <Card>
            <div className="p-6">
              <div className="text-sm text-slate-600">Ativos</div>
              <div className="text-2xl font-semibold text-slate-900">{loading ? '—' : resumo.ativos}</div>
            </div>
```

```
          </Card>
          <Card>
            <div className="p-6">
              <div className="text-sm text-slate-600">Inadimplentes</div>
              <div className="text-2xl font-semibold text-slate-900">{loading ? '—' : resumo.inadimplente}</div>
            </div>
          </Card>
        </div>
      </div>
    </AdminShell>
  )
}
```

smagenda/src/views/admin/AdminLoginPage.tsx

```
import { useMemo, useState } from 'react'
import { Link, useNavigate } from 'react-router-dom'
import { Button } from '../../components/ui/Button'
import { Input } from '../../components/ui/Input'
import { supabase } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

export function AdminLoginPage() {
  const navigate = useNavigate()
  const { refresh } = useAuth()

  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')
  const [submitting, setSubmitting] = useState(false)
  const [resending, setResending] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [info, setInfo] = useState<string | null>(null)

  const canSubmit = useMemo(() => email.trim() && password.trim(), [email, password])

  const canResend = useMemo(() => email.trim().length > 0 && error?.toLowerCase().includes('confirm') === true, [email, error])

  const resend = async () => {
    if (!email.trim()) return
    setResending(true)
    setInfo(null)
    try {
      const { error: resendErr } = await supabase.auth.resend({ type: 'signup', email: email.trim() })
      if (resendErr) {
        setError(resendErr.message)
        return
      }
      setInfo('Email de confirmação reenviado. Verifique sua caixa de entrada e spam.')
    } finally {
      setResending(false)
    }
  }

  const submit = async () => {
    const inputEmail = email.trim().toLowerCase()
    setSubmitting(true)
    setError(null)
    setInfo(null)
    const { error: signInError } = await supabase.auth.signInWithPassword({ email: inputEmail, password })
    if (signInError) {
      const msg = signInError.message
      const notConfirmed = msg.toLowerCase().includes('confirm')
      setError(notConfirmed ? 'Email não confirmado. Reenvie o email de confirmação ou confirme no Supabase.' : msg)
      setSubmitting(false)
      return
    }
    await refresh()

    const { data: sessionData } = await supabase.auth.getSession()
    const userId = sessionData.session?.user?.id
    if (!userId) {
      setError('Falha ao iniciar sessão')
      setSubmitting(false)
      return
    }

    const { data: adminRow, error: adminErr } = await supabase.from('super_admin').select('id').eq('id', userId).maybeSingle()
    if (adminErr) {
      const msg = adminErr.message
      const missingTable = msg.includes("Could not find the table 'public.super_admin'") || msg.includes('schema cache')
      setError(missingTable ? 'Tabela super_admin não configurada no Supabase. Acesse /admin/bootstrap para ver o SQL.' : msg)
      await supabase.auth.signOut()
```

```
      setSubmitting(false)
      return
    }

    if (!adminRow) {
      setError('Conta não autorizada para Super Admin. Use o login normal em /login.')
      await supabase.auth.signOut()
      setSubmitting(false)
      return
    }
    setSubmitting(false)
    navigate('/admin/dashboard')
  }

  return (
    <div className="min-h-screen bg-slate-50 flex items-center justify-center p-4">
      <div className="w-full max-w-md space-y-6">
        <div>
          <div className="text-xs font-semibold tracking-wide text-slate-500">SMagenda</div>
          <div className="text-2xl font-semibold text-slate-900">Login Super Admin</div>
        </div>
        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}
        {info ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">{info}</div> : null}
        <div className="space-y-4 rounded-xl border border-slate-200 bg-white p-6 shadow-sm">
          <Input label="Email" value={email} onChange={(e) => setEmail(e.target.value)} />
          <Input label="Senha" type="password" value={password} onChange={(e) => setPassword(e.target.value)} />
          <Button fullWidth onClick={submit} disabled={!canSubmit || submitting}>
            Entrar
          </Button>
          <div className="text-center text-sm text-slate-600">
            <Link to="/esqueci-senha" className="font-medium text-slate-900 hover:underline">
              Esqueci minha senha
            </Link>
          </div>
          {canResend ? (
            <Button fullWidth variant="secondary" onClick={resend} disabled={resending}>
              Reenviar email de confirmação
            </Button>
          ) : null}
        </div>
      </div>
    </div>
  )
}
```

smagenda/src/views/admin/AdminLogsPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { AdminShell } from '../../components/layout/AdminShell'
import { Card } from '../../components/ui/Card'
import { supabase } from '../../lib/supabase'

type AuditLogRow = {
  id: string
  criado_em: string
  usuario_id: string | null
  tabela: string
  acao: string
  registro_id: string | null
  ator_email: string | null
}

type UsuarioMini = { id: string; nome_negocio: string; slug: string }

export function AdminLogsPage() {
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [logs, setLogs] = useState<AuditLogRow[]>([])
  const [usuariosById, setUsuariosById] = useState<Record<string, UsuarioMini>>({})

  useEffect(() => {
    const run = async () => {
      setLoading(true)
      setError(null)

      const { data, error: err } = await supabase
        .from('audit_logs')
        .select('id,criado_em,usuario_id,tabela,acao,registro_id,ator_email')
        .order('criado_em', { ascending: false })
        .limit(200)
      if (err) {
        const msg = err.message
        const missingTable = msg.includes("Could not find the table 'public.audit_logs'") || msg.includes('schema
cache')
        setError(missingTable ? 'Tabela audit_logs não configurada. Acesse /admin/configuracoes e execute o bloco “SQL
de Logs de Auditoria” no Supabase.' : msg)
        setLoading(false)
        return
      }
      const rows = (data ?? []) as unknown as AuditLogRow[]
      setLogs(rows)

      const ids = Array.from(new Set(rows.map((r) => r.usuario_id))).filter(Boolean)
      if (ids.length === 0) {
        setUsuariosById({})
        setLoading(false)
        return
      }

      const { data: usuariosData, error: usuariosErr } = await
supabase.from('usuarios').select('id,nome_negocio,slug').in('id', ids)
      if (usuariosErr) {
        setError(usuariosErr.message)
        setLoading(false)
        return
      }

      const nextMap: Record<string, UsuarioMini> = {}
      for (const u of (usuariosData ?? []) as unknown as UsuarioMini[]) {
        nextMap[u.id] = u
      }
      setUsuariosById(nextMap)
      setLoading(false)
    }
    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar')
      setLoading(false)
```

```
    })
  }, [])

  const items = useMemo(() => {
    return logs.map((l) => ({
      ...l,
      usuario: l.usuario_id ? usuariosById[l.usuario_id] ?? null : null,
    }))
  }, [logs, usuariosById])

  return (
    <AdminShell>
      <div className="space-y-6">
        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}
        <Card>
          <div className="p-6">
            <div className="text-sm font-semibold text-slate-900">Atividade recente</div>
            <div className="text-sm text-slate-600">Logs de auditoria (todos os clientes)</div>
          </div>
          <div className="divide-y divide-slate-100">
            {loading ? (
              <div className="p-6 text-sm text-slate-600">Carregando…</div>
            ) : items.length === 0 ? (
              <div className="p-6 text-sm text-slate-600">Sem dados.</div>
            ) : (
              items.map((l) => (
                <div key={l.id} className="p-4">
                  <div className="flex flex-wrap items-center justify-between gap-2">
                    <div>
                      <div className="text-sm font-semibold text-slate-900">
                        {l.usuario ? l.usuario.nome_negocio : l.usuario_id ?? '—'} • {l.acao.toUpperCase()} {l.tabela}
                      </div>
                      <div className="text-sm text-slate-600">
                        {l.criado_em}
                        {l.usuario ? ` • /${l.usuario.slug}` : ''}
                        {l.ator_email ? ` • ${l.ator_email}` : ''}
                        {l.registro_id ? ` • ${l.registro_id}` : ''}
                      </div>
                    </div>
                    <div className="text-sm font-medium text-slate-700">{l.tabela}</div>
                  </div>
                </div>
              ))
            )}
          </div>
        </Card>
      </div>
    </AdminShell>
  )
}
```

smagenda/src/views/admin/AdminWhatsappAvisosPage.tsx

```
import { useEffect, useMemo, useRef, useState } from 'react'
import { AdminShell } from '../../components/layout/AdminShell'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type ClienteRow = {
  id: string
  nome_negocio: string
  slug: string
  telefone: string | null
  plano: string
  status_pagamento: string
  ativo: boolean
  whatsapp_habilitado?: boolean | null
  tema_prospector_habilitado?: boolean | null
}

function normalizePlanoLabel(planoRaw: string) {
  const p = String(planoRaw ?? '').trim().toLowerCase()
  if (p === 'enterprise') return 'EMPRESA'
  if (p === 'pro' || p === 'team') return 'PRO'
  if (p === 'basic') return 'BASIC'
  if (p === 'free') return 'FREE'
  return planoRaw
}

type SuperAdminConfigRow = {
  id: string
  whatsapp_api_url: string | null
  whatsapp_api_key=[REDACTED]
  whatsapp_instance_name: string | null
}

function decodeJwtPayload(jwt: string) {
  const parts = jwt.split('.')
  if (parts.length !== 3) return null
  const payload = parts[1]
  if (!payload) return null

  const normalized = payload.replace(/-/g, '+').replace(/_/g, '/')
  const pad = normalized.length % 4
  const padded = normalized + (pad ? '='.repeat(4 - pad) : '')

  try {
    const json = JSON.parse(atob(padded)) as Record<string, unknown>
    return {
      iss: typeof json.iss === 'string' ? json.iss : null,
      aud: typeof json.aud === 'string' ? json.aud : null,
      exp: typeof json.exp === 'number' ? json.exp : null,
      role: typeof json.role === 'string' ? json.role : null,
      sub: typeof json.sub === 'string' ? json.sub : null,
    }
  } catch {
    return null
  }
}

async function callAdminWhatsapp(body: unknown) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env', missing: supabaseEnv.missing },
fnVersion: null as string | null, elapsedMs: 0 }
  }

  const supabaseUrl = supabaseEnv.values.VITE_SUPABASE_URL
  const supabaseAnonKey = supabaseEnv.values.VITE_SUPABASE_ANON_KEY
```

```
  const tryRefresh = async () => {
    const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
    if (refreshErr) return null
    return refreshed.session ?? null
  }

  const getSession = async () => {
    const { data: sessionData } = await supabase.auth.getSession()
    const sess = sessionData.session
    if (!sess) return null
    const expiresAt = sess.expires_at ?? null
    const now = Math.floor(Date.now() / 1000)
    if (expiresAt && expiresAt <= now + 30) {
      const refreshed = await tryRefresh()
      return refreshed ?? sess
    }
    return sess
  }

  const sess = await getSession()
  if (!sess?.access_token) {
    return { ok: false as const, status: 401, body: { error: 'unauthorized', message: 'missing_session' }, fnVersion:
null as string | null, elapsedMs: 0 }
  }

  const fnUrl = `${supabaseUrl.replace(/\/+$/, '')}/functions/v1/whatsapp`

  const post = async (token: string) => {
    const startedAt = performance.now()
    const controller = new AbortController()
    const timeoutId = setTimeout(() => controller.abort(), 45000)
    try {
      const res = await fetch(fnUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          apikey=[REDACTED]
          Authorization: `Bearer ${token}`,
          'x-user-jwt': token,
        },
        body: JSON.stringify(body),
        signal: controller.signal,
      })

      const fnVersion = res.headers.get('x-smagenda-fn')

      let text = ''
      try {
        text = await res.text()
      } catch (e: unknown) {
        const msg = e instanceof Error ? e.message : 'Falha de rede'
        return { ok: false as const, status: 0, body: { error: 'network_error', message: msg }, fnVersion, elapsedMs:
Math.round(performance.now() - startedAt) }
      }

      let parsed: unknown = null
      try {
        parsed = text ? JSON.parse(text) : null
      } catch {
        parsed = text
      }

      const elapsedMs = Math.round(performance.now() - startedAt)
      if (!res.ok) return { ok: false as const, status: res.status, body: parsed, fnVersion, elapsedMs }
      return { ok: true as const, status: res.status, body: parsed, fnVersion, elapsedMs }
    } catch (e: unknown) {
      const msg = e instanceof Error ? e.message : 'Falha de rede'
      return { ok: false as const, status: 0, body: { error: 'network_error', message: msg }, fnVersion: null,
elapsedMs: Math.round(performance.now() - startedAt) }
    } finally {
      clearTimeout(timeoutId)
    }
```

```
  }

  const first = await post(sess.access_token)
  const fnVersion = first.fnVersion
  const parsed = first.body
  const resStatus = first.status
  const resOk = first.ok

  if (!resOk && resStatus === 401) {
    if (!fnVersion) {
      if (
        parsed &&
        typeof parsed === 'object' &&
        (parsed as Record<string, unknown>).message === 'Invalid JWT' &&
        (parsed as Record<string, unknown>).code === 401
      ) {
        return { ok: false as const, status: 401, body: { error: 'supabase_gateway_invalid_jwt' } }
      }
    }
    const refreshed = await tryRefresh()
    if (refreshed?.access_token) {
      const second = await post(refreshed.access_token)
      if (!second.ok) {
        const withFn = second.fnVersion
          ? typeof second.body === 'object' && second.body !== null
            ? { ...(second.body as Record<string, unknown>), fn: second.fnVersion }
            : { error: 'fn_error', fn: second.fnVersion, details: second.body }
          : second.body
        return { ok: false as const, status: second.status, body: withFn, fnVersion: second.fnVersion, elapsedMs: 
second.elapsedMs }
      }
      return { ok: true as const, status: second.status, body: second.body, fnVersion: second.fnVersion, elapsedMs: 
second.elapsedMs }
    }
  }

  if (!resOk) return { ok: false as const, status: resStatus, body: parsed, fnVersion, elapsedMs: first.elapsedMs }
  return { ok: true as const, status: resStatus, body: parsed, fnVersion, elapsedMs: first.elapsedMs }
}

export function AdminWhatsappAvisosPage() {
  const { principal } = useAuth()
  const superAdminId = principal?.kind === 'super_admin' ? principal.profile.id : null

  const [configLoading, setConfigLoading] = useState(true)
  const [schemaIncompleto, setSchemaIncompleto] = useState(false)
  const [apiUrl, setApiUrl] = useState('')
  const [apiKey, setApiKey] = useState('')
  const [instanceName, setInstanceName] = useState('')
  const [savingConfig, setSavingConfig] = useState(false)
  const [configSaved, setConfigSaved] = useState(false)

  const [instanceState, setInstanceState] = useState<string | null>(null)
  const [qrBase64, setQrBase64] = useState<string | null>(null)
  const [pairingCode, setPairingCode] = useState<string | null>(null)
  const [checkingStatus, setCheckingStatus] = useState(false)
  const [connecting, setConnecting] = useState(false)
  const [disconnecting, setDisconnecting] = useState(false)

  const [clientesLoading, setClientesLoading] = useState(true)
  const [clientes, setClientes] = useState<ClienteRow[]>([])
  const [query, setQuery] = useState('')
  const [selected, setSelected] = useState<Record<string, boolean>>({})
  const [mensagem, setMensagem] = useState('')
  const [sending, setSending] = useState(false)
  const [lastSendSummary, setLastSendSummary] = useState<string | null>(null)

  const [diagRunning, setDiagRunning] = useState(false)
  const [diagText, setDiagText] = useState<string | null>(null)

  const [updatingClientesWhatsapp, setUpdatingClientesWhatsapp] = useState(false)
  const [schemaClientesWhatsappIncompleto, setSchemaClientesWhatsappIncompleto] = useState(false)
```

```
  const [updatingClientesTemaProspector, setUpdatingClientesTemaProspector] = useState(false)
  const [schemaClientesTemaProspectorIncompleto, setSchemaClientesTemaProspectorIncompleto] = useState(false)

  const [error, setError] = useState<string | null>(null)
  const aliveRef = useRef(true)

  useEffect(() => {
    aliveRef.current = true
    return () => {
      aliveRef.current = false
    }
  }, [])

  useEffect(() => {
    const isMissingColumnError = (message: string) => message.toLowerCase().includes('does not exist') &&
message.toLowerCase().includes('column')

    const loadConfig = async () => {
      if (!superAdminId) return
      setConfigLoading(true)
      setError(null)
      setLastSendSummary(null)
      setSchemaIncompleto(false)
      const { data, error: err } = await supabase
        .from('super_admin')
        .select('id,whatsapp_api_url,whatsapp_api_key,whatsapp_instance_name')
        .eq('id', superAdminId)
        .maybeSingle()
      if (err) {
        if (isMissingColumnError(err.message)) {
          setSchemaIncompleto(true)
        } else {
          setError(err.message)
        }
        setConfigLoading(false)
        return
      }

      const row = (data ?? null) as unknown as SuperAdminConfigRow | null
      setApiUrl(row?.whatsapp_api_url ?? '')
      setApiKey(row?.whatsapp_api_key ?? '')
      setInstanceName(row?.whatsapp_instance_name ?? '')
      setConfigLoading(false)
    }

    const loadClientes = async () => {
      setClientesLoading(true)
      setSchemaClientesWhatsappIncompleto(false)
      setSchemaClientesTemaProspectorIncompleto(false)

      const first = await supabase
        .from('usuarios')

.select('id,nome_negocio,slug,telefone,plano,status_pagamento,ativo,whatsapp_habilitado,tema_prospector_habilitado')
        .order('criado_em', { ascending: false })
        .limit(500)
      if (first.error) {
        if (isMissingColumnError(first.error.message)) {
          const second = await supabase
            .from('usuarios')
            .select('id,nome_negocio,slug,telefone,plano,status_pagamento,ativo,whatsapp_habilitado')
            .order('criado_em', { ascending: false })
            .limit(500)
          if (second.error) {
            if (isMissingColumnError(second.error.message)) {
              setSchemaClientesWhatsappIncompleto(true)
              setSchemaClientesTemaProspectorIncompleto(true)
              const third = await supabase
                .from('usuarios')
                .select('id,nome_negocio,slug,telefone,plano,status_pagamento,ativo')
                .order('criado_em', { ascending: false })
```

```
                .limit(500)
              if (third.error) {
                const msg = third.error.message
                const rls = msg.toLowerCase().includes('row-level security') || msg.toLowerCase().includes('permission denied')
                setError(rls ? 'Sem permissão para listar clientes. Em /admin/configuracoes, execute “SQL de políticas (Super Admin)” e “SQL do WhatsApp (Super Admin)” no Supabase.' : msg)
                setClientes([])
                setClientesLoading(false)
                return
              }
              setClientes((third.data ?? []) as unknown as ClienteRow[])
              setClientesLoading(false)
              return
            }
            const msg = second.error.message
            const rls = msg.toLowerCase().includes('row-level security') || msg.toLowerCase().includes('permission denied')
            setError(rls ? 'Sem permissão para listar clientes. Em /admin/configuracoes, execute “SQL de políticas (Super Admin)” e “SQL do WhatsApp (Super Admin)” no Supabase.' : msg)
            setClientes([])
            setClientesLoading(false)
            return
          }

          setSchemaClientesTemaProspectorIncompleto(true)
          setClientes((second.data ?? []) as unknown as ClienteRow[])
          setClientesLoading(false)
          return
        }

        const msg = first.error.message
        const rls = msg.toLowerCase().includes('row-level security') || msg.toLowerCase().includes('permission denied')
        setError(rls ? 'Sem permissão para listar clientes. Em /admin/configuracoes, execute “SQL de políticas (Super Admin)” e “SQL do WhatsApp (Super Admin)” no Supabase.' : msg)
        setClientes([])
        setClientesLoading(false)
        return
      }

      setClientes((first.data ?? []) as unknown as ClienteRow[])
      setClientesLoading(false)
    }

    loadConfig()
      .then(() => loadClientes())
      .catch((e: unknown) => {
        setError(e instanceof Error ? e.message : 'Erro ao carregar')
      })
  }, [superAdminId])

  const conectado = useMemo(() => Boolean(apiUrl.trim() && apiKey.trim()), [apiUrl, apiKey])
  const filteredClientes = useMemo(() => {
    const q = query.trim().toLowerCase()
    if (!q) return clientes
    return clientes.filter((c) => (c.nome_negocio ?? '').toLowerCase().includes(q) || (c.slug ?? '').toLowerCase().includes(q) || (c.telefone ?? '').toLowerCase().includes(q))
  }, [clientes, query])

  const selectedIds = useMemo(() => Object.keys(selected).filter((id) => selected[id] === true), [selected])
  const selectedCount = selectedIds.length

  const enableWhatsappSelected = async (enabled: boolean) => {
    if (selectedCount === 0) return
    if (enabled && (!apiUrl.trim() || !apiKey.trim())) {
      setError('Preencha e salve a Evolution API acima antes de habilitar para clientes.')
      return
    }
    setUpdatingClientesWhatsapp(true)
    setError(null)
    setLastSendSummary(null)
    const patch: Record<string, unknown> = enabled
```

```
      ? {
          whatsapp_habilitado: true,
        }
      : {
          whatsapp_habilitado: false,
        }

    const { error: err } = await supabase.from('usuarios').update(patch).in('id', selectedIds)
    if (err) {
      const msg = err.message
      if (msg.toLowerCase().includes('does not exist') && msg.toLowerCase().includes('column')) {
        setError('Seu Supabase não tem a coluna de habilitação do WhatsApp. Em /admin/configuracoes, execute o bloco
“SQL do WhatsApp (habilitação por cliente)”.')
      } else {
        setError(msg)
      }
      setUpdatingClientesWhatsapp(false)
      return
    }

    setClientes((prev) =>
      prev.map((c) => {
        if (!selectedIds.includes(c.id)) return c
        return { ...c, whatsapp_habilitado: enabled }
      })
    )
    setUpdatingClientesWhatsapp(false)
  }

  const enableTemaProspectorSelected = async (enabled: boolean) => {
    if (selectedCount === 0) return
    setUpdatingClientesTemaProspector(true)
    setError(null)
    setLastSendSummary(null)

    const patch: Record<string, unknown> = enabled
      ? {
          tema_prospector_habilitado: true,
        }
      : {
          tema_prospector_habilitado: false,
        }

    const { error: err } = await supabase.from('usuarios').update(patch).in('id', selectedIds)
    if (err) {
      const msg = err.message
      if (msg.toLowerCase().includes('does not exist') && msg.toLowerCase().includes('column')) {
        setError('Seu Supabase não tem a coluna de habilitação do Tema Prospector. Em /admin/configuracoes, execute o
bloco “SQL do Tema Prospector (habilitação por cliente)”.')
      } else {
        setError(msg)
      }
      setUpdatingClientesTemaProspector(false)
      return
    }

    setClientes((prev) =>
      prev.map((c) => {
        if (!selectedIds.includes(c.id)) return c
        return { ...c, tema_prospector_habilitado: enabled }
      })
    )
    setUpdatingClientesTemaProspector(false)
  }

  const saveConfig = async () => {
    if (!superAdminId) return
    if (schemaIncompleto) return
    setSavingConfig(true)
    setError(null)
    setConfigSaved(false)
    const payload = {
```

```
      whatsapp_api_url: apiUrl.trim() || null,
      whatsapp_api_key=[REDACTED]
      whatsapp_instance_name: instanceName.trim() || null,
    }
    const { data, error: err } = await supabase
      .from('super_admin')
      .upsert({ id: superAdminId, ...payload }, { onConflict: 'id' })
      .select('id,whatsapp_api_url,whatsapp_api_key,whatsapp_instance_name')
      .maybeSingle()
    if (err) {
      setError(err.message)
      setSavingConfig(false)
      return
    }
    if (!data) {
      setError('Não foi possível salvar a configuração do Super Admin. Em /admin/configuracoes, execute “SQL do WhatsApp (Super Admin)” e “SQL do Bootstrap (Super Admin)”.')
      setSavingConfig(false)
      return
    }
    const row = data as unknown as SuperAdminConfigRow
    setApiUrl(row.whatsapp_api_url ?? '')
    setApiKey(row.whatsapp_api_key ?? '')
    setInstanceName(row.whatsapp_instance_name ?? '')
    setConfigSaved(true)
    setSavingConfig(false)
  }

  const runDiagnostics = async () => {
    setDiagRunning(true)
    setError(null)
    setDiagText(null)
    try {
      const envInfo = supabaseEnv.ok
        ? { supabaseUrl: supabaseEnv.values.VITE_SUPABASE_URL, anonKeyPrefix: `${supabaseEnv.values.VITE_SUPABASE_ANON_KEY.slice(0, 12)}…` }
        : { error: 'missing_supabase_env', missing: supabaseEnv.missing }

      const { data: sessData, error: sessErr } = await supabase.auth.getSession()
      const sess = sessData.session
      const access = sess?.access_token ?? null
      const tokenInfo = access
        ? {
            prefix: `${access.slice(0, 12)}…`,
            length: access.length,
            hasDots: access.split('.').length === 3,
            expiresAt: sess?.expires_at ?? null,
            claims: decodeJwtPayload(access),
          }
        : null

      const { data: userData, error: userErr } = await supabase.auth.getUser()

      const edge = await callAdminWhatsapp({ action: 'admin_diagnostics' })
      if (!aliveRef.current) return
      const edgeBody = typeof edge.body === 'string' ? edge.body : JSON.stringify(edge.body, null, 2)
      setDiagText(
        JSON.stringify(
          {
            env: envInfo,
            session: {
              ok: !sessErr,
              error: sessErr?.message ?? null,
              hasSession: Boolean(sess),
              userId: sess?.user?.id ?? null,
              token: tokenInfo,
            },
            auth: {
              ok: !userErr,
              error: userErr?.message ?? null,
              userId: userData.user?.id ?? null,
              email: userData.user?.email ?? null,
```

```
          },
          edge: {
            ok: edge.ok,
            status: edge.status,
            fn: edge.fnVersion,
            elapsedMs: edge.elapsedMs,
            body: edgeBody,
          },
        },
        null,
        2
      )
    )
  } catch (e: unknown) {
    if (!aliveRef.current) return
    setDiagText(JSON.stringify({ ok: false, error: e instanceof Error ? e.message : 'Falha no diagnóstico' }, null,
2))
  } finally {
    if (aliveRef.current) setDiagRunning(false)
  }
}

const checkStatus = async () => {
  if (!conectado) {
    setError('Preencha a Evolution API (URL e Key), clique em Salvar e tente novamente.')
    return
  }
  setCheckingStatus(true)
  setError(null)
  setQrBase64(null)
  setPairingCode(null)
  try {
    const res = await callAdminWhatsapp({ action: 'admin_status' })
    if (!aliveRef.current) return
    if (!res.ok) {
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'whatsapp_not_configured') {
        setError('WhatsApp ainda não foi configurado. Preencha a Evolution API acima e clique em Salvar.')
        return
      }
      if (
        typeof res.body === 'object' &&
        res.body !== null &&
        (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
      ) {
        setError(
          'O Supabase rejeitou o JWT antes de executar a Edge Function. Isso costuma acontecer quando a função está
com verify_jwt=true no deploy (ou quando o app aponta para outro projeto). Faça logout/login e confirme se o deploy da
função whatsapp está com verify_jwt=false.'
        )
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).hint) {
        const hint = String((res.body as Record<string, unknown>).hint)
        const details = JSON.stringify(res.body)
        setError(`Falha ao verificar status (HTTP ${res.status}): ${details}\n\nDica: ${hint}`)
        return
      }
      const details =
        typeof res.body === 'string'
          ? res.body.toLowerCase().includes('cloudflare tunnel error')
            ? 'Cloudflare Tunnel error (HTML)'
            : res.body.length > 900
              ? `${res.body.slice(0, 900)}…`
              : res.body
          : JSON.stringify(res.body)
      setError(`Falha ao verificar status (HTTP ${res.status}): ${details}`)
      return
    }
    const obj = res.body as Record<string, unknown>
    setInstanceState(typeof obj.state === 'string' ? obj.state : null)
  } catch (e: unknown) {
```

```
      if (!aliveRef.current) return
      setError(e instanceof Error ? e.message : 'Falha ao verificar status')
    } finally {
      if (aliveRef.current) setCheckingStatus(false)
    }
  }

  const connect = async () => {
    if (!conectado) {
      setError('Preencha a Evolution API (URL e Key), clique em Salvar e tente novamente.')
      return
    }
    setConnecting(true)
    setError(null)
    setQrBase64(null)
    setPairingCode(null)

    try {
      const res = await callAdminWhatsapp({ action: 'admin_connect', number: null })
      if (!aliveRef.current) return
      if (!res.ok) {
        if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'whatsapp_not_configured') {
          setError('WhatsApp ainda não foi configurado. Preencha a Evolution API acima e clique em Salvar.')
          return
        }
        if (
          typeof res.body === 'object' &&
          res.body !== null &&
          (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
        ) {
          setError(
            'O Supabase rejeitou o JWT antes de executar a Edge Function. Isso costuma acontecer quando a função está
com verify_jwt=true no deploy (ou quando o app aponta para outro projeto). Faça logout/login e confirme se o deploy da
função whatsapp está com verify_jwt=false.'
          )
          return
        }
        if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).hint) {
          const hint = String((res.body as Record<string, unknown>).hint)
          const details = JSON.stringify(res.body)
          setError(`Falha ao gerar QR Code (HTTP ${res.status}): ${details}\n\nDica: ${hint}`)
          return
        }
        const details =
          typeof res.body === 'string'
            ? res.body.toLowerCase().includes('cloudflare tunnel error')
              ? 'Cloudflare Tunnel error (HTML)'
              : res.body.length > 900
                ? `${res.body.slice(0, 900)}…`
                : res.body
            : JSON.stringify(res.body)
        setError(`Falha ao gerar QR Code (HTTP ${res.status}): ${details}`)
        return
      }
      const obj = res.body as Record<string, unknown>
      setInstanceState(typeof obj.state === 'string' ? obj.state : null)
      setQrBase64(typeof obj.qrBase64 === 'string' ? obj.qrBase64 : null)
      setPairingCode(typeof obj.pairingCode === 'string' ? obj.pairingCode : null)
    } catch (e: unknown) {
      if (!aliveRef.current) return
      setError(e instanceof Error ? e.message : 'Falha ao gerar QR Code')
    } finally {
      if (aliveRef.current) setConnecting(false)
    }
  }

  const disconnect = async () => {
    if (!conectado) {
      setError('Preencha a Evolution API (URL e Key), clique em Salvar e tente novamente.')
      return
    }
```

```
    setDisconnecting(true)
    setError(null)
    try {
      const res = await callAdminWhatsapp({ action: 'admin_disconnect' })
      if (!aliveRef.current) return
      if (!res.ok) {
        if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error === 'whatsapp_not_configured') {
          setError('WhatsApp ainda não foi configurado. Preencha a Evolution API acima e clique em Salvar.')
          return
        }
        if (
          typeof res.body === 'object' &&
          res.body !== null &&
          (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
        ) {
          setError(
            'O Supabase rejeitou o JWT antes de executar a Edge Function. Isso costuma acontecer quando a função está com verify_jwt=true no deploy (ou quando o app aponta para outro projeto). Faça logout/login e confirme se o deploy da função whatsapp está com verify_jwt=false.'
          )
          return
        }
        if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).hint) {
          const hint = String((res.body as Record<string, unknown>).hint)
          const details = JSON.stringify(res.body)
          setError(`Falha ao desconectar (HTTP ${res.status}): ${details}\n\nDica: ${hint}`)
          return
        }
        const details = typeof res.body === 'string' ? res.body : JSON.stringify(res.body)
        setError(`Falha ao desconectar (HTTP ${res.status}): ${details}`)
        return
      }
      setInstanceState(null)
      setQrBase64(null)
      setPairingCode(null)
    } catch (e: unknown) {
      if (!aliveRef.current) return
      setError(e instanceof Error ? e.message : 'Falha ao desconectar')
    } finally {
      if (aliveRef.current) setDisconnecting(false)
    }
  }

  const selectAllFiltered = () => {
    const next: Record<string, boolean> = { ...selected }
    for (const c of filteredClientes) next[c.id] = true
    setSelected(next)
  }

  const clearSelection = () => {
    setSelected({})
  }

  const sendAvisos = async () => {
    const text = mensagem.trim()
    if (!text) {
      setError('Digite a mensagem.')
      return
    }
    if (selectedCount === 0) {
      setError('Selecione pelo menos 1 cliente.')
      return
    }
    setSending(true)
    setError(null)
    setLastSendSummary(null)
    try {
      const res = await callAdminWhatsapp({ action: 'admin_send_aviso', cliente_ids: selectedIds, text })
      if (!aliveRef.current) return
      if (!res.ok) {
        if (
```

```
          typeof res.body === 'object' &&
          res.body !== null &&
          (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
        ) {
          setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
          return
        }
        if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).hint) {
          const hint = String((res.body as Record<string, unknown>).hint)
          const details = JSON.stringify(res.body)
          setError(`Falha ao enviar (HTTP ${res.status}): ${details}\n\nDica: ${hint}`)
          return
        }
        const details = typeof res.body === 'string' ? res.body : JSON.stringify(res.body)
        setError(`Falha ao enviar (HTTP ${res.status}): ${details}`)
        return
      }

      const obj = res.body as Record<string, unknown>
    const totals = (obj.totals ?? null) as Record<string, unknown> | null
    const sent = typeof totals?.sent === 'number' ? totals.sent : null
    const failed = typeof totals?.failed === 'number' ? totals.failed : null
    const skippedNoPhone = typeof totals?.skipped_no_phone === 'number' ? totals.skipped_no_phone : null
    const skippedInvalidPhone = typeof totals?.skipped_invalid_phone === 'number' ? totals.skipped_invalid_phone : null
    const parts = [
      sent !== null ? `enviados=${sent}` : null,
      failed !== null ? `falhas=${failed}` : null,
      skippedNoPhone !== null ? `sem_telefone=${skippedNoPhone}` : null,
      skippedInvalidPhone !== null ? `telefone_invalido=${skippedInvalidPhone}` : null,
    ].filter(Boolean)
    setLastSendSummary(parts.length ? parts.join(' • ') : 'Enviado.')
    } catch (e: unknown) {
      if (!aliveRef.current) return
      setError(e instanceof Error ? e.message : 'Falha ao enviar')
    } finally {
      if (aliveRef.current) setSending(false)
    }
  }

  return (
    <AdminShell>
      <div className="space-y-6">
        <div>
          <div className="text-sm font-semibold text-slate-500">WhatsApp</div>
          <div className="text-xl font-semibold text-slate-900">Avisos para clientes</div>
        </div>

        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}
        {configSaved ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">Salvo.</div> : null}
        {lastSendSummary ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">{lastSendSummary}</div> : null}

        {schemaIncompleto ? (
          <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-800">
            Seu Supabase ainda não tem as colunas de WhatsApp no Super Admin. Acesse <span className="font-mono text-xs">/admin/configuracoes</span> e execute o bloco “SQL do WhatsApp (Super Admin)”.
          </div>
        ) : null}

        <Card>
          <div className="p-6 space-y-4">
            <div className="text-sm font-semibold text-slate-900">Configuração</div>
            {configLoading ? <div className="text-sm text-slate-600">Carregando…</div> : null}
            <Input label="API URL" value={apiUrl} onChange={(e) => setApiUrl(e.target.value)} placeholder="https://..." />
            <Input label="API Key" value={apiKey} onChange={(e) => setApiKey(e.target.value)} placeholder="..." />
            <Input label="Instance name (opcional)" value={instanceName} onChange={(e) => setInstanceName(e.target.value)} placeholder="smagenda-admin" />
            <div className="flex justify-end">
              <Button onClick={saveConfig} disabled={savingConfig || schemaIncompleto}>
```

```
                Salvar
              </Button>
            </div>
          </div>
        </Card>

        <Card>
          <div className="p-6 space-y-4">
            <div className="text-sm font-semibold text-slate-900">Status e conexão</div>
            <div className="text-sm text-slate-700">Configuração: {conectado ? '🟢 OK' : '🟡 Pendente'}</div>
            <div className="text-sm text-slate-700">Conexão: {checkingStatus ? 'verificando…' : instanceState ?? '—'}
</div>
            <div className="flex flex-wrap gap-2">
              <Button variant="secondary" onClick={checkStatus} disabled={checkingStatus || connecting ||
disconnecting}>
                Atualizar status
              </Button>
              <Button onClick={connect} disabled={connecting || disconnecting}>
                Conectar
              </Button>
              <Button variant="secondary" onClick={disconnect} disabled={disconnecting || connecting}>
                Desconectar
              </Button>
            </div>
            {pairingCode ? (
              <div className="rounded-xl border border-slate-200 bg-white p-4">
                <div className="text-sm font-semibold text-slate-900">Código de pareamento</div>
                <div className="mt-1 font-mono text-lg tracking-wider text-slate-900">{pairingCode}</div>
              </div>
            ) : null}
            {qrBase64 ? (
              <div className="rounded-xl border border-slate-200 bg-white p-4">
                <div className="text-sm font-semibold text-slate-900">QR Code</div>
                <div className="mt-3 flex justify-center">
                  <img src={`data:image/png;base64,${qrBase64}`} alt="QR Code" className="h-64 w-64" />
                </div>
              </div>
            ) : null}
          </div>
        </Card>

        <Card>
          <div className="p-6 space-y-4">
            <div className="flex flex-wrap items-center justify-between gap-3">
              <div>
                <div className="text-sm font-semibold text-slate-900">Diagnóstico</div>
                <div className="text-sm text-slate-600">Valida sessão, Super Admin e Evolution API.</div>
              </div>
              <Button variant="secondary" onClick={() => void runDiagnostics()} disabled={diagRunning}>
                {diagRunning ? 'Executando…' : 'Executar diagnóstico'}
              </Button>
            </div>
            {diagText ? (
              <pre className="whitespace-pre-wrap break-words rounded-xl border border-slate-200 bg-slate-50 p-3 text-xs
text-slate-800">{diagText}</pre>
            ) : null}
          </div>
        </Card>

        <Card>
          <div className="p-6 space-y-4">
            <div className="flex flex-wrap items-center justify-between gap-3">
              <div>
                <div className="text-sm font-semibold text-slate-900">Destinatários</div>
                <div className="text-sm text-slate-600">Selecionados: {selectedCount}</div>
              </div>
              <div className="flex flex-wrap items-center gap-2">
                <Button
                  onClick={() => void enableWhatsappSelected(true)}
                  disabled={selectedCount === 0 || updatingClientesWhatsapp || schemaIncompleto}
                >
                  Habilitar WhatsApp
```

```
                </Button>
                <Button variant="secondary" onClick={() => void enableWhatsappSelected(false)} disabled={selectedCount
=== 0 || updatingClientesWhatsapp}>
                  Desabilitar WhatsApp
                </Button>
                <Button
                  variant="secondary"
                  onClick={() => void enableTemaProspectorSelected(true)}
                  disabled={selectedCount === 0 || updatingClientesTemaProspector}
                >
                  Habilitar Tema Prospector
                </Button>
                <Button
                  variant="secondary"
                  onClick={() => void enableTemaProspectorSelected(false)}
                  disabled={selectedCount === 0 || updatingClientesTemaProspector}
                >
                  Desabilitar Tema Prospector
                </Button>
                <Button variant="secondary" onClick={selectAllFiltered} disabled={clientesLoading ||
filteredClientes.length === 0}>
                  Selecionar filtrados
                </Button>
                <Button variant="secondary" onClick={clearSelection} disabled={selectedCount === 0}>
                  Limpar
                </Button>
              </div>
            </div>

            {schemaClientesWhatsappIncompleto ? (
              <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-800">
                Seu Supabase ainda não tem a coluna de habilitação do WhatsApp por cliente. Execute o bloco “SQL do
WhatsApp (habilitação por cliente)” em <span className="font-mono text-xs">/admin/configuracoes</span>.
              </div>
            ) : null}

            {schemaClientesTemaProspectorIncompleto ? (
              <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-800">
                Seu Supabase ainda não tem a coluna de habilitação do Tema Prospector por cliente. Execute o bloco “SQL
do Tema Prospector (habilitação por cliente)” em <span className="font-mono text-xs">/admin/configuracoes</span>.
              </div>
            ) : null}

            <Input label="Buscar" value={query} onChange={(e) => setQuery(e.target.value)} placeholder="Nome, slug ou
telefone" />

            <div className="rounded-xl border border-slate-200 bg-white">
              <div className="divide-y divide-slate-100">
                {clientesLoading ? (
                  <div className="p-4 text-sm text-slate-600">Carregando…</div>
                ) : filteredClientes.length === 0 ? (
                  <div className="p-4 text-sm text-slate-600">Nenhum cliente.</div>
                ) : (
                  filteredClientes.map((c) => (
                    <label key={c.id} className="flex items-start gap-3 p-4 hover:bg-slate-50 cursor-pointer">
                      <input
                        type="checkbox"
                        checked={selected[c.id] === true}
                        onChange={(e) => setSelected((prev) => ({ ...prev, [c.id]: e.target.checked }))}
                        className="mt-1"
                      />
                      <div className="flex-1">
                        <div className="flex items-start justify-between gap-3">
                          <div>
                            <div className="text-sm font-semibold text-slate-900">{c.nome_negocio}</div>
                            <div className="text-sm text-slate-600">/{c.slug}</div>
                            {c.telefone ? <div className="text-sm text-slate-600">{c.telefone}</div> : <div
className="text-sm text-slate-600">Sem telefone</div>}
                          </div>
                        <div className="flex items-center gap-2">
                          {c.whatsapp_habilitado === true ? <Badge tone="green">WhatsApp</Badge> : <Badge
tone="yellow">Sem WhatsApp</Badge>}
```

```
                          {c.tema_prospector_habilitado === true ? <Badge tone="slate">Tema Prospector</Badge> : null}
                          {c.ativo ? <Badge tone="green">Ativo</Badge> : <Badge tone="red">Inativo</Badge>}
                          <Badge tone="slate">{normalizePlanoLabel(c.plano)}</Badge>
                          <Badge tone={c.status_pagamento === 'inadimplente' ? 'red' : 'slate'}>{c.status_pagamento}
</Badge>
                        </div>
                        </div>
                      </div>
                    </label>
                  ))
                )}
              </div>
            </div>
          </div>
        </Card>

        <Card>
          <div className="p-6 space-y-4">
            <div className="text-sm font-semibold text-slate-900">Mensagem</div>
            <textarea
              className="w-full min-h-[140px] rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-
900 outline-none focus:ring-2 focus:ring-slate-300"
              value={mensagem}
              onChange={(e) => setMensagem(e.target.value)}
              placeholder="Digite o aviso..."
            />
            <div className="flex justify-end">
              <Button onClick={sendAvisos} disabled={sending || selectedCount === 0 || !conectado}>
                Enviar avisos
              </Button>
            </div>
            {!conectado ? <div className="text-sm text-slate-600">Configure a Evolution API e conecte o WhatsApp para
enviar.</div> : null}
          </div>
        </Card>
      </div>
    </AdminShell>
  )
}
```

smagenda/src/views/app/ClienteDetalhesPage.tsx

```
import { useCallback, useEffect, useMemo, useState } from 'react'
import { Link, useNavigate, useParams } from 'react-router-dom'
import { AppShell } from '../../components/layout/AppShell'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { formatBRMoney, parseTimeToMinutes } from '../../lib/dates'
import { supabase } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type Agendamento = {
  id: string
  cliente_nome: string | null
  cliente_telefone: string | null
  extras: Record<string, unknown> | null
  data: string
  hora_inicio: string
  hora_fim: string | null
  status: string
  servico: { id: string; nome: string; preco: number; cor: string | null } | null
}

function readExtrasText(extras: unknown, key: string) {
  if (!extras || typeof extras !== 'object') return ''
  const v = (extras as Record<string, unknown>)[key]
  if (typeof v !== 'string') return ''
  const t = v.trim()
  return t ? t : ''
}

function normalizeEmail(value: string) {
  const v = String(value ?? '').trim().toLowerCase()
  if (!v) return ''
  const ok = /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(v)
  return ok ? v : ''
}

function resolveStatusUi(status: string): { label: string; tone: 'slate' | 'green' | 'yellow' | 'red' } {
  const s = status.trim().toLowerCase()
  if (s === 'confirmado') return { label: 'Confirmado', tone: 'green' }
  if (s === 'cancelado') return { label: 'Cancelado', tone: 'red' }
  if (s === 'pendente') return { label: 'Pendente', tone: 'yellow' }
  if (s === 'concluido' || s === 'concluído') return { label: 'Concluído', tone: 'green' }
  if (s === 'nao_compareceu' || s === 'não_compareceu' || s === 'no_show') return { label: 'No-show', tone: 'red' }
  return { label: status || 'Status', tone: 'slate' }
}

export function ClienteDetalhesPage() {
  const { appPrincipal, masterUsuario, masterUsuarioLoading } = useAuth()
  const funcionario = appPrincipal?.kind === 'funcionario' ? appPrincipal.profile : null
  const isGerente = appPrincipal?.kind === 'funcionario' && appPrincipal.profile.permissao === 'admin'
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : isGerente ? masterUsuario : null
  const usuarioId = usuario?.id ?? null
  const navigate = useNavigate()

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Resumo do cliente',
          body: 'Aqui você vê o total de agendamentos, no-show e o último atendimento do cliente.',
          target: 'header' as const,
        },
        {
          title: 'Detalhes rápidos',
          body: 'O cartão mostra cancelados e o valor previsto do histórico (com base nos serviços).',
          target: 'summary' as const,
        },
```

```
        {
          title: 'Histórico',
          body: 'A lista abaixo traz todos os agendamentos do cliente, com serviço e status.',
          target: 'list' as const,
        },
      ] as const,
    []
  )

  const params = useParams()
  const telefone = useMemo(() => {
    try {
      return decodeURIComponent(params.telefone ?? '').trim()
    } catch {
      return (params.telefone ?? '').trim()
    }
  }, [params.telefone])

  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [agendamentos, setAgendamentos] = useState<Agendamento[]>([])

  const [deleting, setDeleting] = useState(false)

  const [editAgendamentoId, setEditAgendamentoId] = useState<string | null>(null)
  const [editEndereco, setEditEndereco] = useState('')
  const [editPlaca, setEditPlaca] = useState('')
  const [editEmail, setEditEmail] = useState('')
  const [editSaving, setEditSaving] = useState(false)
  const [editError, setEditError] = useState<string | null>(null)

  const load = useCallback(async () => {
    if (!usuarioId) {
      setLoading(false)
      return
    }
    if (!telefone) {
      setError('Telefone inválido')
      setLoading(false)
      return
    }

    setLoading(true)
    setError(null)

    const { data, error: err } = await supabase
      .from('agendamentos')

.select('id,cliente_nome,cliente_telefone,extras,data,hora_inicio,hora_fim,status,servico:servico_id(id,nome,preco,cor)'
)
      .eq('usuario_id', usuarioId)
      .eq('cliente_telefone', telefone)
      .order('data', { ascending: false })
      .order('hora_inicio', { ascending: false })
      .limit(2000)

    if (err) {
      setError(err.message)
      setLoading(false)
      return
    }
    setAgendamentos((data ?? []) as unknown as Agendamento[])
    setLoading(false)
  }, [telefone, usuarioId])

  const ocultarCliente = async () => {
    if (!usuarioId) return
    if (!telefone) return
    setError(null)
    const ok = window.confirm(
      'Ocultar cliente? Isso vai remover o telefone dos agendamentos desse número, e ele não aparecerá mais na lista de
clientes.'
```

```
    )
    if (!ok) return

    setDeleting(true)
    const { error: updErr } = await supabase
      .from('agendamentos')
      .update({ cliente_telefone: '', cliente_nome: 'Cliente oculto' })
      .eq('usuario_id', usuarioId)
      .eq('cliente_telefone', telefone)

    if (updErr) {
      setError(updErr.message)
      setDeleting(false)
      return
    }

    setDeleting(false)
    navigate('/clientes')
  }

  useEffect(() => {
    void (async () => {
      await Promise.resolve()
      await load()
    })().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar histórico')
      setLoading(false)
    })
  }, [load])

  const openEditExtras = (ag: Agendamento) => {
    setEditAgendamentoId(ag.id)
    setEditEndereco(readExtrasText(ag.extras, 'endereco'))
    setEditPlaca(readExtrasText(ag.extras, 'placa'))
    setEditEmail(readExtrasText(ag.extras, 'email'))
    setEditError(null)
  }

  const closeEditExtras = () => {
    setEditAgendamentoId(null)
    setEditEndereco('')
    setEditPlaca('')
    setEditEmail('')
    setEditError(null)
  }

  const saveEditExtras = async () => {
    if (!usuarioId || !editAgendamentoId) return
    setEditSaving(true)
    setEditError(null)

    const target = agendamentos.find((a) => a.id === editAgendamentoId) ?? null
    if (!target) {
      setEditSaving(false)
      closeEditExtras()
      return
    }

    const endereco = editEndereco.trim()
    const placa = editPlaca.trim().replace(/[^a-zA-Z0-9-]/g, '').toUpperCase()
    const email = normalizeEmail(editEmail)
    if (editEmail.trim() && !email) {
      setEditError('Email inválido.')
      setEditSaving(false)
      return
    }

    const base = target.extras && typeof target.extras === 'object' && !Array.isArray(target.extras) ? { ...
(target.extras as Record<string, unknown>) } : {}
    if (endereco) base.endereco = endereco
    else delete base.endereco
    if (placa) base.placa = placa
```

```
    else delete base.placa
    if (email) base.email = email
    else delete base.email

    const extrasFinal = Object.keys(base).length > 0 ? base : null

    const { error: updErr } = await supabase.from('agendamentos').update({ extras: extrasFinal }).eq('id',
editAgendamentoId).eq('usuario_id', usuarioId)
    if (updErr) {
      setEditError(updErr.message)
      setEditSaving(false)
      return
    }

    setAgendamentos((prev) => prev.map((a) => (a.id === editAgendamentoId ? { ...a, extras: extrasFinal as
Record<string, unknown> | null } : a)))
    setEditSaving(false)
    closeEditExtras()
  }

  const resumo = useMemo(() => {
    const total = agendamentos.filter((a) => a.status !== 'cancelado').length
    const cancelados = agendamentos.filter((a) => a.status === 'cancelado').length
    const noShows = agendamentos.filter((a) => {
      const s = (a.status ?? '').toLowerCase()
      return s === 'nao_compareceu' || s === 'não_compareceu' || s === 'no_show'
    }).length
    const nome = agendamentos.find((a) => (a.cliente_nome ?? '').trim())?.cliente_nome?.trim() ?? 'Cliente'
    const nomes = Array.from(
      new Set(
        agendamentos
          .map((a) => (a.cliente_nome ?? '').trim())
          .filter(Boolean)
          .map((n) => n.toLowerCase())
      )
    )
      .map((lower) => agendamentos.find((a) => (a.cliente_nome ?? '').trim().toLowerCase() ===
lower)?.cliente_nome?.trim() ?? lower)
      .filter(Boolean)
    const ultimoDia = agendamentos[0]?.data ?? null
    const totalPrevisto = agendamentos
      .filter((a) => a.status !== 'cancelado')
      .reduce((sum, a) => sum + (a.servico?.preco ? Number(a.servico.preco) : 0), 0)
    return { total, cancelados, noShows, nome, nomes, ultimoDia, totalPrevisto }
  }, [agendamentos])

  if (!usuario) {
    return (
      <AppShell>
        <div className="text-slate-700">{isGerente && masterUsuarioLoading ? 'Carregando…' : 'Acesso restrito.'}</div>
      </AppShell>
    )
  }

  if (funcionario && !funcionario.pode_ver_clientes_de_outros) {
    return (
      <AppShell>
        <div className="text-slate-700">Acesso restrito.</div>
      </AppShell>
    )
  }

  return (
    <PageTutorial usuarioId={usuarioId} page="cliente_detalhes">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="space-y-6">
            {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}
</div> : null}

            <div
              className={
```

```
    tutorialOpen && tutorialSteps[tutorialStep]?.target === 'header'
      ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl p-2 -m-2'
      : ''
  }
>
  <div className="flex items-center justify-between gap-3">
    <div>
      <div className="text-sm text-slate-600">
        <Link to="/clientes" className="hover:underline">
          Clientes
        </Link>
        <span className="text-slate-400"> / </span>
        <span className="text-slate-700">{telefone || 'Detalhes'}</span>
      </div>
      <div className="text-lg font-semibold text-slate-900">{resumo.nome}</div>
      <div className="text-sm text-slate-600">📱 {telefone}</div>
    </div>
    <div className="flex flex-wrap items-center justify-end gap-2">
      <Button variant="secondary" onClick={resetTutorial}>
        Rever tutorial
      </Button>
      <Button variant="secondary" onClick={ocultarCliente} disabled={loading || deleting}>
        Ocultar cliente
      </Button>
      <Badge tone="slate">{resumo.total} ag.</Badge>
      {resumo.noShows > 0 ? <Badge tone="red">No-show {resumo.noShows}</Badge> : null}
      {resumo.nomes.length > 1 ? <Badge tone="yellow">Nomes {resumo.nomes.length}</Badge> : null}
    </div>
  </div>
</div>

{resumo.nomes.length > 1 ? (
  <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-900">
    <div className="font-semibold">Atenção: este telefone aparece com nomes diferentes</div>
    <div className="text-amber-800">{resumo.nomes.join(' • ')}</div>
  </div>
) : null}

<div
  className={
    tutorialOpen && tutorialSteps[tutorialStep]?.target === 'summary'
      ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
      : ''
  }
>
  <Card>
    <div className="p-6 grid grid-cols-1 gap-3 sm:grid-cols-3">
      <div>
        <div className="text-xs font-semibold text-slate-500">Último agendamento</div>
        <div className="text-sm font-semibold text-slate-900">
          {resumo.ultimoDia ? new Date(resumo.ultimoDia).toLocaleDateString('pt-BR') : '—'}
        </div>
      </div>
      <div>
        <div className="text-xs font-semibold text-slate-500">Cancelados</div>
        <div className="text-sm font-semibold text-slate-900">{resumo.cancelados}</div>
      </div>
      <div>
        <div className="text-xs font-semibold text-slate-500">Valor previsto (histórico)</div>
        <div className="text-sm font-semibold text-slate-900">{formatBRMoney(resumo.totalPrevisto)}</div>
      </div>
    </div>
  </Card>
</div>

<div
  className={
    tutorialOpen && tutorialSteps[tutorialStep]?.target === 'list'
      ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
      : ''
  }
>
```

```
              <Card>
                <div className="divide-y divide-slate-100">
                  {loading ? (
                    <div className="p-6 text-sm text-slate-600">Carregando…</div>
                  ) : agendamentos.length === 0 ? (
                    <div className="p-6 space-y-3">
                      <div className="text-sm font-semibold text-slate-900">Sem histórico para este telefone</div>
                      <div className="text-sm text-slate-600">Se o cliente já agendou, confirme se o telefone está igual no agendamento.</div>
                      <div>
                        <Button variant="secondary" onClick={() => navigate('/dashboard')}>
                          Ir para Agenda
                        </Button>
                      </div>
                    </div>
                  ) : (
                    agendamentos
                      .slice()
                      .sort((a, b) => {
                        const dateCmp = b.data.localeCompare(a.data)
                        if (dateCmp !== 0) return dateCmp
                        return parseTimeToMinutes(b.hora_inicio) - parseTimeToMinutes(a.hora_inicio)
                      })
                      .map((a) => {
                        const statusUi = resolveStatusUi(a.status)
                        const endereco = readExtrasText(a.extras, 'endereco')
                        const placa = readExtrasText(a.extras, 'placa')
                        const email = readExtrasText(a.extras, 'email')
                        const editing = editAgendamentoId === a.id
                        return (
                          <div key={a.id} className="p-4 space-y-3">
                            <div className="flex items-start justify-between gap-3">
                              <div>
                                <div className="text-sm font-semibold text-slate-900 flex items-center gap-2">
                                  {a.servico?.cor ? <span className="h-2.5 w-2.5 rounded-full" style={{ backgroundColor: a.servico.cor }} /> : null}
                                  <span>
                                    {new Date(a.data).toLocaleDateString('pt-BR')} • {a.hora_inicio}
                                  </span>
                                </div>
                                <div className="text-sm text-slate-600">✂️ {a.servico?.nome ?? 'Serviço'}</div>
                                {a.servico?.preco ? <div className="text-xs text-slate-500"> {formatBRMoney(Number(a.servico.preco))}</div> : null}
                                {endereco ? <div className="text-sm text-slate-600">📍 {endereco}</div> : null}
                                {placa ? <div className="text-sm text-slate-600">🚗 {placa}</div> : null}
                                {email ? <div className="text-sm text-slate-600">✉️ {email}</div> : null}
                              </div>
                              <div className="flex flex-col items-end gap-2">
                                <Badge tone={statusUi.tone}>{statusUi.label}</Badge>
                                <Button variant="secondary" onClick={() => (editing ? closeEditExtras() : openEditExtras(a))}>
                                  {editing ? 'Fechar' : 'Editar dados'}
                                </Button>
                              </div>
                            </div>

                            {editing ? (
                              <div className="rounded-xl border border-slate-200 bg-white p-3 space-y-3">
                                {editError ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{editError}</div> : null}
                                <div className="grid grid-cols-1 gap-3 sm:grid-cols-2">
                                  <Input label="Endereço" value={editEndereco} onChange={(e) => setEditEndereco(e.target.value)} />
                                  <Input label="Placa" value={editPlaca} onChange={(e) => setEditPlaca(e.target.value)} />
                                  <Input label="Email" value={editEmail} onChange={(e) => setEditEmail(e.target.value)} />
                                </div>
                                <div className="flex flex-wrap gap-2">
                                  <Button onClick={saveEditExtras} disabled={editSaving}>
                                    {editSaving ? 'Salvando…' : 'Salvar'}
                                  </Button>
```

```
                                <Button variant="secondary" onClick={closeEditExtras} disabled={editSaving}>
                                  Cancelar
                                </Button>
                              </div>
                            </div>
                          ) : null}
                        </div>
                      )
                    })
                  )}
                </div>
              </Card>
            </div>
          </div>

          <TutorialOverlay
            open={tutorialOpen}
            steps={tutorialSteps}
            step={tutorialStep}
            onStepChange={setTutorialStep}
            onClose={closeTutorial}
            titleFallback="Cliente"
          />
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/ClientesPage.tsx

```
import { useCallback, useEffect, useMemo, useState } from 'react'
import { Link, useNavigate } from 'react-router-dom'
import { AppShell } from '../../components/layout/AppShell'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { supabase } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type AgendamentoRow = {
  cliente_nome: string | null
  cliente_telefone: string | null
  data: string
  hora_inicio: string
  status: string
}

type ClienteResumo = {
  telefone: string
  nome: string
  nomes: string[]
  total: number
  noShows: number
  cancelados: number
  ultimoDia: string
}

function normalizeText(value: string) {
  return value
    .toLowerCase()
    .normalize('NFD')
    .replace(/\p{Diacritic}/gu, '')
}

export function ClientesPage() {
  const { appPrincipal, masterUsuario, masterUsuarioLoading } = useAuth()
  const funcionario = appPrincipal?.kind === 'funcionario' ? appPrincipal.profile : null
  const isGerente = appPrincipal?.kind === 'funcionario' && appPrincipal.profile.permissao === 'admin'
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : isGerente ? masterUsuario : null
  const usuarioId = usuario?.id ?? null
  const navigate = useNavigate()

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Clientes vêm dos agendamentos',
          body: 'A lista é gerada a partir dos agendamentos do seu negócio. Faça 1 agendamento para ver seus primeiros
clientes aqui.',
          target: 'header' as const,
        },
        {
          title: 'Buscar rápido',
          body: 'Use a busca por nome ou telefone para encontrar um cliente em segundos.',
          target: 'header' as const,
        },
        {
          title: 'Abrir histórico',
          body: 'Toque em um cliente para ver o histórico de agendamentos e métricas (no-show, cancelados,
recorrência).',
          target: 'list' as const,
        },
      ] as const,
    []
  )

  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
```

```
  const [agendamentos, setAgendamentos] = useState<AgendamentoRow[]>([])
  const [search, setSearch] = useState('')

  const load = useCallback(async () => {
    if (!usuarioId) return
    setLoading(true)
    setError(null)

    const { data, error: err } = await supabase
      .from('agendamentos')
      .select('cliente_nome,cliente_telefone,data,hora_inicio,status')
      .eq('usuario_id', usuarioId)
      .not('cliente_telefone', 'is', null)
      .neq('cliente_telefone', '')
      .order('data', { ascending: false })
      .order('hora_inicio', { ascending: false })
      .limit(5000)

    if (err) {
      setError(err.message)
      setLoading(false)
      return
    }
    setAgendamentos((data ?? []) as unknown as AgendamentoRow[])
    setLoading(false)
  }, [usuarioId])

  useEffect(() => {
    void (async () => {
      await Promise.resolve()
      await load()
    })().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar clientes')
      setLoading(false)
    })
  }, [load])

  const clientes = useMemo(() => {
    const map = new Map<string, ClienteResumo>()

    for (const a of agendamentos) {
      const telefone = (a.cliente_telefone ?? '').trim()
      if (!telefone) continue

      const prev = map.get(telefone)
      const nome = (a.cliente_nome ?? '').trim()
      const status = (a.status ?? '').trim().toLowerCase()
      const cancelado = status === 'cancelado'
      const noShow = status === 'nao_compareceu' || status === 'não_compareceu' || status === 'no_show'

      const nomesPrev = prev?.nomes ?? []
      const nomesNext = nome && !nomesPrev.some((n) => n.toLowerCase() === nome.toLowerCase()) ? [...nomesPrev, nome] :
nomesPrev

      const next: ClienteResumo = prev
        ? { ...prev }
        : {
            telefone,
            nome: nome || 'Cliente',
            nomes: nome ? [nome] : [],
            total: 0,
            noShows: 0,
            cancelados: 0,
            ultimoDia: a.data,
          }

      if (nome && (!prev || prev.nome === 'Cliente')) next.nome = nome
      if (nomesNext !== next.nomes) next.nomes = nomesNext
      if (a.data > next.ultimoDia) next.ultimoDia = a.data
      if (cancelado) next.cancelados += 1
      else next.total += 1
      if (noShow) next.noShows += 1
```

```
      map.set(telefone, next)
    }

    return Array.from(map.values()).sort((x, y) => y.ultimoDia.localeCompare(x.ultimoDia))
  }, [agendamentos])

  const filtered = useMemo(() => {
    const q = normalizeText(search.trim())
    if (!q) return clientes
    return clientes.filter((c) => {
      const hay = normalizeText(`${c.nome} ${c.nomes.join(' ')} ${c.telefone}`)
      return hay.includes(q)
    })
  }, [clientes, search])

  if (!usuario) {
    return (
      <AppShell>
        <div className="text-slate-700">{isGerente && masterUsuarioLoading ? 'Carregando…' : 'Acesso restrito.'}</div>
      </AppShell>
    )
  }

  if (funcionario && !funcionario.pode_ver_clientes_de_outros) {
    return (
      <AppShell>
        <div className="text-slate-700">Acesso restrito.</div>
      </AppShell>
    )
  }

  return (
    <PageTutorial usuarioId={usuarioId} page="clientes">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="space-y-6">
            {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}

            <Card>
              <div
                className={
                  tutorialOpen && tutorialSteps[tutorialStep]?.target === 'header'
                    ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                    : ''
                }
              >
                <div className="p-6 space-y-3">
                  <div className="flex items-center justify-between gap-3">
                    <div>
                      <div className="text-sm font-semibold text-slate-900">Clientes</div>
                      <div className="text-xs text-slate-600">Lista baseada nos agendamentos do seu negócio.</div>
                    </div>
                    <div className="flex items-center gap-2">
                      <Button variant="secondary" onClick={resetTutorial}>
                        Rever tutorial
                      </Button>
                      <div className="text-sm font-semibold text-slate-900">{clientes.length}</div>
                    </div>
                  </div>
                  <Input label="Buscar" value={search} onChange={(e) => setSearch(e.target.value)} placeholder="Nome ou telefone" />
                </div>
              </div>
            </Card>

            <Card>
              <div
                className={
                  tutorialOpen && tutorialSteps[tutorialStep]?.target === 'list'
                    ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
```

```
                    : ''
                }
              >
                <div className="divide-y divide-slate-100">
                  {loading ? (
                    <div className="p-6 text-sm text-slate-600">Carregando…</div>
                  ) : clientes.length === 0 ? (
                    <div className="p-6 space-y-3">
                      <div className="text-sm font-semibold text-slate-900">Nenhum cliente ainda</div>
                      <div className="text-sm text-slate-600">Clientes aparecem depois do primeiro agendamento.</div>
                      <div>
                        <Button variant="secondary" onClick={() => navigate('/dashboard')}>
                          Ir para Agenda
                        </Button>
                      </div>
                    </div>
                  ) : filtered.length === 0 ? (
                    <div className="p-6 text-sm text-slate-600">Nenhum cliente encontrado.</div>
                  ) : (
                    filtered.map((c) => {
                      const recorrente = c.total >= 2
                      const hasNomes = c.nomes.length > 1
                      return (
                        <Link key={c.telefone} to={`/clientes/${encodeURIComponent(c.telefone)}`} className="block p-4 hover:bg-slate-50">
                          <div className="flex items-start justify-between gap-3">
                            <div>
                              <div className="text-sm font-semibold text-slate-900">{c.nome}</div>
                              <div className="text-sm text-slate-600">📱 {c.telefone}</div>
                              <div className="text-xs text-slate-500">Último: {new Date(c.ultimoDia).toLocaleDateString('pt-BR')}</div>
                            </div>
                            <div className="flex items-center gap-2">
                              {recorrente ? <Badge tone="green">Recorrente</Badge> : <Badge>Novo</Badge>}
                              {c.noShows > 0 ? <Badge tone="red">No-show {c.noShows}</Badge> : null}
                              {hasNomes ? <Badge tone="yellow">Nomes {c.nomes.length}</Badge> : null}
                              <Badge tone="slate">{c.total}</Badge>
                            </div>
                          </div>
                        </Link>
                      )
                    })
                  )}
                </div>
              </div>
            </Card>
          </div>

          <TutorialOverlay
            open={tutorialOpen}
            steps={tutorialSteps}
            step={tutorialStep}
            onStepChange={setTutorialStep}
            onClose={closeTutorial}
            titleFallback="Clientes"
          />
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/DashboardPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { useLocation, useNavigate } from 'react-router-dom'
import { AppShell } from '../../components/layout/AppShell'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { formatBRMoney, minutesToTime, normalizeTimeHHMM, parseTimeToMinutes, toISODate } from '../../lib/dates'
import { checkJwtProject, supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type Agendamento = {
  id: string
  cliente_nome: string
  cliente_telefone: string
  data: string
  hora_inicio: string
  hora_fim: string | null
  status: string
  funcionario_id: string | null
  extras: Record<string, unknown> | null
  servico: { id: string; nome: string; preco: number; duracao_minutos: number; cor: string | null } | null
}

type Funcionario = { id: string; nome_completo: string; ativo: boolean }

type ServicoOption = { id: string; nome: string; cor: string | null }

type Bloqueio = { id: string; data: string; hora_inicio: string; hora_fim: string; motivo: string | null;
funcionario_id: string | null }

function formatSupabaseError(err: { message: string; details?: string | null; hint?: string | null; code?: string | null
}) {
  const parts = [err.message]
  if (err.code) parts.push(`code=${err.code}`)
  if (err.details) parts.push(err.details)
  if (err.hint) parts.push(err.hint)
  return parts.filter((p) => typeof p === 'string' && p.trim()).join(' • ')
}

function buildSlots(
  start: string,
  end: string,
  intervalStart: string | null,
  intervalEnd: string | null,
  stepMinutes: number
): string[] {
  const startMin = parseTimeToMinutes(start)
  const endMin = parseTimeToMinutes(end)
  const ivStart = intervalStart ? parseTimeToMinutes(intervalStart) : null
  const ivEnd = intervalEnd ? parseTimeToMinutes(intervalEnd) : null
  const slots: string[] = []
  for (let m = startMin; m < endMin; m += stepMinutes) {
    if (ivStart !== null && ivEnd !== null && m >= ivStart && m < ivEnd) continue
    slots.push(minutesToTime(m))
  }
  return slots
}

function readExtrasEndereco(extras: unknown) {
  if (!extras || typeof extras !== 'object') return null
  const v = (extras as Record<string, unknown>).endereco
  if (typeof v !== 'string') return null
  const t = v.trim()
  return t ? t : null
}

function startOfWeek(value: Date) {
  const d = new Date(value)
```

```
  d.setHours(0, 0, 0, 0)
  const day = d.getDay()
  const diff = (day + 6) % 7
  d.setDate(d.getDate() - diff)
  return d
}

function addDays(value: Date, days: number) {
  const d = new Date(value)
  d.setDate(d.getDate() + days)
  return d
}

function addMonths(value: Date, months: number) {
  const d = new Date(value)
  const day = d.getDate()
  d.setDate(1)
  d.setMonth(d.getMonth() + months)
  const lastDay = new Date(d.getFullYear(), d.getMonth() + 1, 0).getDate()
  d.setDate(Math.min(day, lastDay))
  return d
}

function parseDateKey(value: string) {
  const [y, m, d] = value.split('-').map((v) => Number(v))
  const out = new Date(y, m - 1, d)
  out.setHours(0, 0, 0, 0)
  return out
}

function overlaps(startA: number, endA: number, startB: number, endB: number) {
  return startA < endB && endA > startB
}

type RepeatKind = 'none' | 'daily' | 'weekly' | 'monthly'

function buildRepeatKeys(startKey: string, repeat: RepeatKind, untilKey: string | null) {
  if (repeat === 'none') return { keys: [startKey], error: null as string | null }
  if (!untilKey) return { keys: [] as string[], error: 'Selecione a data final da recorrência.' }

  const start = parseDateKey(startKey)
  const until = parseDateKey(untilKey)
  if (until < start) return { keys: [] as string[], error: 'A data final deve ser igual ou posterior ao início.' }

  const keys: string[] = []
  const max = 90
  let cur = start
  while (cur <= until && keys.length < max) {
    keys.push(toISODate(cur))
    cur = repeat === 'daily' ? addDays(cur, 1) : repeat === 'weekly' ? addDays(cur, 7) : addMonths(cur, 1)
  }

  if (cur <= until) {
    return { keys: [] as string[], error: 'Intervalo muito grande. Reduza a data final da recorrência.' }
  }

  return { keys, error: null as string | null }
}

function normalizeText(value: string) {
  return value
    .toLowerCase()
    .normalize('NFD')
    .replace(/\p{Diacritic}/gu, '')
}

function resolveStatusUi(status: string | null | undefined): { label: string; tone: 'slate' | 'green' | 'yellow' | 'red'
} {
  const raw = String(status ?? '')
  const s = raw.trim().toLowerCase()
  if (!s) return { label: 'Pendente', tone: 'yellow' }
  if (s === 'confirmado') return { label: 'Confirmado', tone: 'green' }
```

```
  if (s === 'cancelado') return { label: 'Cancelado', tone: 'red' }
  if (s === 'pendente') return { label: 'Pendente', tone: 'yellow' }
  if (s === 'concluido' || s === 'concluído') return { label: 'Concluído', tone: 'green' }
  if (s === 'nao_compareceu' || s === 'não_compareceu' || s === 'no_show') return { label: 'No-show', tone: 'red' }
  return { label: raw || 'Status', tone: 'slate' }
}

async function sendConfirmacaoWhatsapp(agendamentoId: string) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } }
  }

  const supabaseUrl = supabaseEnv.values.VITE_SUPABASE_URL
  const supabaseAnonKey = supabaseEnv.values.VITE_SUPABASE_ANON_KEY

  const tryRefresh = async () => {
    const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
    if (refreshErr) return null
    return refreshed.session ?? null
  }

  const { data: sessionData } = await supabase.auth.getSession()
  let session = sessionData.session
  const now = Math.floor(Date.now() / 1000)
  const expiresAt = session?.expires_at ?? null

  if (session && expiresAt && expiresAt <= now + 60) {
    const refreshed = await tryRefresh()
    if (refreshed) session = refreshed
  }

  const token = session?.access_token ?? null
  if (!token) {
    return { ok: false as const, status: 401, body: { error: 'session_expired' } }
  }

  const tokenProject = checkJwtProject(token, supabaseUrl)
  if (!tokenProject.ok) {
    await supabase.auth.signOut().catch(() => undefined)
    return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: tokenProject.iss, expected:
tokenProject.expectedPrefix } }
  }

  const callFetch = async (jwt: string) => {
    const fnUrl = `${supabaseUrl}/functions/v1/whatsapp`
    let res: Response
    try {
      res = await fetch(fnUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          apikey=[REDACTED]
          Authorization: `Bearer ${jwt}`,
          'x-user-jwt': jwt,
        },
        body: JSON.stringify({ action: 'send_confirmacao', agendamento_id: agendamentoId }),
      })
    } catch (e: unknown) {
      const msg = e instanceof Error ? e.message : 'Falha de rede'
      return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
    }

    const fnVersion = res.headers.get('x-smagenda-fn')

    const text = await res.text()
    let parsed: unknown = null
    try {
      parsed = text ? JSON.parse(text) : null
    } catch {
      parsed = text
    }
```

```
    if (!res.ok && res.status === 401 && !fnVersion) {
      if (
        parsed &&
        typeof parsed === 'object' &&
        (parsed as Record<string, unknown>).message === 'Invalid JWT' &&
        (parsed as Record<string, unknown>).code === 401
      ) {
        return { ok: false as const, status: 401, body: { error: 'supabase_gateway_invalid_jwt' } }
      }
    }

    if (!res.ok) return { ok: false as const, status: res.status, body: parsed }
    return { ok: true as const, status: res.status, body: parsed }
  }

  const isInvalidJwtPayload = (payload: unknown) => {
    if (typeof payload === 'string') return payload.includes('Invalid JWT')
    if (!payload || typeof payload !== 'object') return false
    const obj = payload as Record<string, unknown>
    return obj.message === 'Invalid JWT' || obj.error === 'invalid_jwt'
  }

  const first = await callFetch(token)
  if (
    !first.ok &&
    first.status === 401 &&
    typeof first.body === 'object' &&
    first.body !== null &&
    (first.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
  ) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) {
      await supabase.auth.signOut().catch(() => undefined)
      return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    }
    const nextProject = checkJwtProject(nextToken, supabaseUrl)
    if (!nextProject.ok) {
      await supabase.auth.signOut().catch(() => undefined)
      return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: nextProject.iss, expected: 
nextProject.expectedPrefix } }
    }
    return callFetch(nextToken)
  }

  if (!first.ok && first.status === 401 && isInvalidJwtPayload(first.body)) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    return callFetch(nextToken)
  }

  return first
}

async function sendCancelamentoWhatsapp(agendamentoId: string) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } }
  }

  const supabaseUrl = supabaseEnv.values.VITE_SUPABASE_URL
  const supabaseAnonKey = supabaseEnv.values.VITE_SUPABASE_ANON_KEY

  const tryRefresh = async () => {
    const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
    if (refreshErr) return null
    return refreshed.session ?? null
  }

  const { data: sessionData } = await supabase.auth.getSession()
  let session = sessionData.session
  const now = Math.floor(Date.now() / 1000)
```

```
  const expiresAt = session?.expires_at ?? null

  if (session && expiresAt && expiresAt <= now + 60) {
    const refreshed = await tryRefresh()
    if (refreshed) session = refreshed
  }

  const token = session?.access_token ?? null
  if (!token) {
    return { ok: false as const, status: 401, body: { error: 'session_expired' } }
  }

  const tokenProject = checkJwtProject(token, supabaseUrl)
  if (!tokenProject.ok) {
    await supabase.auth.signOut().catch(() => undefined)
    return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: tokenProject.iss, expected:
tokenProject.expectedPrefix } }
  }

  const callFetch = async (jwt: string) => {
    const fnUrl = `${supabaseUrl}/functions/v1/whatsapp`
    let res: Response
    try {
      res = await fetch(fnUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          apikey=[REDACTED]
          Authorization: `Bearer ${jwt}`,
          'x-user-jwt': jwt,
        },
        body: JSON.stringify({ action: 'send_cancelamento', agendamento_id: agendamentoId }),
      })
    } catch (e: unknown) {
      const msg = e instanceof Error ? e.message : 'Falha de rede'
      return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
    }

    const fnVersion = res.headers.get('x-smagenda-fn')

    const text = await res.text()
    let parsed: unknown = null
    try {
      parsed = text ? JSON.parse(text) : null
    } catch {
      parsed = text
    }

    if (!res.ok && res.status === 401 && !fnVersion) {
      if (
        parsed &&
        typeof parsed === 'object' &&
        (parsed as Record<string, unknown>).message === 'Invalid JWT' &&
        (parsed as Record<string, unknown>).code === 401
      ) {
        return { ok: false as const, status: 401, body: { error: 'supabase_gateway_invalid_jwt' } }
      }
    }

    if (!res.ok) return { ok: false as const, status: res.status, body: parsed }
    return { ok: true as const, status: res.status, body: parsed }
  }

  const isInvalidJwtPayload = (payload: unknown) => {
    if (typeof payload === 'string') return payload.includes('Invalid JWT')
    if (!payload || typeof payload !== 'object') return false
    const obj = payload as Record<string, unknown>
    return obj.message === 'Invalid JWT' || obj.error === 'invalid_jwt'
  }

  const first = await callFetch(token)
  if (
```

```
    !first.ok &&
    first.status === 401 &&
    typeof first.body === 'object' &&
    first.body !== null &&
    (first.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
  ) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) {
      await supabase.auth.signOut().catch(() => undefined)
      return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    }
    const nextProject = checkJwtProject(nextToken, supabaseUrl)
    if (!nextProject.ok) {
      await supabase.auth.signOut().catch(() => undefined)
      return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: nextProject.iss, expected:
nextProject.expectedPrefix } }
    }
    return callFetch(nextToken)
  }

  if (!first.ok && first.status === 401 && isInvalidJwtPayload(first.body)) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    return callFetch(nextToken)
  }

  return first
}

export function DashboardPage() {
  const { appPrincipal, masterUsuario, masterUsuarioLoading } = useAuth()
  const location = useLocation()
  const navigate = useNavigate()
  const funcionario = appPrincipal?.kind === 'funcionario' ? appPrincipal.profile : null
  const isGerente = appPrincipal?.kind === 'funcionario' && appPrincipal.profile.permissao === 'admin'
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : isGerente ? masterUsuario : null
  const usuarioId = usuario?.id ?? null

  const canVerFinanceiro = appPrincipal?.kind === 'usuario' ? true : Boolean(funcionario?.pode_ver_financeiro)
  const canBloquearHorarios = appPrincipal?.kind === 'usuario' ? true : Boolean(funcionario?.pode_bloquear_horarios)
  const canConfirmarAgendamento = appPrincipal?.kind === 'usuario' ? true :
Boolean(funcionario?.pode_criar_agendamentos)
  const canCancelarAgendamento = appPrincipal?.kind === 'usuario' ? true :
Boolean(funcionario?.pode_cancelar_agendamentos)

  const canUseRecurringBlocks = useMemo(() => {
    const p = String(usuario?.plano ?? '').trim().toLowerCase()
    return p === 'pro' || p === 'team' || p === 'enterprise'
  }, [usuario?.plano])

  useEffect(() => {
    const search = location.search ?? ''
    if (!search) return
    const params = new URLSearchParams(search)
    const checkout = (params.get('checkout') ?? '').trim().toLowerCase()
    if (checkout === 'success' || checkout === 'cancel') {
      navigate(`/pagamento${search}`, { replace: true })
    }
  }, [location.search, navigate])

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Navegação do dia',
          body: 'Use as setas para avançar/voltar o dia e conferir a agenda.',
          target: 'nav' as const,
        },
        {
          title: 'Filtro por profissional',
```

```
        body: 'Filtre para ver apenas os agendamentos de um profissional específico.',
        target: 'filter' as const,
      },
      {
        title: 'Bloquear horário',
        body: 'Crie bloqueios gerais ou por profissional para impedir novos horários.',
        target: 'block' as const,
      },
      {
        title: 'Agenda do dia',
        body: 'Aqui aparecem os horários livres, bloqueados e agendamentos.',
        target: 'agenda' as const,
      },
      {
        title: 'Resumo',
        body: 'Veja quantidade de agendamentos e total do dia em um resumo rápido.',
        target: 'summary' as const,
      },
    ] as const,
  []
)

const [date, setDate] = useState(() => {
  const now = new Date()
  now.setHours(0, 0, 0, 0)
  return now
})

const [viewMode, setViewMode] = useState<'dia' | 'semana'>('dia')

const [funcionarios, setFuncionarios] = useState<Funcionario[]>([])
const [filterFuncionarioId, setFilterFuncionarioId] = useState<string>('')
const [servicos, setServicos] = useState<ServicoOption[]>([])

const [search, setSearch] = useState('')
const [statusFilter, setStatusFilter] = useState('')
const [servicoFilterId, setServicoFilterId] = useState('')

const [agendamentos, setAgendamentos] = useState<Agendamento[]>([])
const [bloqueios, setBloqueios] = useState<Bloqueio[]>([])
const [loading, setLoading] = useState(true)
const [error, setError] = useState<string | null>(null)
const [success, setSuccess] = useState<string | null>(null)

const [blockStart, setBlockStart] = useState('')
const [blockEnd, setBlockEnd] = useState('')
const [blockMotivo, setBlockMotivo] = useState('')
const [blockFuncionarioId, setBlockFuncionarioId] = useState<string>('')
const [blockRepeat, setBlockRepeat] = useState<RepeatKind>('none')
const [blockRepeatUntil, setBlockRepeatUntil] = useState('')
const [savingBlock, setSavingBlock] = useState(false)

const weekStartDate = useMemo(() => startOfWeek(date), [date])
const weekEndDate = useMemo(() => addDays(weekStartDate, 6), [weekStartDate])
const dayKey = useMemo(() => toISODate(date), [date])
const weekStartKey = useMemo(() => toISODate(weekStartDate), [weekStartDate])
const weekEndKey = useMemo(() => toISODate(weekEndDate), [weekEndDate])

const setDateFromIso = (iso: string) => {
  const parts = iso.split('-')
  if (parts.length !== 3) return
  const y = Number(parts[0])
  const m = Number(parts[1])
  const d = Number(parts[2])
  if (!Number.isFinite(y) || !Number.isFinite(m) || !Number.isFinite(d)) return
  const next = new Date(y, m - 1, d)
  if (!Number.isFinite(next.getTime())) return
  next.setHours(0, 0, 0, 0)
  setDate(next)
}

const rangeLabel = useMemo(() => {
```

```
    if (viewMode === 'dia') {
      return date.toLocaleDateString('pt-BR', { day: '2-digit', month: 'short', year: 'numeric' })
    }
    const startLabel = weekStartDate.toLocaleDateString('pt-BR', { day: '2-digit', month: 'short' })
    const endLabel = weekEndDate.toLocaleDateString('pt-BR', { day: '2-digit', month: 'short', year: 'numeric' })
    return `${startLabel} – ${endLabel}`
  }, [date, viewMode, weekEndDate, weekStartDate])

  const searchNorm = useMemo(() => normalizeText(search.trim()), [search])
  const matchesFilters = useMemo(() => {
    const statusNorm = statusFilter.trim().toLowerCase()
    const servicoId = servicoFilterId.trim()
    return (a: Agendamento) => {
      const aStatus = (a.status ?? '').trim().toLowerCase()
      if (!statusNorm && aStatus === 'cancelado') return false
      if (statusNorm && aStatus !== statusNorm) return false
      if (servicoId && a.servico?.id !== servicoId) return false
      if (!searchNorm) return true
      const hay = normalizeText([a.cliente_nome, a.cliente_telefone, a.servico?.nome ?? '',
a.hora_inicio].filter(Boolean).join(' '))
      return hay.includes(searchNorm)
    }
  }, [searchNorm, servicoFilterId, statusFilter])

  const hasAnyFilter = useMemo(() => Boolean(statusFilter.trim() || servicoFilterId.trim() || searchNorm.trim()),
[searchNorm, servicoFilterId, statusFilter])

  const slotStepMinutes = 30

  const slots = useMemo(() => {
    if (!usuario?.horario_inicio || !usuario?.horario_fim) return []
    return buildSlots(usuario.horario_inicio, usuario.horario_fim, usuario.intervalo_inicio, usuario.intervalo_fim,
slotStepMinutes)
  }, [slotStepMinutes, usuario])

  const totalDia = useMemo(() => {
    if (!canVerFinanceiro) return 0
    return agendamentos
      .filter((a) => a.status !== 'cancelado')
      .reduce((sum, a) => sum + (a.servico?.preco ? Number(a.servico.preco) : 0), 0)
  }, [agendamentos, canVerFinanceiro])

  useEffect(() => {
    const run = async () => {
      if (!usuarioId) return
      const { data: servicosData, error: servicosError } = await supabase
        .from('servicos')
        .select('id,nome,cor')
        .eq('usuario_id', usuarioId)
        .eq('ativo', true)
        .order('ordem', { ascending: true })
        .order('criado_em', { ascending: true })
      if (!servicosError) {
        setServicos((servicosData ?? []) as unknown as ServicoOption[])
      }

      setLoading(true)
      setError(null)

      const { data: funcionariosData, error: funcionariosError } = await supabase
        .from('funcionarios')
        .select('id,nome_completo,ativo')
        .eq('usuario_master_id', usuarioId)
        .eq('permissao', 'funcionario')
        .order('criado_em', { ascending: true })
      if (funcionariosError) {
        setError(funcionariosError.message)
        setLoading(false)
        return
      }
      setFuncionarios(funcionariosData ?? [])
```

```
      const base = supabase
        .from('agendamentos')

.select('id,cliente_nome,cliente_telefone,data,hora_inicio,hora_fim,status,funcionario_id,extras,servico:servico_id(id,n
ome,preco,duracao_minutos,cor)')
        .eq('usuario_id', usuarioId)
        .order('data', { ascending: true })
        .order('hora_inicio', { ascending: true })

      const baseWithRange =
        viewMode === 'dia'
          ? base.eq('data', dayKey)
          : base.gte('data', weekStartKey).lte('data', weekEndKey)

      const { data: agData, error: agError } = filterFuncionarioId
        ? await baseWithRange.eq('funcionario_id', filterFuncionarioId)
        : await baseWithRange
      if (agError) {
        setError(agError.message)
        setLoading(false)
        return
      }
      const normalizedAgs = (agData ?? []).map((row) => {
        const r = row as unknown as Record<string, unknown>
        const horaInicio = normalizeTimeHHMM(String(r.hora_inicio ?? ''))
        const horaFimRaw = r.hora_fim
        const horaFim = horaFimRaw ? normalizeTimeHHMM(String(horaFimRaw)) : null
        return { ...r, hora_inicio: horaInicio, hora_fim: horaFim } as unknown as Agendamento
      })
      setAgendamentos(normalizedAgs)

      let bloqueiosQuery = supabase
        .from('bloqueios')
        .select('id,data,hora_inicio,hora_fim,motivo,funcionario_id')
        .eq('usuario_id', usuarioId)

      bloqueiosQuery =
        viewMode === 'dia'
          ? bloqueiosQuery.eq('data', dayKey)
          : bloqueiosQuery.gte('data', weekStartKey).lte('data', weekEndKey)
      if (filterFuncionarioId) {
        bloqueiosQuery = bloqueiosQuery.or(`funcionario_id.is.null,funcionario_id.eq.${filterFuncionarioId}`)
      }

      const { data: bloqueiosData, error: bloqueiosError } = await bloqueiosQuery
      if (bloqueiosError) {
        setError(bloqueiosError.message)
        setLoading(false)
        return
      }
      const normalizedBlocks = (bloqueiosData ?? []).map((row) => {
        const r = row as unknown as Record<string, unknown>
        const horaInicio = normalizeTimeHHMM(String(r.hora_inicio ?? ''))
        const horaFim = normalizeTimeHHMM(String(r.hora_fim ?? ''))
        return { ...r, hora_inicio: horaInicio, hora_fim: horaFim } as unknown as Bloqueio
      })
      setBloqueios(normalizedBlocks)
      setLoading(false)
    }
    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar')
      setLoading(false)
    })
  }, [usuarioId, dayKey, filterFuncionarioId, viewMode, weekEndKey, weekStartKey])

  const prevPeriod = () => {
    if (viewMode === 'semana') {
      setDate((prev) => addDays(prev, -7))
      return
    }
    setDate((prev) => addDays(prev, -1))
  }
```

```
  const nextPeriod = () => {
    if (viewMode === 'semana') {
      setDate((prev) => addDays(prev, 7))
      return
    }
    setDate((prev) => addDays(prev, 1))
  }

  const findAgendamentoAt = (time: string) => {
    const t = parseTimeToMinutes(time)
    const overlap = (a: Agendamento) => {
      const start = parseTimeToMinutes(a.hora_inicio)
      if (!Number.isFinite(start)) return false
      const end = (() => {
        if (a.hora_fim) {
          const v = parseTimeToMinutes(a.hora_fim)
          if (Number.isFinite(v)) return v
        }
        const dur = a.servico?.duracao_minutos != null ? Number(a.servico.duracao_minutos) : 0
        if (Number.isFinite(dur) && dur > 0) return start + dur
        return start + 1
      })()
      return t >= start && t < end
    }

    const candidates = agendamentos.filter(overlap)
    if (candidates.length === 0) return null

    if (hasAnyFilter) {
      const filtered = candidates.filter(matchesFilters)
      return filtered[0] ?? null
    }

    const active = candidates.filter((a) => (a.status ?? '').trim().toLowerCase() !== 'cancelado')
    return active[0] ?? null
  }

  const findAgendamentoStartInSlot = (time: string) => {
    const t = parseTimeToMinutes(time)
    if (!Number.isFinite(t)) return null
    const end = t + slotStepMinutes

    const candidates = agendamentos
      .filter((a) => {
        const start = parseTimeToMinutes(a.hora_inicio)
        return Number.isFinite(start) && start >= t && start < end
      })
      .slice()
      .sort((x, y) => parseTimeToMinutes(x.hora_inicio) - parseTimeToMinutes(y.hora_inicio))

    if (candidates.length === 0) return null

    if (hasAnyFilter) {
      const filtered = candidates.filter(matchesFilters)
      return filtered[0] ?? null
    }

    const active = candidates.filter((a) => (a.status ?? '').trim().toLowerCase() !== 'cancelado')
    return active[0] ?? candidates[0] ?? null
  }

  const findBloqueioAt = (time: string) => {
    const t = parseTimeToMinutes(time)
    return bloqueios.find((b) => {
      const s = parseTimeToMinutes(b.hora_inicio)
      const e = parseTimeToMinutes(b.hora_fim)
      return t >= s && t < e
    })
  }

  const findBloqueioStartInSlot = (time: string) => {
```

```
    const t = parseTimeToMinutes(time)
    if (!Number.isFinite(t)) return null
    const end = t + slotStepMinutes

    const candidates = bloqueios
      .filter((b) => {
        const start = parseTimeToMinutes(b.hora_inicio)
        return Number.isFinite(start) && start >= t && start < end
      })
      .slice()
      .sort((x, y) => parseTimeToMinutes(x.hora_inicio) - parseTimeToMinutes(y.hora_inicio))

    return candidates[0] ?? null
  }

  const effectiveBlockRepeat = canUseRecurringBlocks ? blockRepeat : 'none'
  const effectiveBlockRepeatUntil = canUseRecurringBlocks ? blockRepeatUntil : ''

  const repeatPreview = useMemo(() => {
    return buildRepeatKeys(dayKey, effectiveBlockRepeat, effectiveBlockRepeatUntil.trim() ? effectiveBlockRepeatUntil :
null)
  }, [dayKey, effectiveBlockRepeat, effectiveBlockRepeatUntil])

  const canSaveBlock = useMemo(() => {
    if (!usuarioId) return false
    if (!blockStart || !blockEnd) return false
    const s = parseTimeToMinutes(blockStart)
    const e = parseTimeToMinutes(blockEnd)
    if (!Number.isFinite(s) || !Number.isFinite(e)) return false
    if (e <= s) return false
    if (repeatPreview.error) return false
    if (repeatPreview.keys.length === 0) return false
    return true
  }, [usuarioId, blockStart, blockEnd, repeatPreview.error, repeatPreview.keys.length])

  const resolveFuncionarioLabel = (id: string | null) => {
    if (!id) return 'Geral'
    return funcionarios.find((f) => f.id === id)?.nome_completo ?? 'Profissional'
  }

  const weekDays = useMemo(() => {
    if (viewMode !== 'semana') return [] as Array<{ date: Date; key: string }>
    return Array.from({ length: 7 }).map((_, i) => {
      const d = addDays(weekStartDate, i)
      return { date: d, key: toISODate(d) }
    })
  }, [viewMode, weekStartDate])

  const agendamentosByDay = useMemo(() => {
    const map: Record<string, Agendamento[]> = {}
    for (const a of agendamentos) {
      const key = a.data
      if (!map[key]) map[key] = []
      map[key].push(a)
    }
    for (const key of Object.keys(map)) {
      map[key].sort((x, y) => parseTimeToMinutes(x.hora_inicio) - parseTimeToMinutes(y.hora_inicio))
    }
    return map
  }, [agendamentos])

  const bloqueiosByDay = useMemo(() => {
    const map: Record<string, Bloqueio[]> = {}
    for (const b of bloqueios) {
      const key = b.data
      if (!map[key]) map[key] = []
      map[key].push(b)
    }
    for (const key of Object.keys(map)) {
      map[key].sort((x, y) => parseTimeToMinutes(x.hora_inicio) - parseTimeToMinutes(y.hora_inicio))
    }
    return map
```

```
  }, [bloqueios])

  const createBloqueio = async () => {
    if (!usuarioId || !canSaveBlock || !canBloquearHorarios) return
    setSavingBlock(true)
    setError(null)
    const funcionarioId = blockFuncionarioId.trim() ? blockFuncionarioId : null
    const motivo = blockMotivo.trim() ? blockMotivo.trim() : null

    const { keys, error: repeatErr } = buildRepeatKeys(
      dayKey,
      effectiveBlockRepeat,
      effectiveBlockRepeatUntil.trim() ? effectiveBlockRepeatUntil : null
    )
    if (repeatErr) {
      setError(repeatErr)
      setSavingBlock(false)
      return
    }

    const s = parseTimeToMinutes(blockStart)
    const e = parseTimeToMinutes(blockEnd)
    type OcupacaoRow = { start_min: number; end_min: number }
    for (const k of keys) {
      const { data: ocupacoesData, error: ocupacoesErr } = await supabase.rpc('public_get_ocupacoes', {
        p_usuario_id: usuarioId,
        p_data: k,
        p_funcionario_id: funcionarioId,
      })
      if (ocupacoesErr) {
        setError(ocupacoesErr.message)
        setSavingBlock(false)
        return
      }
      const occupied = ((ocupacoesData ?? []) as unknown as OcupacaoRow[]).some((r) => overlaps(s, e, r.start_min,
r.end_min))
      if (occupied) {
        setError(`Conflito de horário em ${new Date(k).toLocaleDateString('pt-BR')}.`)
        setSavingBlock(false)
        return
      }
    }

    const rows = keys.map((k) => ({
      usuario_id: usuarioId,
      data: k,
      hora_inicio: blockStart,
      hora_fim: blockEnd,
      motivo,
      funcionario_id: funcionarioId,
    }))

    const { data: createdRows, error: err } = await supabase
      .from('bloqueios')
      .insert(rows)
      .select('id,data,hora_inicio,hora_fim,motivo,funcionario_id')

    if (err) {
      setError(err.message)
      setSavingBlock(false)
      return
    }

    const inserted = (createdRows ?? []) as unknown as Bloqueio[]
    const inRange = (b: Bloqueio) =>
      viewMode === 'dia' ? b.data === dayKey : b.data >= weekStartKey && b.data <= weekEndKey

    setBloqueios((prev) => [...prev, ...inserted.filter(inRange)])
    setBlockMotivo('')
    setBlockRepeat('none')
    setBlockRepeatUntil('')
    setSavingBlock(false)
```

```
  }

  const removeBloqueio = async (b: Bloqueio) => {
    if (!usuarioId || !canBloquearHorarios) return
    setError(null)
    const ok = window.confirm('Remover este bloqueio?')
    if (!ok) return
    const { error: err } = await supabase.from('bloqueios').delete().eq('id', b.id).eq('usuario_id', usuarioId)
    if (err) {
      setError(err.message)
      return
    }
    setBloqueios((prev) => prev.filter((x) => x.id !== b.id))
  }

  if (!usuario) {
    return (
      <AppShell>
        <div className="text-slate-700">{isGerente && masterUsuarioLoading ? 'Carregando…' : 'Acesso restrito.'}</div>
      </AppShell>
    )
  }

  if (funcionario && !funcionario.pode_ver_agenda) {
    return (
      <AppShell>
        <div className="text-slate-700">Acesso restrito.</div>
      </AppShell>
    )
  }

  return (
    <PageTutorial usuarioId={usuarioId} page="dashboard">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="space-y-6">
            <Card>
              <div className="p-6">
                <div
                  className={[
                    'flex flex-col gap-4 sm:flex-row sm:items-center sm:justify-between',
                    tutorialOpen && tutorialSteps[tutorialStep]?.target === 'nav'
                      ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl p-2 -m-2'
                      : '',
                  ].join(' ')}
                >
                  <div className="flex flex-wrap items-center gap-3">
                    <Button variant="secondary" onClick={prevPeriod}>
                      &lt;
                    </Button>
                    <div className="text-sm font-semibold text-slate-900">{rangeLabel}</div>
                    <Button variant="secondary" onClick={nextPeriod}>
                      &gt;
                    </Button>

                    <div className="flex items-center gap-1 rounded-lg bg-slate-100 p-1">
                      <button
                        type="button"
                        onClick={() => setViewMode('dia')}
                        className={[
                          'rounded-md px-3 py-1 text-sm font-medium',
                          viewMode === 'dia' ? 'bg-white text-slate-900 shadow-sm' : 'text-slate-700 hover:text-slate-
900',
                        ].join(' ')}
                      >
                        Dia
                      </button>
                      <button
                        type="button"
                        onClick={() => setViewMode('semana')}
                        className={[
                          'rounded-md px-3 py-1 text-sm font-medium',
```

```
                        viewMode === 'semana' ? 'bg-white text-slate-900 shadow-sm' : 'text-slate-700 hover:text-
slate-900',
                      ].join(' ')}
                    >
                      Semana
                    </button>
                  </div>

                  <Button variant="secondary" onClick={resetTutorial}>
                    Rever tutorial
                  </Button>
                </div>

                <div
                  className={[
                    'flex flex-wrap items-center gap-2',
                    tutorialOpen && tutorialSteps[tutorialStep]?.target === 'filter'
                      ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl p-2'
                      : '',
                  ].join(' ')}
                >
              <div className="text-sm text-slate-600">Filtrar por:</div>
              <button
                type="button"
                onClick={() => {
                  setFilterFuncionarioId('')
                  setBlockFuncionarioId('')
                }}
                className={[
                  'rounded-lg px-3 py-1.5 text-sm font-medium',
                  !filterFuncionarioId ? 'bg-slate-900 text-white' : 'bg-slate-100 text-slate-700 hover:bg-slate-200',
                ].join(' ')}
              >
                Todos
              </button>
              {funcionarios
                .filter((f) => f.ativo)
                .map((f) => (
                  <button
                    key={f.id}
                    type="button"
                    onClick={() => {
                      setFilterFuncionarioId(f.id)
                      setBlockFuncionarioId(f.id)
                    }}
                    className={[
                      'rounded-lg px-3 py-1.5 text-sm font-medium',
                      filterFuncionarioId === f.id
                        ? 'bg-slate-900 text-white'
                        : 'bg-slate-100 text-slate-700 hover:bg-slate-200',
                    ].join(' ')}
                  >
                    {f.nome_completo.split(' ')[0]}
                  </button>
                ))}
            </div>
          </div>

          <div className="mt-4 grid grid-cols-1 gap-3 sm:grid-cols-12">
            <div className="sm:col-span-3">
              <Input
                label="Data"
                type="date"
                value={dayKey}
                onChange={(e) => {
                  const v = e.target.value
                  if (!v.trim()) return
                  setDateFromIso(v.trim())
                }}
              />
            </div>
            <div className="sm:col-span-4">
```

```
              <Input label="Buscar" value={search} onChange={(e) => setSearch(e.target.value)} placeholder="Nome,
telefone, serviço…" />
            </div>
            <label className="block sm:col-span-2">
              <div className="text-sm font-medium text-slate-700 mb-1">Status</div>
              <select
                className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                value={statusFilter}
                onChange={(e) => setStatusFilter(e.target.value)}
              >
                <option value="">Todos</option>
                <option value="pendente">Pendente</option>
                <option value="confirmado">Confirmado</option>
                <option value="concluido">Concluído</option>
                <option value="nao_compareceu">No-show</option>
                <option value="cancelado">Cancelado</option>
              </select>
            </label>
            <label className="block sm:col-span-3">
              <div className="text-sm font-medium text-slate-700 mb-1">Serviço</div>
              <select
                className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                value={servicoFilterId}
                onChange={(e) => setServicoFilterId(e.target.value)}
              >
                <option value="">Todos</option>
                {servicos.map((s) => (
                  <option key={s.id} value={s.id}>
                    {s.nome}
                  </option>
                ))}
              </select>
            </label>
          </div>
        </div>
      </Card>

      {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> :
null}
      {success ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">
{success}</div> : null}

      <div
        className={
          tutorialOpen && tutorialSteps[tutorialStep]?.target === 'block'
            ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
            : ''
        }
      >
        <Card>
          {canBloquearHorarios ? (
          <div className="p-6 space-y-4">
          <div className="flex items-center justify-between gap-3">
            <div>
              <div className="text-sm font-semibold text-slate-900">Bloquear horário</div>
              <div className="text-xs text-slate-600">Bloqueio geral ou por profissional.</div>
            </div>
          </div>

          <div className="grid grid-cols-1 gap-4 sm:grid-cols-4">
            <Input label="Início" type="time" value={blockStart} onChange={(e) => setBlockStart(e.target.value)} step=
{900} />
            <Input label="Fim" type="time" value={blockEnd} onChange={(e) => setBlockEnd(e.target.value)} step={900}
/>
            <label className="block">
              <div className="text-sm font-medium text-slate-700 mb-1">Profissional</div>
              <select
                className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                value={blockFuncionarioId}
```

```
         onChange={(e) => setBlockFuncionarioId(e.target.value)}
        >
         <option value="">Geral</option>
         {funcionarios
          .filter((f) => f.ativo)
          .map((f) => (
           <option key={f.id} value={f.id}>
            {f.nome_completo}
           </option>
          ))}
        </select>
       </label>
       <Input label="Motivo (opcional)" value={blockMotivo} onChange={(e) => setBlockMotivo(e.target.value)} />
      </div>

      <div className="grid grid-cols-1 gap-4 sm:grid-cols-4">
       <label className="block">
        <div className="text-sm font-medium text-slate-700 mb-1">Repetir</div>
        <select
         className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
         value={blockRepeat}
         disabled={!canUseRecurringBlocks}
         onChange={(e) => {
          if (!canUseRecurringBlocks) return
          const next = e.target.value as RepeatKind
          setBlockRepeat(next)
          if (next === 'none') {
           setBlockRepeatUntil('')
           return
          }
          if (!blockRepeatUntil.trim()) {
           setBlockRepeatUntil(toISODate(addDays(date, next === 'monthly' ? 180 : 30)))
          }
         }}
        >
         <option value="none">Não repetir</option>
         {canUseRecurringBlocks ? <option value="daily">Diariamente</option> : null}
         {canUseRecurringBlocks ? <option value="weekly">Semanalmente</option> : null}
         {canUseRecurringBlocks ? <option value="monthly">Mensalmente</option> : null}
        </select>
       </label>
       <div className="sm:col-span-3">
        {blockRepeat !== 'none' ? (
         <div className="grid grid-cols-1 gap-4 sm:grid-cols-3">
          <Input
           label="Até"
           type="date"
           min={dayKey}
           value={blockRepeatUntil}
           onChange={(e) => setBlockRepeatUntil(e.target.value)}
          />
          <div className="sm:col-span-2 flex items-end">
           <div className="text-xs text-slate-600">
            {repeatPreview.error ? repeatPreview.error : `Serão criados ${repeatPreview.keys.length}
bloqueios.`}
           </div>
          </div>
         </div>
        ) : null}
       </div>
      </div>

      <div className="flex justify-end">
       <Button onClick={createBloqueio} disabled={!canSaveBlock || savingBlock}>
        Salvar bloqueio
       </Button>
      </div>

      {bloqueios.length > 0 ? (
       <div className="rounded-xl border border-slate-200 divide-y divide-slate-100">
        {bloqueios
```

```
                  .slice()
                  .sort((a, b) => parseTimeToMinutes(a.hora_inicio) - parseTimeToMinutes(b.hora_inicio))
                  .map((b) => (
                    <div key={b.id} className="p-3 flex items-center justify-between gap-3">
                      <div className="text-sm text-slate-700">
                        <span className="font-semibold text-slate-900">{b.hora_inicio}</span>
                        {' – '}
                        <span className="font-semibold text-slate-900">{b.hora_fim}</span>
                        {' • '}
                        <span className="text-slate-700">{resolveFuncionarioLabel(b.funcionario_id)}</span>
                        {b.motivo ? <span className="text-slate-500"> {' • '}{b.motivo}</span> : null}
                      </div>
                      {canBloquearHorarios ? (
                        <Button variant="danger" onClick={() => removeBloqueio(b)}>
                          Remover
                        </Button>
                      ) : null}
                    </div>
                  ))}
              </div>
            ) : null}
            </div>
            ) : (
              <div className="p-6 text-sm text-slate-700">Acesso restrito.</div>
            )}
          </Card>
        </div>

        <div
          className={
            tutorialOpen && tutorialSteps[tutorialStep]?.target === 'agenda'
              ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
              : ''
          }
        >
          <Card>
            <div className="divide-y divide-slate-100">
            {loading ? (
              <div className="p-6 text-sm text-slate-600">Carregando agenda…</div>
            ) : slots.length === 0 ? (
              <div className="p-6 space-y-3">
                <div className="text-sm text-slate-600">Configure seu horário em /onboarding.</div>
                {agendamentos.length > 0 ? (
                  <div className="rounded-xl border border-slate-200 divide-y divide-slate-100">
                    {agendamentos.map((ag) => {
                      const statusUi = resolveStatusUi(ag.status)
                      const endereco = readExtrasEndereco(ag.extras)
                      return (
                        <div key={ag.id} className="p-3 flex items-start justify-between gap-3">
                          <div className="min-w-0">
                            <div className="text-sm font-semibold text-slate-900 truncate">
                              {(normalizeTimeHHMM(ag.hora_inicio) || '—')} — {ag.cliente_nome}
                            </div>
                            <div className="text-sm text-slate-600">📱 {ag.cliente_telefone}</div>
                            {endereco ? <div className="text-sm text-slate-600">📍 {endereco}</div> : null}
                            <div className="text-sm text-slate-700">
                              ✂️ {ag.servico?.nome} {canVerFinanceiro && ag.servico?.preco ? `- ${formatBRMoney(Number(ag.servico.preco))}` : ''}
                            </div>
                          </div>
                          <Badge tone={statusUi.tone}>{statusUi.label}</Badge>
                        </div>
                      )
                    })}
                  </div>
                ) : null}
              </div>
            ) : viewMode === 'dia' ? (
              slots.map((time) => {
                const agStart = findAgendamentoStartInSlot(time)
                const agCover = agStart ?? findAgendamentoAt(time)
                const blockStart = findBloqueioStartInSlot(time)
```

```
                const blockCover = blockStart ?? findBloqueioAt(time)

                if (agCover) {
                  const isStart = Boolean(agStart)
                  const visible = isStart && matchesFilters(agCover)
                  const statusUi = resolveStatusUi(agCover.status)
                  const startLabel = normalizeTimeHHMM(agCover.hora_inicio)
                  const timeLabel = isStart && startLabel ? startLabel : time
                  const endereco = visible ? readExtrasEndereco(agCover.extras) : null
                  return (
                    <div key={time} className="p-4 flex items-start justify-between gap-3">
                      <div>
                        <div className="text-sm font-semibold text-slate-900 flex items-center gap-2">
                          {agCover.servico?.cor ? (
                            <span className="h-2.5 w-2.5 rounded-full" style={{ backgroundColor: agCover.servico.cor }}
/>
                          ) : null}
                          <span>
                            {timeLabel} - {visible ? agCover.cliente_nome : 'Ocupado'}
                          </span>
                        </div>
                        {visible ? <div className="text-sm text-slate-600">📱 {agCover.cliente_telefone}</div> : null}
                        {endereco ? <div className="text-sm text-slate-600">📍 {endereco}</div> : null}
                        {visible ? (
                          <div className="text-sm text-slate-700">
                            ✂️ {agCover.servico?.nome} {canVerFinanceiro && agCover.servico?.preco ? `-
${formatBRMoney(Number(agCover.servico.preco))}` : ''}
                          </div>
                        ) : null}
                      </div>
                      <div className="flex flex-col items-end gap-2">
                        <Badge tone={statusUi.tone}>{statusUi.label}</Badge>
                        {agCover.status !== 'cancelado' && visible ? (
                          <div className="flex gap-2">
                            {canConfirmarAgendamento && String(agCover.status ?? '').trim().toLowerCase() !==
'confirmado' ? (
                              <Button
                                variant="secondary"
                                onClick={async () => {
                                  if (!canConfirmarAgendamento) return
                                setError(null)
                                const { error: updErr } = await supabase.from('agendamentos').update({ status:
'confirmado' }).eq('id', agCover.id)
                                if (updErr) {
                                  const formatted = formatSupabaseError(updErr)
                                  const lower = formatted.toLowerCase()
                                  if (lower.includes('internal server error') || lower.includes('500')) {
                                    setError(
                                      `${formatted} • Provável trigger/função no Postgres falhando. Reexecute no
Supabase o “SQL do WhatsApp (trigger confirmação imediata)” e o “SQL de Logs de Auditoria”.`
                                    )
                                    return
                                  }
                                  setError(formatted)
                                  return
                                }
                                setAgendamentos((prev) => prev.map((x) => (x.id === agCover.id ? { ...x, status:
'confirmado' } : x)))
                                const sendRes = await sendConfirmacaoWhatsapp(agCover.id)
                                if (!sendRes.ok) {
                                  if (typeof sendRes.body === 'object' && sendRes.body !== null && (sendRes.body as
Record<string, unknown>).error === 'supabase_gateway_invalid_jwt') {
                                    setError('JWT inválido para chamar a Edge Function. Saia e entre novamente. Se
persistir, reimplante a função com verify_jwt=false (--no-verify-jwt).')
                                    return
                                  }
                                  if (typeof sendRes.body === 'object' && sendRes.body !== null && (sendRes.body as
Record<string, unknown>).error === 'jwt_project_mismatch') {
                                    setError('Sessão do Supabase pertence a outro projeto. Saia e entre novamente no
sistema.')
                                    return
                                  }
```

```
                                  if (typeof sendRes.body === 'object' && sendRes.body !== null && (sendRes.body as Record<string, unknown>).error === 'invalid_jwt') {
                                    setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
                                    return
                                  }
                                  if (typeof sendRes.body === 'object' && sendRes.body !== null) {
                                    const hint = (sendRes.body as Record<string, unknown>).hint
                                    if (typeof hint === 'string' && hint.trim()) {
                                      setError(hint)
                                      return
                                    }
                                    const code = (sendRes.body as Record<string, unknown>).error
                                    if (code === 'instance_not_connected') {
                                      setError('WhatsApp não conectado. Vá em Configurações > WhatsApp e conecte a instância (QR Code).')

                                      return
                                    }
                                  }
                                  const details = typeof sendRes.body === 'string' ? sendRes.body : JSON.stringify(sendRes.body)
                                  setError(`Falha ao enviar confirmação (HTTP ${sendRes.status}): ${details}`)
                                }
                              }}
                              >
                                ✓ Confirmar
                              </Button>
                            ) : null}
                            {canCancelarAgendamento ? (
                              <Button
                                variant="secondary"
                                onClick={async () => {
                                  if (!canCancelarAgendamento) return
                                setError(null)
                                const ok = window.confirm('Marcar como no-show?')
                                if (!ok) return
                                const { error: updErr } = await supabase
                                  .from('agendamentos')
                                  .update({ status: 'nao_compareceu' })
                                  .eq('id', agCover.id)
                                if (updErr) {
                                  setError(formatSupabaseError(updErr))
                                  return
                                }
                                setAgendamentos((prev) => prev.map((x) => (x.id === agCover.id ? { ...x, status: 'nao_compareceu' } : x)))
                              }}
                              >
                                No-show
                              </Button>
                            ) : null}
                            {canCancelarAgendamento ? (
                              <Button
                                variant="danger"
                                onClick={async () => {
                                  if (!canCancelarAgendamento) return
                                setError(null)
                                setSuccess(null)
                                const { error: updErr } = await supabase
                                  .from('agendamentos')
                                  .update({ status: 'cancelado', cancelado_em: new Date().toISOString() })
                                  .eq('id', agCover.id)
                                if (updErr) {
                                  setError(formatSupabaseError(updErr))
                                  return
                                }
                                setAgendamentos((prev) => prev.map((x) => (x.id === agCover.id ? { ...x, status: 'cancelado' } : x)))

                                setSuccess('Agendamento cancelado.')
                                const sendRes = await sendCancelamentoWhatsapp(agCover.id)
                                if (sendRes.ok) {
                                  if (typeof sendRes.body === 'object' && sendRes.body !== null) {
                                    const skipped = (sendRes.body as Record<string, unknown>).skipped
```

```
                                    if (skipped === 'not_configured') {
                                      setSuccess('Agendamento cancelado. WhatsApp não configurado.')
                                      return
                                    }
                                    if (skipped === 'disabled') {
                                      setSuccess('Agendamento cancelado. Envio automático desabilitado.')
                                      return
                                    }
                                  }
                                  setSuccess('Agendamento cancelado e aviso enviado.')
                                  return
                                }
                                if (
                                  typeof sendRes.body === 'object' &&
                                  sendRes.body !== null &&
                                  (sendRes.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
                                ) {
                                  setError('JWT inválido para chamar a Edge Function. Saia e entre novamente. Se
persistir, reimplante a função com verify_jwt=false (--no-verify-jwt).')
                                  return
                                }
                                if (typeof sendRes.body === 'object' && sendRes.body !== null && (sendRes.body as
Record<string, unknown>).error === 'jwt_project_mismatch') {
                                  setError('Sessão do Supabase pertence a outro projeto. Saia e entre novamente no
sistema.')
                                  return
                                }
                                if (typeof sendRes.body === 'object' && sendRes.body !== null && (sendRes.body as
Record<string, unknown>).error === 'invalid_jwt') {
                                  setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
                                  return
                                }
                                if (typeof sendRes.body === 'object' && sendRes.body !== null) {
                                  const hint = (sendRes.body as Record<string, unknown>).hint
                                  if (typeof hint === 'string' && hint.trim()) {
                                    setError(hint)
                                    return
                                  }
                                  const code = (sendRes.body as Record<string, unknown>).error
                                  if (code === 'instance_not_connected') {
                                    setError('WhatsApp não conectado. Vá em Configurações > WhatsApp e conecte a
instância (QR Code).')
                                    return
                                  }
                                }
                                const details = typeof sendRes.body === 'string' ? sendRes.body :
JSON.stringify(sendRes.body)
                                setError(`Falha ao enviar cancelamento (HTTP ${sendRes.status}): ${details}`)
                              }}
                              >
                                ✗ Cancelar
                              </Button>
                            ) : null}
                          </div>
                        ) : null}
                      </div>
                    </div>
                  )
                }
                if (blockCover) {
                  const isStart = Boolean(blockStart)
                  const timeLabel = isStart ? blockCover.hora_inicio : time
                  return (
                    <div key={time} className="p-4 flex items-center justify-between">
                      <div className="text-sm font-semibold text-slate-900">
                        {timeLabel} - 🔒 BLOQUEADO {isStart && blockCover.motivo ? `(${blockCover.motivo})` : ''}
                      </div>
                      <div className="flex items-center gap-2">
                        {isStart && blockCover.funcionario_id ? <Badge>
{resolveFuncionarioLabel(blockCover.funcionario_id)}</Badge> : null}
                        <Badge tone="yellow">Bloqueado</Badge>
                      </div>
```

```
                    </div>
                  )
                }
                return (
                  <div key={time} className="p-4 flex items-center justify-between">
                    <div className="text-sm text-slate-600">{time} - LIVRE</div>
                  </div>
                )
              })
            ) : (
              <div className="p-4">
                <div className="overflow-x-auto">
                  <div className="grid grid-cols-7 gap-3 min-w-[980px]">
                    {weekDays.map((d) => {
                      const ags = agendamentosByDay[d.key] ?? []
                      const bls = bloqueiosByDay[d.key] ?? []
                      const visibleAgs = ags.filter(matchesFilters)
                      const visibleAgCount = visibleAgs.length
                      const label = d.date.toLocaleDateString('pt-BR', { weekday: 'short', day: '2-digit', month: '2-
digit' })
                      return (
                        <div key={d.key} className="rounded-xl border border-slate-200 bg-white overflow-hidden">
                          <div className="px-3 py-2 border-b border-slate-100 bg-slate-50">
                            <div className="text-sm font-semibold text-slate-900 flex items-center justify-between gap-
2">
                              <span>{label}</span>
                              <span className="text-xs text-slate-600">{visibleAgCount} ag.</span>
                            </div>
                          </div>
                          <div className="divide-y divide-slate-100">
                            {visibleAgs.length === 0 && bls.length === 0 ? (
                              <div className="px-3 py-3 space-y-2">
                                <div className="text-sm text-slate-600">Sem itens</div>
                                {hasAnyFilter ? (
                                  <div>
                                    <Button
                                      variant="secondary"
                                      onClick={() => {
                                        setSearch('')
                                        setStatusFilter('')
                                        setServicoFilterId('')
                                      }}
                                    >
                                      Limpar filtros
                                    </Button>
                                  </div>
                                ) : null}
                              </div>
                            ) : (
                              [...visibleAgs.map((a) => ({ kind: 'ag' as const, time: a.hora_inicio, ag: a })),
...bls.map((b) => ({ kind: 'b' as const, time: b.hora_inicio, b }))]
                                .sort((x, y) => parseTimeToMinutes(x.time) - parseTimeToMinutes(y.time))
                                .map((item) => {
                                  if (item.kind === 'b') {
                                    return (
                                      <div key={`b:${item.b.id}`} className="px-3 py-2 flex items-start justify-between
gap-2">
                                        <div className="text-sm text-slate-700">
                                          <div className="font-semibold text-slate-900">
                                            {item.b.hora_inicio}–{item.b.hora_fim}
                                          </div>
                                          <div className="text-slate-600">🔒 Bloqueado {item.b.motivo ? `•
${item.b.motivo}` : ''}</div>
                                        </div>
                                        <Badge tone="yellow">Bloqueado</Badge>
                                      </div>
                                    )
                                  }
                                  const a = item.ag
                                  const visible = true
                                  const statusUi = resolveStatusUi(a.status)
                                  return (
```

```
                                    <div key={`ag:${a.id}`} className="px-3 py-2 flex items-start justify-between gap-
2">
                                      <div className="text-sm text-slate-700">
                                        <div className="font-semibold text-slate-900 flex items-center gap-2">
                                          {a.servico?.cor ? (
                                            <span className="h-2.5 w-2.5 rounded-full" style={{ backgroundColor:
a.servico.cor }} />
                                          ) : null}
                                          <span>
                                            {a.hora_inicio} • {visible ? a.cliente_nome : 'Ocupado'}
                                          </span>
                                        </div>
                                        {visible ? <div className="text-slate-600">{a.servico?.nome ?? 'Serviço'}</div>
: null}
                                        {visible ? (() => {
                                          const endereco = readExtrasEndereco(a.extras)
                                          return endereco ? <div className="text-slate-600">📍 {endereco}</div> : null
                                        })() : null}
                                      </div>
                                      <Badge tone={statusUi.tone}>{statusUi.label}</Badge>
                                    </div>
                                  )
                                })
                            )}
                          </div>
                        </div>
                      )
                    })}
                  </div>
                </div>
              </div>
            )}
            </div>
          </Card>
        </div>

        <div
          className={
            tutorialOpen && tutorialSteps[tutorialStep]?.target === 'summary'
              ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
              : ''
          }
        >
          <Card>
            <div className="p-6 flex items-center justify-between">
            <div>
              <div className="text-sm font-semibold text-slate-900">Resumo do Dia</div>
              <div className="text-sm text-slate-600">
                {agendamentos.filter((a) => a.status !== 'cancelado').length} agendamentos
              </div>
            </div>
            <div className="text-lg font-semibold text-slate-900">{canVerFinanceiro ? formatBRMoney(totalDia) : '—'}
</div>
            </div>
          </Card>
        </div>
      </div>

          <TutorialOverlay open={tutorialOpen} steps={tutorialSteps} step={tutorialStep} onStepChange={setTutorialStep}
onClose={closeTutorial} />
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/FuncionarioAgendaPage.tsx

```
import { useEffect, useMemo, useRef, useState } from 'react'
import { AppShell } from '../../components/layout/AppShell'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { formatBRMoney, minutesToTime, normalizeTimeHHMM, parseTimeToMinutes, toISODate } from '../../lib/dates'
import { checkJwtProject, supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type Agendamento = {
  id: string
  cliente_nome: string
  cliente_telefone: string
  data: string
  hora_inicio: string
  hora_fim: string | null
  status: string
  funcionario_id: string | null
  extras: Record<string, unknown> | null
  servico: { id: string; nome: string; preco: number; duracao_minutos: number; cor: string | null } | null
}

function readExtrasEndereco(extras: unknown) {
  if (!extras || typeof extras !== 'object') return null
  const v = (extras as Record<string, unknown>).endereco
  if (typeof v !== 'string') return null
  const t = v.trim()
  return t ? t : null
}

type ServicoOption = { id: string; nome: string; cor: string | null }

type Bloqueio = { id: string; data: string; hora_inicio: string; hora_fim: string; motivo: string | null;
funcionario_id: string | null }

type FuncionarioOption = {
  id: string
  nome_completo: string
  ativo: boolean
  horario_inicio: string | null
  horario_fim: string | null
  intervalo_inicio: string | null
  intervalo_fim: string | null
}

function formatSupabaseError(err: { message: string; details?: string | null; hint?: string | null; code?: string | null
}) {
  const parts = [err.message]
  if (err.code) parts.push(`code=${err.code}`)
  if (err.details) parts.push(err.details)
  if (err.hint) parts.push(err.hint)
  return parts.filter((p) => typeof p === 'string' && p.trim()).join(' • ')
}

function buildSlots(
  start: string,
  end: string,
  intervalStart: string | null,
  intervalEnd: string | null,
  stepMinutes: number
): string[] {
  const startMin = parseTimeToMinutes(start)
  const endMin = parseTimeToMinutes(end)
  const ivStart = intervalStart ? parseTimeToMinutes(intervalStart) : null
  const ivEnd = intervalEnd ? parseTimeToMinutes(intervalEnd) : null
  const slots: string[] = []
  for (let m = startMin; m < endMin; m += stepMinutes) {
    if (ivStart !== null && ivEnd !== null && m >= ivStart && m < ivEnd) continue
```

```
    slots.push(minutesToTime(m))
  }
  return slots
}

function startOfWeek(value: Date) {
  const d = new Date(value)
  d.setHours(0, 0, 0, 0)
  const day = d.getDay()
  const diff = (day + 6) % 7
  d.setDate(d.getDate() - diff)
  return d
}

function addDays(value: Date, days: number) {
  const d = new Date(value)
  d.setDate(d.getDate() + days)
  return d
}

function addMonths(value: Date, months: number) {
  const d = new Date(value)
  const day = d.getDate()
  d.setDate(1)
  d.setMonth(d.getMonth() + months)
  const lastDay = new Date(d.getFullYear(), d.getMonth() + 1, 0).getDate()
  d.setDate(Math.min(day, lastDay))
  return d
}

function parseDateKey(value: string) {
  const [y, m, d] = value.split('-').map((v) => Number(v))
  const out = new Date(y, m - 1, d)
  out.setHours(0, 0, 0, 0)
  return out
}

function overlaps(startA: number, endA: number, startB: number, endB: number) {
  return startA < endB && endA > startB
}

type RepeatKind = 'none' | 'daily' | 'weekly' | 'monthly'

function buildRepeatKeys(startKey: string, repeat: RepeatKind, untilKey: string | null) {
  if (repeat === 'none') return { keys: [startKey], error: null as string | null }
  if (!untilKey) return { keys: [] as string[], error: 'Selecione a data final da recorrência.' }

  const start = parseDateKey(startKey)
  const until = parseDateKey(untilKey)
  if (until < start) return { keys: [] as string[], error: 'A data final deve ser igual ou posterior ao início.' }

  const keys: string[] = []
  const max = 90
  let cur = start
  while (cur <= until && keys.length < max) {
    keys.push(toISODate(cur))
    cur = repeat === 'daily' ? addDays(cur, 1) : repeat === 'weekly' ? addDays(cur, 7) : addMonths(cur, 1)
  }

  if (cur <= until) {
    return { keys: [] as string[], error: 'Intervalo muito grande. Reduza a data final da recorrência.' }
  }

  return { keys, error: null as string | null }
}

function normalizeText(value: string) {
  return value
    .toLowerCase()
    .normalize('NFD')
    .replace(/\p{Diacritic}/gu, '')
}
```

```
function resolveStatusUi(status: string | null | undefined): { label: string; tone: 'slate' | 'green' | 'yellow' | 'red'
} {
  const raw = String(status ?? '')
  const s = raw.trim().toLowerCase()
  if (!s) return { label: 'Pendente', tone: 'yellow' }
  if (s === 'confirmado') return { label: 'Confirmado', tone: 'green' }
  if (s === 'cancelado') return { label: 'Cancelado', tone: 'red' }
  if (s === 'pendente') return { label: 'Pendente', tone: 'yellow' }
  if (s === 'concluido' || s === 'concluído') return { label: 'Concluído', tone: 'green' }
  if (s === 'nao_compareceu' || s === 'não_compareceu' || s === 'no_show') return { label: 'No-show', tone: 'red' }
  return { label: raw || 'Status', tone: 'slate' }
}

async function sendConfirmacaoWhatsapp(agendamentoId: string) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } }
  }

  const supabaseUrl = supabaseEnv.values.VITE_SUPABASE_URL
  const supabaseAnonKey = supabaseEnv.values.VITE_SUPABASE_ANON_KEY

  const tryRefresh = async () => {
    const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
    if (refreshErr) return null
    return refreshed.session ?? null
  }

  const { data: sessionData } = await supabase.auth.getSession()
  let session = sessionData.session
  const now = Math.floor(Date.now() / 1000)
  const expiresAt = session?.expires_at ?? null

  if (session && expiresAt && expiresAt <= now + 60) {
    const refreshed = await tryRefresh()
    if (refreshed) session = refreshed
  }

  const token = session?.access_token ?? null
  if (!token) {
    return { ok: false as const, status: 401, body: { error: 'session_expired' } }
  }

  const tokenProject = checkJwtProject(token, supabaseUrl)
  if (!tokenProject.ok) {
    await supabase.auth.signOut().catch(() => undefined)
    return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: tokenProject.iss, expected:
tokenProject.expectedPrefix } }
  }

  const callFetch = async (jwt: string) => {
    const fnUrl = `${supabaseUrl}/functions/v1/whatsapp`
    let res: Response
    try {
      res = await fetch(fnUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          apikey=[REDACTED]
          Authorization: `Bearer ${jwt}`,
          'x-user-jwt': jwt,
        },
        body: JSON.stringify({ action: 'send_confirmacao', agendamento_id: agendamentoId }),
      })
    } catch (e: unknown) {
      const msg = e instanceof Error ? e.message : 'Falha de rede'
      return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
    }

    const fnVersion = res.headers.get('x-smagenda-fn')

    const text = await res.text()
```

```
    let parsed: unknown = null
    try {
      parsed = text ? JSON.parse(text) : null
    } catch {
      parsed = text
    }

    if (!res.ok && res.status === 401 && !fnVersion) {
      if (
        parsed &&
        typeof parsed === 'object' &&
        (parsed as Record<string, unknown>).message === 'Invalid JWT' &&
        (parsed as Record<string, unknown>).code === 401
      ) {
        return { ok: false as const, status: 401, body: { error: 'supabase_gateway_invalid_jwt' } }
      }
    }

    if (!res.ok) return { ok: false as const, status: res.status, body: parsed }
    return { ok: true as const, status: res.status, body: parsed }
  }

  const isInvalidJwtPayload = (payload: unknown) => {
    if (typeof payload === 'string') return payload.includes('Invalid JWT')
    if (!payload || typeof payload !== 'object') return false
    const obj = payload as Record<string, unknown>
    return obj.message === 'Invalid JWT' || obj.error === 'invalid_jwt'
  }

  const first = await callFetch(token)
  if (
    !first.ok &&
    first.status === 401 &&
    typeof first.body === 'object' &&
    first.body !== null &&
    (first.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
  ) {
    return first
  }

  if (!first.ok && first.status === 401 && isInvalidJwtPayload(first.body)) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    return callFetch(nextToken)
  }

  return first
}

async function sendCancelamentoWhatsapp(agendamentoId: string) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } }
  }

  const supabaseUrl = supabaseEnv.values.VITE_SUPABASE_URL
  const supabaseAnonKey = supabaseEnv.values.VITE_SUPABASE_ANON_KEY

  const tryRefresh = async () => {
    const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
    if (refreshErr) return null
    return refreshed.session ?? null
  }

  const { data: sessionData } = await supabase.auth.getSession()
  let session = sessionData.session
  const now = Math.floor(Date.now() / 1000)
  const expiresAt = session?.expires_at ?? null

  if (session && expiresAt && expiresAt <= now + 60) {
    const refreshed = await tryRefresh()
    if (refreshed) session = refreshed
```

```
  }

  const token = session?.access_token ?? null
  if (!token) {
    return { ok: false as const, status: 401, body: { error: 'session_expired' } }
  }

  const tokenProject = checkJwtProject(token, supabaseUrl)
  if (!tokenProject.ok) {
    await supabase.auth.signOut().catch(() => undefined)
    return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: tokenProject.iss, expected:
tokenProject.expectedPrefix } }
  }

  const callFetch = async (jwt: string) => {
    const fnUrl = `${supabaseUrl}/functions/v1/whatsapp`
    let res: Response
    try {
      res = await fetch(fnUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          apikey=[REDACTED]
          Authorization: `Bearer ${jwt}`,
          'x-user-jwt': jwt,
        },
        body: JSON.stringify({ action: 'send_cancelamento', agendamento_id: agendamentoId }),
      })
    } catch (e: unknown) {
      const msg = e instanceof Error ? e.message : 'Falha de rede'
      return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
    }

    const fnVersion = res.headers.get('x-smagenda-fn')

    const text = await res.text()
    let parsed: unknown = null
    try {
      parsed = text ? JSON.parse(text) : null
    } catch {
      parsed = text
    }

    if (!res.ok && res.status === 401 && !fnVersion) {
      if (
        parsed &&
        typeof parsed === 'object' &&
        (parsed as Record<string, unknown>).message === 'Invalid JWT' &&
        (parsed as Record<string, unknown>).code === 401
      ) {
        return { ok: false as const, status: 401, body: { error: 'supabase_gateway_invalid_jwt' } }
      }
    }

    if (!res.ok) return { ok: false as const, status: res.status, body: parsed }
    return { ok: true as const, status: res.status, body: parsed }
  }

  const isInvalidJwtPayload = (payload: unknown) => {
    if (typeof payload === 'string') return payload.includes('Invalid JWT')
    if (!payload || typeof payload !== 'object') return false
    const obj = payload as Record<string, unknown>
    return obj.message === 'Invalid JWT' || obj.error === 'invalid_jwt'
  }

  const first = await callFetch(token)
  if (
    !first.ok &&
    first.status === 401 &&
    typeof first.body === 'object' &&
    first.body !== null &&
    (first.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
```

```
  ) {
    return first
  }

  if (!first.ok && first.status === 401 && isInvalidJwtPayload(first.body)) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    return callFetch(nextToken)
  }

  return first
}

export function FuncionarioAgendaPage() {
  const { appPrincipal, refresh } = useAuth()
  const funcionario = appPrincipal?.kind === 'funcionario' ? appPrincipal.profile : null
  const funcionarioId = funcionario?.id ? funcionario.id : null
  const usuarioMasterId = (() => {
    const raw = funcionario?.usuario_master_id
    const v = typeof raw === 'string' ? raw.trim() : ''
    return v ? v : null
  })()

  const formatQueryError = (err: { message: string; details?: string | null; hint?: string | null; code?: string | null
}) => {
    const formatted = formatSupabaseError(err)
    const lower = formatted.toLowerCase()
    const isRls = lower.includes('row-level security') || lower.includes('rls')
    const isDenied = lower.includes('permission denied') || lower.includes('not allowed') || err.code === '42501'
    if (isRls || isDenied) {
      return `${formatted} • Verifique se o atendente está com “Ver agenda” ativo e reexecute no Supabase o “SQL de
políticas (Usuário / Funcionário)”.`
    }
    return formatted
  }

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Navegação e filtros',
          body: 'Use dia/semana, setas e filtros para encontrar agendamentos rapidamente.',
          target: 'filtros' as const,
        },
        {
          title: 'Bloquear horário',
          body: 'Crie bloqueios para reservar horários na sua agenda (quando permitido).',
          target: 'bloqueio' as const,
        },
        {
          title: 'Sua agenda',
          body: 'Aqui aparecem seus horários e agendamentos. Você pode confirmar, marcar no-show ou cancelar (conforme
permissão).',
          target: 'agenda' as const,
        },
      ] as const,
    []
  )

  const [usuarioMasterHorario, setUsuarioMasterHorario] = useState<{
    horario_inicio: string | null
    horario_fim: string | null
    intervalo_inicio: string | null
    intervalo_fim: string | null
    dias_trabalho: number[] | null
  } | null>(null)

  const [usuarioMasterPlano, setUsuarioMasterPlano] = useState<string | null>(null)

  const [date, setDate] = useState(() => {
    const now = new Date()
```

```
    now.setHours(0, 0, 0, 0)
    return now
  })

  const [viewMode, setViewMode] = useState<'dia' | 'semana'>('dia')

  const agendaLayoutKey = useMemo(() => {
    const id = (funcionarioId ?? '').trim()
    if (!id) return null
    return `smagenda:agenda_layout:${id}`
  }, [funcionarioId])

  const [agendaLayoutVersion, setAgendaLayoutVersion] = useState(0)

  const agendaLayout = useMemo(() => {
    void agendaLayoutVersion
    const fallback: 'grade' | 'lista' = funcionario?.permissao === 'atendente' ? 'lista' : 'grade'
    if (!agendaLayoutKey) return fallback
    try {
      const saved = localStorage.getItem(agendaLayoutKey)
      if (saved === 'grade' || saved === 'lista') return saved
      return fallback
    } catch {
      return fallback
    }
  }, [agendaLayoutKey, agendaLayoutVersion, funcionario?.permissao])

  const canVerAgendaTodos = useMemo(() => {
    if (!funcionario) return false
    if (!funcionario.pode_ver_agenda) return false
    return funcionario.permissao === 'admin' || funcionario.permissao === 'atendente'
  }, [funcionario])

  const [servicos, setServicos] = useState<ServicoOption[]>([])
  const [funcionarios, setFuncionarios] = useState<FuncionarioOption[]>([])
  const [filterFuncionarioId, setFilterFuncionarioId] = useState<string>('')
  const [search, setSearch] = useState('')
  const [statusFilter, setStatusFilter] = useState('')
  const [servicoFilterId, setServicoFilterId] = useState('')

  const [agendamentos, setAgendamentos] = useState<Agendamento[]>([])
  const [bloqueios, setBloqueios] = useState<Bloqueio[]>([])
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [success, setSuccess] = useState<string | null>(null)

  const [browserNotifsPermission, setBrowserNotifsPermission] = useState<'default' | 'granted' | 'denied'>(() => {
    if (typeof window === 'undefined') return 'default'
    if (!('Notification' in window)) return 'denied'
    return Notification.permission
  })

  const notifiedIdsRef = useRef<Set<string>>(new Set())

  const [blockStart, setBlockStart] = useState('')
  const [blockEnd, setBlockEnd] = useState('')
  const [blockMotivo, setBlockMotivo] = useState('')
  const [blockRepeat, setBlockRepeat] = useState<RepeatKind>('none')
  const [blockRepeatUntil, setBlockRepeatUntil] = useState('')
  const [savingBlock, setSavingBlock] = useState(false)

  const canUseRecurringBlocks = useMemo(() => {
    const p = String(usuarioMasterPlano ?? '').trim().toLowerCase()
    return p === 'pro' || p === 'team' || p === 'enterprise'
  }, [usuarioMasterPlano])

  const effectiveBlockRepeat = canUseRecurringBlocks ? blockRepeat : 'none'
  const effectiveBlockRepeatUntil = canUseRecurringBlocks ? blockRepeatUntil : ''

  const weekStartDate = useMemo(() => startOfWeek(date), [date])
  const weekEndDate = useMemo(() => addDays(weekStartDate, 6), [weekStartDate])
  const dayKey = useMemo(() => toISODate(date), [date])
```

```
  const weekStartKey = useMemo(() => toISODate(weekStartDate), [weekStartDate])
  const weekEndKey = useMemo(() => toISODate(weekEndDate), [weekEndDate])

  const setDateFromIso = (iso: string) => {
    const parts = iso.split('-')
    if (parts.length !== 3) return
    const y = Number(parts[0])
    const m = Number(parts[1])
    const d = Number(parts[2])
    if (!Number.isFinite(y) || !Number.isFinite(m) || !Number.isFinite(d)) return
    const next = new Date(y, m - 1, d)
    if (!Number.isFinite(next.getTime())) return
    next.setHours(0, 0, 0, 0)
    setDate(next)
  }

  const rangeLabel = useMemo(() => {
    if (viewMode === 'dia') {
      return date.toLocaleDateString('pt-BR', { day: '2-digit', month: 'short', year: 'numeric' })
    }
    const startLabel = weekStartDate.toLocaleDateString('pt-BR', { day: '2-digit', month: 'short' })
    const endLabel = weekEndDate.toLocaleDateString('pt-BR', { day: '2-digit', month: 'short', year: 'numeric' })
    return `${startLabel} – ${endLabel}`
  }, [date, viewMode, weekEndDate, weekStartDate])

  const searchNorm = useMemo(() => normalizeText(search.trim()), [search])
  const matchesFilters = useMemo(() => {
    const statusNorm = statusFilter.trim().toLowerCase()
    const servicoId = servicoFilterId.trim()
    return (a: Agendamento) => {
      const aStatus = (a.status ?? '').trim().toLowerCase()
      if (!statusNorm && aStatus === 'cancelado') return false
      if (statusNorm && aStatus !== statusNorm) return false
      if (servicoId && a.servico?.id !== servicoId) return false
      if (!searchNorm) return true
      const hay = normalizeText([a.cliente_nome, a.cliente_telefone, a.servico?.nome ?? '',
a.hora_inicio].filter(Boolean).join(' '))
      return hay.includes(searchNorm)
    }
  }, [searchNorm, servicoFilterId, statusFilter])

  const hasAnyFilter = useMemo(() => Boolean(statusFilter.trim() || servicoFilterId.trim() || searchNorm.trim()),
[searchNorm, servicoFilterId, statusFilter])

  const slotStepMinutes = 30

  const selectedFuncionario = useMemo(() => {
    if (!canVerAgendaTodos) return null
    if (!filterFuncionarioId) return null
    return funcionarios.find((f) => f.id === filterFuncionarioId) ?? null
  }, [canVerAgendaTodos, filterFuncionarioId, funcionarios])

  const slots = useMemo(() => {
    const start = canVerAgendaTodos
      ? selectedFuncionario?.horario_inicio ?? usuarioMasterHorario?.horario_inicio ?? null
      : funcionario?.horario_inicio ?? usuarioMasterHorario?.horario_inicio ?? null

    const end = canVerAgendaTodos
      ? selectedFuncionario?.horario_fim ?? usuarioMasterHorario?.horario_fim ?? null
      : funcionario?.horario_fim ?? usuarioMasterHorario?.horario_fim ?? null

    if (!start || !end) return []

    const intervalStart = canVerAgendaTodos
      ? selectedFuncionario?.intervalo_inicio ?? usuarioMasterHorario?.intervalo_inicio ?? null
      : funcionario?.intervalo_inicio ?? usuarioMasterHorario?.intervalo_inicio ?? null

    const intervalEnd = canVerAgendaTodos
      ? selectedFuncionario?.intervalo_fim ?? usuarioMasterHorario?.intervalo_fim ?? null
      : funcionario?.intervalo_fim ?? usuarioMasterHorario?.intervalo_fim ?? null

    return buildSlots(start, end, intervalStart, intervalEnd, slotStepMinutes)
```

```
  }, [
    canVerAgendaTodos,
    funcionario?.horario_inicio,
    funcionario?.horario_fim,
    funcionario?.intervalo_inicio,
    funcionario?.intervalo_fim,
    selectedFuncionario?.horario_inicio,
    selectedFuncionario?.horario_fim,
    selectedFuncionario?.intervalo_inicio,
    selectedFuncionario?.intervalo_fim,
    slotStepMinutes,
    usuarioMasterHorario?.horario_inicio,
    usuarioMasterHorario?.horario_fim,
    usuarioMasterHorario?.intervalo_inicio,
    usuarioMasterHorario?.intervalo_fim,
  ])

  const totalDia = useMemo(() => {
    if (!funcionario?.pode_ver_financeiro) return 0
    return agendamentos
      .filter((a) => a.status !== 'cancelado')
      .reduce((sum, a) => sum + (a.servico?.preco ? Number(a.servico.preco) : 0), 0)
  }, [agendamentos, funcionario?.pode_ver_financeiro])

  useEffect(() => {
    const run = async () => {
      if (!funcionarioId || !usuarioMasterId) {
        setError('Este usuário não está vinculado a um usuário master (funcionarios.usuario_master_id vazio).')
        setLoading(false)
        return
      }
      setLoading(true)
      setError(null)

      const { data: masterHorarioData } = await supabase
        .from('usuarios')
        .select('horario_inicio,horario_fim,intervalo_inicio,intervalo_fim,dias_trabalho,plano')
        .eq('id', usuarioMasterId)
        .maybeSingle()
      if (masterHorarioData) {
        const row = masterHorarioData as unknown as Record<string, unknown>
        setUsuarioMasterHorario({
          horario_inicio: typeof row.horario_inicio === 'string' ? row.horario_inicio : null,
          horario_fim: typeof row.horario_fim === 'string' ? row.horario_fim : null,
          intervalo_inicio: typeof row.intervalo_inicio === 'string' ? row.intervalo_inicio : null,
          intervalo_fim: typeof row.intervalo_fim === 'string' ? row.intervalo_fim : null,
          dias_trabalho: Array.isArray(row.dias_trabalho) ? (row.dias_trabalho as number[]) : null,
        })
        setUsuarioMasterPlano(typeof row.plano === 'string' ? row.plano : null)
      } else {
        setUsuarioMasterHorario(null)
        setUsuarioMasterPlano(null)
      }

      const { data: servicosData, error: servicosError } = await supabase
        .from('servicos')
        .select('id,nome,cor')
        .eq('usuario_id', usuarioMasterId)
        .eq('ativo', true)
        .order('ordem', { ascending: true })
        .order('criado_em', { ascending: true })
      if (servicosError) {
        setError(formatQueryError(servicosError))
        setLoading(false)
        return
      }
      setServicos((servicosData ?? []) as unknown as ServicoOption[])

      if (canVerAgendaTodos) {
        const { data: funcionariosData, error: funcionariosError } = await supabase
          .from('funcionarios')
          .select('id,nome_completo,ativo,horario_inicio,horario_fim,intervalo_inicio,intervalo_fim')
```

```
          .eq('usuario_master_id', usuarioMasterId)
          .eq('permissao', 'funcionario')
          .order('criado_em', { ascending: true })
        if (funcionariosError) {
          setError(formatQueryError(funcionariosError))
          setLoading(false)
          return
        }
        setFuncionarios((funcionariosData ?? []) as unknown as FuncionarioOption[])
      } else {
        setFuncionarios([])
      }

      const base = supabase
        .from('agendamentos')

.select('id,cliente_nome,cliente_telefone,data,hora_inicio,hora_fim,status,funcionario_id,extras,servico:servico_id(id,n
ome,preco,duracao_minutos,cor)')
        .eq('usuario_id', usuarioMasterId)
        .order('data', { ascending: true })
        .order('hora_inicio', { ascending: true })

      const baseScoped = canVerAgendaTodos
        ? filterFuncionarioId
          ? base.eq('funcionario_id', filterFuncionarioId)
          : base
        : base.eq('funcionario_id', funcionarioId)

      const baseWithRange =
        viewMode === 'dia'
          ? baseScoped.eq('data', dayKey)
          : baseScoped.gte('data', weekStartKey).lte('data', weekEndKey)

      const { data: agData, error: agError } = await baseWithRange
      if (agError) {
        setError(formatQueryError(agError))
        setLoading(false)
        return
      }
      const normalizedAgs = (agData ?? []).map((row) => {
        const r = row as unknown as Record<string, unknown>
        const horaInicio = normalizeTimeHHMM(String(r.hora_inicio ?? ''))
        const horaFimRaw = r.hora_fim
        const horaFim = horaFimRaw ? normalizeTimeHHMM(String(horaFimRaw)) : null
        return { ...r, hora_inicio: horaInicio, hora_fim: horaFim } as unknown as Agendamento
      })
      setAgendamentos(normalizedAgs)

      let bloqueiosQuery = supabase
        .from('bloqueios')
        .select('id,data,hora_inicio,hora_fim,motivo,funcionario_id')
        .eq('usuario_id', usuarioMasterId)

      if (canVerAgendaTodos) {
        if (filterFuncionarioId) {
          bloqueiosQuery = bloqueiosQuery.or(`funcionario_id.is.null,funcionario_id.eq.${filterFuncionarioId}`)
        }
      } else {
        bloqueiosQuery = bloqueiosQuery.or(`funcionario_id.is.null,funcionario_id.eq.${funcionarioId}`)
      }

      bloqueiosQuery =
        viewMode === 'dia'
          ? bloqueiosQuery.eq('data', dayKey)
          : bloqueiosQuery.gte('data', weekStartKey).lte('data', weekEndKey)

      const { data: bloqueiosData, error: bloqueiosError } = await bloqueiosQuery
      if (bloqueiosError) {
        setError(formatQueryError(bloqueiosError))
        setLoading(false)
        return
      }
```

```
      const normalizedBlocks = (bloqueiosData ?? []).map((row) => {
        const r = row as unknown as Record<string, unknown>
        const horaInicio = normalizeTimeHHMM(String(r.hora_inicio ?? ''))
        const horaFim = normalizeTimeHHMM(String(r.hora_fim ?? ''))
        return { ...r, hora_inicio: horaInicio, hora_fim: horaFim } as unknown as Bloqueio
      })
      setBloqueios(normalizedBlocks)
      setLoading(false)
    }
    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar')
      setLoading(false)
    })
  }, [canVerAgendaTodos, dayKey, filterFuncionarioId, funcionarioId, usuarioMasterId, viewMode, weekEndKey, weekStartKey])

  useEffect(() => {
    if (!funcionarioId || !usuarioMasterId) return
    if (funcionario?.permissao !== 'funcionario') return

    const channel = supabase.channel(`agendamentos-funcionario:${funcionarioId}`)
    const handleIncoming = async (idRaw: unknown) => {
      const id = typeof idRaw === 'string' ? idRaw.trim() : ''
      if (!id) return
      if (notifiedIdsRef.current.has(id)) return
      notifiedIdsRef.current.add(id)

      const { data: agRow, error: agErr } = await supabase
        .from('agendamentos')
        .select('id,cliente_nome,cliente_telefone,data,hora_inicio,hora_fim,status,funcionario_id,extras,servico:servico_id(id,nome,preco,duracao_minutos,cor)')
        .eq('id', id)
        .eq('usuario_id', usuarioMasterId)
        .maybeSingle()

      if (agErr || !agRow) return
      const r = agRow as unknown as Record<string, unknown>
      if ((r.funcionario_id ?? null) !== funcionarioId) return
      if (String(r.status ?? '').trim().toLowerCase() === 'cancelado') return

      const horaInicio = normalizeTimeHHMM(String(r.hora_inicio ?? ''))
      const horaFimRaw = r.hora_fim
      const horaFim = horaFimRaw ? normalizeTimeHHMM(String(horaFimRaw)) : null

      const ag = { ...r, hora_inicio: horaInicio, hora_fim: horaFim } as unknown as Agendamento

      setAgendamentos((prev) => {
        const exists = prev.some((x) => x.id === ag.id)
        if (exists) return prev
        const next = [...prev, ag]
        next.sort((a, b) => {
          if (a.data !== b.data) return a.data < b.data ? -1 : 1
          return parseTimeToMinutes(a.hora_inicio) - parseTimeToMinutes(b.hora_inicio)
        })
        return next
      })

      const dateLabel = (() => {
        const parts = String(ag.data ?? '').split('-')
        if (parts.length !== 3) return String(ag.data ?? '')
        return `${parts[2]}/${parts[1]}/${parts[0]}`
      })()

      const msg = `Novo agendamento: ${dateLabel} ${ag.hora_inicio} • ${ag.cliente_nome}`
      setSuccess(msg)

      if (typeof window !== 'undefined' && 'Notification' in window && Notification.permission === 'granted') {
        try {
          new Notification('Novo agendamento', { body: `${dateLabel} ${ag.hora_inicio} • ${ag.cliente_nome}` })
        } catch (e: unknown) {
          void e
```

```
        }
      }
    }

    channel
      .on(
        'postgres_changes',
        {
          event: 'INSERT',
          schema: 'public',
          table: 'agendamentos',
          filter: `usuario_id=eq.${usuarioMasterId}`,
        },
        (payload) => {
          void handleIncoming((payload as { new?: { id?: unknown } }).new?.id)
        }
      )
      .on(
        'postgres_changes',
        {
          event: 'UPDATE',
          schema: 'public',
          table: 'agendamentos',
          filter: `usuario_id=eq.${usuarioMasterId}`,
        },
        (payload) => {
          void handleIncoming((payload as { new?: { id?: unknown } }).new?.id)
        }
      )

    channel.subscribe()

    return () => {
      supabase.removeChannel(channel)
    }
  }, [funcionario?.permissao, funcionarioId, usuarioMasterId])

  const enableBrowserNotifications = async () => {
    if (typeof window === 'undefined' || !('Notification' in window)) {
      setError('Seu navegador não suporta notificações.')
      setBrowserNotifsPermission('denied')
      return
    }
    setError(null)
    try {
      const perm = await Notification.requestPermission()
      setBrowserNotifsPermission(perm)
      if (perm === 'granted') setSuccess('Notificações do navegador ativadas.')
      if (perm === 'denied') setError('Notificações bloqueadas pelo navegador.')
    } catch (e: unknown) {
      setError(e instanceof Error ? e.message : 'Falha ao ativar notificações.')
    }
  }

  const prevPeriod = () => {
    if (viewMode === 'semana') {
      setDate((prev) => addDays(prev, -7))
      return
    }
    setDate((prev) => addDays(prev, -1))
  }

  const nextPeriod = () => {
    if (viewMode === 'semana') {
      setDate((prev) => addDays(prev, 7))
      return
    }
    setDate((prev) => addDays(prev, 1))
  }

  const findAgendamentoAt = (time: string) => {
    const t = parseTimeToMinutes(time)
```

```
    const overlap = (a: Agendamento) => {
      const start = parseTimeToMinutes(a.hora_inicio)
      if (!Number.isFinite(start)) return false
      const end = (() => {
        if (a.hora_fim) {
          const v = parseTimeToMinutes(a.hora_fim)
          if (Number.isFinite(v)) return v
        }
        const dur = a.servico?.duracao_minutos != null ? Number(a.servico.duracao_minutos) : 0
        if (Number.isFinite(dur) && dur > 0) return start + dur
        return start + 1
      })()
      return t >= start && t < end
    }

    const candidates = agendamentos.filter(overlap)
    if (candidates.length === 0) return null

    if (hasAnyFilter) {
      const filtered = candidates.filter(matchesFilters)
      return filtered[0] ?? null
    }

    const active = candidates.filter((a) => (a.status ?? '').trim().toLowerCase() !== 'cancelado')
    return active[0] ?? null
  }

  const findAgendamentoStartInSlot = (time: string) => {
    const t = parseTimeToMinutes(time)
    if (!Number.isFinite(t)) return null
    const end = t + slotStepMinutes

    const candidates = agendamentos
      .filter((a) => {
        const start = parseTimeToMinutes(a.hora_inicio)
        return Number.isFinite(start) && start >= t && start < end
      })
      .slice()
      .sort((x, y) => parseTimeToMinutes(x.hora_inicio) - parseTimeToMinutes(y.hora_inicio))

    if (candidates.length === 0) return null

    if (hasAnyFilter) {
      const filtered = candidates.filter(matchesFilters)
      return filtered[0] ?? null
    }

    const active = candidates.filter((a) => (a.status ?? '').trim().toLowerCase() !== 'cancelado')
    return active[0] ?? candidates[0] ?? null
  }

  const findBloqueioAt = (time: string) => {
    const t = parseTimeToMinutes(time)
    return bloqueios.find((b) => {
      const s = parseTimeToMinutes(b.hora_inicio)
      const e = parseTimeToMinutes(b.hora_fim)
      return t >= s && t < e
    })
  }

  const findBloqueioStartInSlot = (time: string) => {
    const t = parseTimeToMinutes(time)
    if (!Number.isFinite(t)) return null
    const end = t + slotStepMinutes

    const candidates = bloqueios
      .filter((b) => {
        const start = parseTimeToMinutes(b.hora_inicio)
        return Number.isFinite(start) && start >= t && start < end
      })
      .slice()
      .sort((x, y) => parseTimeToMinutes(x.hora_inicio) - parseTimeToMinutes(y.hora_inicio))
```

```
    return candidates[0] ?? null
  }

  const weekDays = useMemo(() => {
    if (viewMode !== 'semana') return [] as Array<{ date: Date; key: string }>
    return Array.from({ length: 7 }).map((_, i) => {
      const d = addDays(weekStartDate, i)
      return { date: d, key: toISODate(d) }
    })
  }, [viewMode, weekStartDate])

  const agendamentosByDay = useMemo(() => {
    const map: Record<string, Agendamento[]> = {}
    for (const a of agendamentos) {
      const key = a.data
      if (!map[key]) map[key] = []
      map[key].push(a)
    }
    for (const key of Object.keys(map)) {
      map[key].sort((x, y) => parseTimeToMinutes(x.hora_inicio) - parseTimeToMinutes(y.hora_inicio))
    }
    return map
  }, [agendamentos])

  const bloqueiosByDay = useMemo(() => {
    const map: Record<string, Bloqueio[]> = {}
    for (const b of bloqueios) {
      const key = b.data
      if (!map[key]) map[key] = []
      map[key].push(b)
    }
    for (const key of Object.keys(map)) {
      map[key].sort((x, y) => parseTimeToMinutes(x.hora_inicio) - parseTimeToMinutes(y.hora_inicio))
    }
    return map
  }, [bloqueios])

  const funcionarioNameById = useMemo(() => {
    const map: Record<string, string> = {}
    for (const f of funcionarios) map[f.id] = f.nome_completo
    return map
  }, [funcionarios])

  const visibleAgendamentos = useMemo(() => {
    return agendamentos
      .filter(matchesFilters)
      .slice()
      .sort((a, b) => {
        if (a.data !== b.data) return a.data < b.data ? -1 : 1
        return parseTimeToMinutes(a.hora_inicio) - parseTimeToMinutes(b.hora_inicio)
      })
  }, [agendamentos, matchesFilters])

  const listItems = useMemo(() => {
    const items: Array<
      | { kind: 'ag'; data: string; hora_inicio: string; ag: Agendamento }
      | { kind: 'b'; data: string; hora_inicio: string; b: Bloqueio }
    > = []

    for (const ag of visibleAgendamentos) items.push({ kind: 'ag', data: ag.data, hora_inicio: ag.hora_inicio, ag })
    for (const b of bloqueios) items.push({ kind: 'b', data: b.data, hora_inicio: b.hora_inicio, b })

    return items.sort((a, b) => {
      if (a.data !== b.data) return a.data < b.data ? -1 : 1
      return parseTimeToMinutes(a.hora_inicio) - parseTimeToMinutes(b.hora_inicio)
    })
  }, [bloqueios, visibleAgendamentos])

  const repeatPreview = useMemo(() => {
    return buildRepeatKeys(dayKey, effectiveBlockRepeat, effectiveBlockRepeatUntil.trim() ? effectiveBlockRepeatUntil :
null)
```

```
  }, [dayKey, effectiveBlockRepeat, effectiveBlockRepeatUntil])

  const canSaveBlock = useMemo(() => {
    if (!funcionarioId || !usuarioMasterId) return false
    if (!funcionario?.pode_bloquear_horarios) return false
    if (!blockStart || !blockEnd) return false
    const s = parseTimeToMinutes(blockStart)
    const e = parseTimeToMinutes(blockEnd)
    if (!Number.isFinite(s) || !Number.isFinite(e)) return false
    if (e <= s) return false
    if (repeatPreview.error) return false
    if (repeatPreview.keys.length === 0) return false
    return true
  }, [funcionario?.pode_bloquear_horarios, funcionarioId, usuarioMasterId, blockStart, blockEnd, repeatPreview.error,
repeatPreview.keys.length])

  const createBloqueio = async () => {
    if (!funcionarioId || !usuarioMasterId || !canSaveBlock) return
    setSavingBlock(true)
    setError(null)
    const motivo = blockMotivo.trim() ? blockMotivo.trim() : null

    const { keys, error: repeatErr } = buildRepeatKeys(
      dayKey,
      effectiveBlockRepeat,
      effectiveBlockRepeatUntil.trim() ? effectiveBlockRepeatUntil : null
    )
    if (repeatErr) {
      setError(repeatErr)
      setSavingBlock(false)
      return
    }

    const s = parseTimeToMinutes(blockStart)
    const e = parseTimeToMinutes(blockEnd)
    type OcupacaoRow = { start_min: number; end_min: number }
    for (const k of keys) {
      const { data: ocupacoesData, error: ocupacoesErr } = await supabase.rpc('public_get_ocupacoes', {
        p_usuario_id: usuarioMasterId,
        p_data: k,
        p_funcionario_id: funcionarioId,
      })
      if (ocupacoesErr) {
        setError(ocupacoesErr.message)
        setSavingBlock(false)
        return
      }
      const occupied = ((ocupacoesData ?? []) as unknown as OcupacaoRow[]).some((r) => overlaps(s, e, r.start_min,
r.end_min))
      if (occupied) {
        setError(`Conflito de horário em ${new Date(k).toLocaleDateString('pt-BR')}.`)
        setSavingBlock(false)
        return
      }
    }

    const rows = keys.map((k) => ({
      usuario_id: usuarioMasterId,
      data: k,
      hora_inicio: blockStart,
      hora_fim: blockEnd,
      motivo,
      funcionario_id: funcionarioId,
    }))

    const { data: createdRows, error: err } = await supabase
      .from('bloqueios')
      .insert(rows)
      .select('id,data,hora_inicio,hora_fim,motivo,funcionario_id')

    if (err) {
      setError(err.message)
```

```
      setSavingBlock(false)
      return
    }

    const inserted = (createdRows ?? []) as unknown as Bloqueio[]
    const inRange = (b: Bloqueio) =>
      viewMode === 'dia' ? b.data === dayKey : b.data >= weekStartKey && b.data <= weekEndKey

    setBloqueios((prev) => [...prev, ...inserted.filter(inRange)])
    setBlockMotivo('')
    setBlockRepeat('none')
    setBlockRepeatUntil('')
    setSavingBlock(false)
  }

  const removeBloqueio = async (b: Bloqueio) => {
    if (!funcionarioId || !usuarioMasterId) return
    if (!funcionario?.pode_bloquear_horarios) return
    if (b.funcionario_id !== funcionarioId) return
    setError(null)
    const ok = window.confirm('Remover este bloqueio?')
    if (!ok) return
    const { error: err } = await supabase.from('bloqueios').delete().eq('id', b.id).eq('usuario_id', usuarioMasterId)
    if (err) {
      setError(err.message)
      return
    }
    setBloqueios((prev) => prev.filter((x) => x.id !== b.id))
  }

  if (!funcionario) {
    return (
      <AppShell>
        <div className="text-slate-700">Acesso restrito.</div>
      </AppShell>
    )
  }

  return (
    <PageTutorial usuarioId={funcionarioId} page="funcionario_agenda">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="space-y-6">
            <div className="flex items-center justify-between gap-3">
              <div>
                <div className="text-sm font-semibold text-slate-500">Agenda</div>
                <div className="text-xl font-semibold text-slate-900">Meus agendamentos</div>
              </div>
              <div className="flex items-center gap-2">
                {funcionario.permissao === 'funcionario' && browserNotifsPermission !== 'granted' ? (
                  <Button variant="secondary" onClick={() => void enableBrowserNotifications()}>
                    Ativar notificações
                  </Button>
                ) : null}
                <Button variant="secondary" onClick={resetTutorial}>
                  Rever tutorial
                </Button>
              </div>
            </div>

            {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}
</div> : null}
            {success ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">
{success}</div> : null}

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'filtros'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
```

```
          <Card>
            <div className="p-6">
              <div className="flex flex-col gap-3 sm:flex-row sm:items-center sm:justify-between">
                <div className="flex flex-wrap items-center gap-3">
                  <Button variant="secondary" onClick={prevPeriod}>
                    &lt;
                  </Button>
                  <div className="text-sm font-semibold text-slate-900">{rangeLabel}</div>
                  <Button variant="secondary" onClick={nextPeriod}>
                    &gt;
                  </Button>

                  <div className="flex items-center gap-1 rounded-lg bg-slate-100 p-1">
                    <button
                      type="button"
                      onClick={() => setViewMode('dia')}
                      className={[
                        'rounded-md px-3 py-1 text-sm font-medium',
                        viewMode === 'dia' ? 'bg-white text-slate-900 shadow-sm' : 'text-slate-700 hover:text-slate-
900',
                      ].join(' ')}
                    >
                      Dia
                    </button>
                    <button
                      type="button"
                      onClick={() => setViewMode('semana')}
                      className={[
                        'rounded-md px-3 py-1 text-sm font-medium',
                        viewMode === 'semana' ? 'bg-white text-slate-900 shadow-sm' : 'text-slate-700 hover:text-
slate-900',
                      ].join(' ')}
                    >
                      Semana
                    </button>
                  </div>

                  <div className="flex items-center gap-1 rounded-lg bg-slate-100 p-1">
                    <button
                      type="button"
                      onClick={() => {
                        if (agendaLayoutKey) {
                          try {
                            localStorage.setItem(agendaLayoutKey, 'grade')
                          } catch {
                            void 0
                          }
                        }
                        setAgendaLayoutVersion((v) => v + 1)
                      }}
                      className={[
                        'rounded-md px-3 py-1 text-sm font-medium',
                        agendaLayout === 'grade'
                          ? 'bg-white text-slate-900 shadow-sm'
                          : 'text-slate-700 hover:text-slate-900',
                      ].join(' ')}
                    >
                      Grade
                    </button>
                    <button
                      type="button"
                      onClick={() => {
                        if (agendaLayoutKey) {
                          try {
                            localStorage.setItem(agendaLayoutKey, 'lista')
                          } catch {
                            void 0
                          }
                        }
                        setAgendaLayoutVersion((v) => v + 1)
                      }}
                      className={[
```

```
                            'rounded-md px-3 py-1 text-sm font-medium',
                            agendaLayout === 'lista'
                              ? 'bg-white text-slate-900 shadow-sm'
                              : 'text-slate-700 hover:text-slate-900',
                          ].join(' ')}
                        >
                          Lista
                        </button>
                      </div>
                    </div>
                    <div className="text-sm text-slate-600">Mostrando: {canVerAgendaTodos ? 'agenda completa' : 'meus agendamentos'}</div>
                  </div>

                  {canVerAgendaTodos ? (
                    <div className="mt-4 flex flex-wrap items-center gap-2">
                      <div className="text-sm text-slate-600">Filtrar por:</div>
                      <button
                        type="button"
                        onClick={() => setFilterFuncionarioId('')}
                        className={[
                          'rounded-lg px-3 py-1.5 text-sm font-medium',
                          !filterFuncionarioId ? 'bg-slate-900 text-white' : 'bg-slate-100 text-slate-700 hover:bg-slate-200',
                        ].join(' ')}
                      >
                        Todos
                      </button>
                      {funcionarios
                        .filter((f) => f.ativo)
                        .map((f) => (
                          <button
                            key={f.id}
                            type="button"
                            onClick={() => setFilterFuncionarioId(f.id)}
                            className={[
                              'rounded-lg px-3 py-1.5 text-sm font-medium',
                              filterFuncionarioId === f.id ? 'bg-slate-900 text-white' : 'bg-slate-100 text-slate-700 hover:bg-slate-200',
                            ].join(' ')}
                          >
                            {f.nome_completo.split(' ')[0]}
                          </button>
                        ))}
                    </div>
                  ) : null}

                  <div className="mt-4 grid grid-cols-1 gap-3 sm:grid-cols-12">
                    <div className="sm:col-span-3">
                      <Input
                        label="Data"
                        type="date"
                        value={dayKey}
                        onChange={(e) => {
                          const v = e.target.value
                          if (!v.trim()) return
                          setDateFromIso(v.trim())
                        }}
                      />
                    </div>
                    <div className="sm:col-span-4">
                      <Input label="Buscar" value={search} onChange={(e) => setSearch(e.target.value)} placeholder="Nome, telefone, serviço…" />
                    </div>
                    <label className="block sm:col-span-2">
                      <div className="text-sm font-medium text-slate-700 mb-1">Status</div>
                      <select
                        className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900 outline-none focus:ring-2 focus:ring-slate-300"
                        value={statusFilter}
                        onChange={(e) => setStatusFilter(e.target.value)}
                      >
```

```
                        <option value="">Todos</option>
                        <option value="pendente">Pendente</option>
                        <option value="confirmado">Confirmado</option>
                        <option value="concluido">Concluído</option>
                        <option value="nao_compareceu">No-show</option>
                        <option value="cancelado">Cancelado</option>
                      </select>
                    </label>
                    <label className="block sm:col-span-3">
                      <div className="text-sm font-medium text-slate-700 mb-1">Serviço</div>
                      <select
                        className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                        value={servicoFilterId}
                        onChange={(e) => setServicoFilterId(e.target.value)}
                      >
                        <option value="">Todos</option>
                        {servicos.map((s) => (
                          <option key={s.id} value={s.id}>
                            {s.nome}
                          </option>
                        ))}
                      </select>
                    </label>
                  </div>
                </div>
              </Card>
            </div>

            {funcionario.pode_bloquear_horarios ? (
              <div
                className={
                  tutorialOpen && tutorialSteps[tutorialStep]?.target === 'bloqueio'
                    ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                    : ''
                }
              >
                <Card>
                  <div className="p-6 space-y-4">
              <div>
                <div className="text-sm font-semibold text-slate-900">Bloquear horário</div>
                <div className="text-xs text-slate-600">Cria um bloqueio apenas para você.</div>
              </div>

              <div className="grid grid-cols-1 gap-4 sm:grid-cols-3">
                <Input label="Início" type="time" value={blockStart} onChange={(e) => setBlockStart(e.target.value)}
step={900} />
                <Input label="Fim" type="time" value={blockEnd} onChange={(e) => setBlockEnd(e.target.value)} step={900}
/>
                <Input label="Motivo (opcional)" value={blockMotivo} onChange={(e) => setBlockMotivo(e.target.value)} />
              </div>

              <div className="grid grid-cols-1 gap-4 sm:grid-cols-3">
                <label className="block">
                  <div className="text-sm font-medium text-slate-700 mb-1">Repetir</div>
                  <select
                    className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                    value={effectiveBlockRepeat}
                    disabled={!canUseRecurringBlocks}
                    onChange={(e) => {
                      if (!canUseRecurringBlocks) return
                      const next = e.target.value as RepeatKind
                      setBlockRepeat(next)
                      if (next === 'none') {
                        setBlockRepeatUntil('')
                        return
                      }
                      if (!blockRepeatUntil.trim()) {
                        setBlockRepeatUntil(toISODate(addDays(date, next === 'monthly' ? 180 : 30)))
                      }
                    }}
```

```
                    >
                      <option value="none">Não repetir</option>
                      {canUseRecurringBlocks ? <option value="daily">Diariamente</option> : null}
                      {canUseRecurringBlocks ? <option value="weekly">Semanalmente</option> : null}
                      {canUseRecurringBlocks ? <option value="monthly">Mensalmente</option> : null}
                    </select>
                  </label>
                  <div className="sm:col-span-2">
                    {effectiveBlockRepeat !== 'none' ? (
                      <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                        <Input
                          label="Até"
                          type="date"
                          min={dayKey}
                          value={effectiveBlockRepeatUntil}
                          onChange={(e) => setBlockRepeatUntil(e.target.value)}
                        />
                        <div className="flex items-end">
                          <div className="text-xs text-slate-600">
                            {repeatPreview.error ? repeatPreview.error : `Serão criados ${repeatPreview.keys.length}
bloqueios.`}
                          </div>
                        </div>
                      </div>
                    ) : null}
                  </div>
                </div>

                <div className="flex justify-end">
                  <Button onClick={createBloqueio} disabled={!canSaveBlock || savingBlock}>
                    Salvar bloqueio
                  </Button>
                </div>

                {bloqueios.filter((b) => b.funcionario_id === funcionarioId).length > 0 ? (
                  <div className="rounded-xl border border-slate-200 divide-y divide-slate-100">
                    {bloqueios
                      .filter((b) => b.funcionario_id === funcionarioId)
                      .slice()
                      .sort((a, b) => parseTimeToMinutes(a.hora_inicio) - parseTimeToMinutes(b.hora_inicio))
                      .map((b) => (
                        <div key={b.id} className="p-3 flex items-center justify-between gap-3">
                          <div className="text-sm text-slate-700">
                            <span className="font-semibold text-slate-900">{b.hora_inicio}</span>
                            {' – '}
                            <span className="font-semibold text-slate-900">{b.hora_fim}</span>
                            {b.motivo ? <span className="text-slate-500"> {' • '}{b.motivo}</span> : null}
                          </div>
                          <Button variant="danger" onClick={() => removeBloqueio(b)}>
                            Remover
                          </Button>
                        </div>
                      ))}
                  </div>
                ) : null}
                    </div>
                  </Card>
                </div>
              ) : null}

              <div
                className={
                  tutorialOpen && tutorialSteps[tutorialStep]?.target === 'agenda'
                    ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                    : ''
                }
              >
                <Card>
                  <div className="divide-y divide-slate-100">
              {loading ? (
                <div className="p-6 text-sm text-slate-600">Carregando agenda…</div>
              ) : agendaLayout === 'lista' ? (
```

```
              <div className="p-6 space-y-4">
                <div className="flex flex-col gap-2 sm:flex-row sm:items-center sm:justify-between">
                  <div className="text-sm text-slate-600">
                    {visibleAgendamentos.length} agendamentos • {bloqueios.length} bloqueios
                  </div>
                  <div className="flex flex-wrap gap-2">
                    {hasAnyFilter ? (
                      <Button
                        variant="secondary"
                        onClick={() => {
                          setSearch('')
                          setStatusFilter('')
                          setServicoFilterId('')
                        }}
                      >
                        Limpar filtros
                      </Button>
                    ) : null}
                    <Button variant="secondary" onClick={() => void refresh()}>
                      Recarregar
                    </Button>
                  </div>
                </div>

                {visibleAgendamentos.length === 0 && bloqueios.length === 0 ? (
                  <div className="rounded-xl border border-slate-200 bg-slate-50 p-4 space-y-2">
                    <div className="text-sm font-semibold text-slate-900">Nenhum item encontrado</div>
                    <div className="text-sm text-slate-600">Não há agendamentos/bloqueios neste período com os filtros atuais.</div>
                    <div className="text-xs text-slate-500 space-y-1">
                      <div>
                        Período: <span className="font-medium text-slate-700">{rangeLabel}</span>
                      </div>
                      {canVerAgendaTodos ? (
                        <div>
                          Profissional:{' '}
                          <span className="font-medium text-slate-700">
                            {filterFuncionarioId
                              ? funcionarios.find((f) => f.id === filterFuncionarioId)?.nome_completo ?? 'Selecionado'
                              : 'Todos'}
                          </span>
                        </div>
                      ) : null}
                      <div>
                        Filtros:{' '}
                        <span className="font-medium text-slate-700">
                          {[
                            searchNorm ? 'busca' : null,
                            statusFilter.trim() ? `status=${statusFilter.trim()}` : null,
                            servicoFilterId.trim() ? 'serviço' : null,
                          ]
                            .filter(Boolean)
                            .join(' • ') || 'nenhum'}
                        </span>
                      </div>
                      {canVerAgendaTodos ? (
                        <div>
                          Profissionais carregados: <span className="font-medium text-slate-700">{funcionarios.filter((f) => f.ativo).length}</span>
                        </div>
                      ) : null}
                    </div>

                    {canVerAgendaTodos && funcionarios.length === 0 ? (
                      <div className="text-xs text-amber-700 rounded-lg border border-amber-200 bg-amber-50 p-2">
                        Se existem agendamentos no período, isso normalmente indica políticas RLS antigas no Supabase (o select retorna vazio sem erro).
                      </div>
                    ) : null}
                  </div>
                ) : (
                  <div className="rounded-xl border border-slate-200 divide-y divide-slate-100 overflow-hidden">
```

```
                    {listItems.map((item) => {
                      if (item.kind === 'b') {
                        const labelDate =
                          viewMode === 'semana'
                            ? parseDateKey(item.data).toLocaleDateString('pt-BR', { weekday: 'short', day: '2-digit',
month: '2-digit' })
                            : null
                        return (
                          <div key={`b:${item.b.id}`} className="p-3 flex items-start justify-between gap-3">
                            <div className="min-w-0">
                              {labelDate ? <div className="text-xs text-slate-500">{labelDate}</div> : null}
                              <div className="text-sm font-semibold text-slate-900">
                                {item.b.hora_inicio}–{item.b.hora_fim} • 🔒 Bloqueado
                              </div>
                              {item.b.motivo ? <div className="text-sm text-slate-600">{item.b.motivo}</div> : null}
                            </div>
                            <Badge tone="yellow">Bloqueado</Badge>
                          </div>
                        )
                      }

                      const ag = item.ag
                      const statusUi = resolveStatusUi(ag.status)
                      const endereco = readExtrasEndereco(ag.extras)
                      const labelDate =
                        viewMode === 'semana'
                          ? parseDateKey(item.data).toLocaleDateString('pt-BR', { weekday: 'short', day: '2-digit',
month: '2-digit' })
                          : null

                      const profName = ag.funcionario_id ? funcionarioNameById[ag.funcionario_id] : ''
                      const profLabel = canVerAgendaTodos && !filterFuncionarioId ? (profName ? profName :
'Profissional') : null

                      return (
                        <div key={`ag:${ag.id}`} className="p-3 flex items-start justify-between gap-3">
                          <div className="min-w-0">
                            {labelDate ? <div className="text-xs text-slate-500">{labelDate}</div> : null}
                            <div className="text-sm font-semibold text-slate-900 truncate">
                              {(normalizeTimeHHMM(ag.hora_inicio) || '—')} — {ag.cliente_nome}
                            </div>
                            {profLabel ? <div className="text-xs text-slate-500">{profLabel}</div> : null}
                            <div className="text-sm text-slate-600">📱 {ag.cliente_telefone}</div>
                            {endereco ? <div className="text-sm text-slate-600">📍 {endereco}</div> : null}
                            <div className="text-sm text-slate-700">
                              ✂️ {ag.servico?.nome}{' '}
                              {funcionario.pode_ver_financeiro && ag.servico?.preco ? `-
${formatBRMoney(Number(ag.servico.preco))}` : ''}
                            </div>
                          </div>
                          <Badge tone={statusUi.tone}>{statusUi.label}</Badge>
                        </div>
                      )
                    })}
                  </div>
                )}
              </div>
            ) : slots.length === 0 ? (
              <div className="p-6 space-y-3">
                <div className="text-sm text-slate-600">
                  {canVerAgendaTodos && filterFuncionarioId
                    ? 'Horário não configurado para este profissional.'
                    : 'Horário não configurado. Peça ao gerente para ajustar seu horário.'}
                </div>
                <div className="flex flex-wrap gap-2">
                  <Button variant="secondary" onClick={() => void refresh()}>
                    Recarregar
                  </Button>
                </div>
                {agendamentos.length > 0 ? (
                  <div className="rounded-xl border border-slate-200 divide-y divide-slate-100">
                    {agendamentos.map((ag) => {
```

```
                      const statusUi = resolveStatusUi(ag.status)
                      const endereco = readExtrasEndereco(ag.extras)
                      return (
                        <div key={ag.id} className="p-3 flex items-start justify-between gap-3">
                          <div className="min-w-0">
                            <div className="text-sm font-semibold text-slate-900 truncate">
                              {(normalizeTimeHHMM(ag.hora_inicio) || '—')} — {ag.cliente_nome}
                            </div>
                            <div className="text-sm text-slate-600">📱 {ag.cliente_telefone}</div>
                            {endereco ? <div className="text-sm text-slate-600">📍 {endereco}</div> : null}
                            <div className="text-sm text-slate-700">
                              ✂️ {ag.servico?.nome}{' '}
                              {funcionario.pode_ver_financeiro && ag.servico?.preco ? `-
${formatBRMoney(Number(ag.servico.preco))}` : ''}
                            </div>
                          </div>
                          <Badge tone={statusUi.tone}>{statusUi.label}</Badge>
                        </div>
                      )
                    })}
                  </div>
                ) : null}
              </div>
            ) : viewMode === 'dia' ? (
              <>
                {visibleAgendamentos.length === 0 && bloqueios.length === 0 ? (
                  <div className="p-4 space-y-2">
                    <div className="text-sm text-slate-600">Nenhum agendamento encontrado neste período.</div>
                    <div className="flex flex-wrap gap-2">
                      {hasAnyFilter ? (
                        <Button
                          variant="secondary"
                          onClick={() => {
                            setSearch('')
                            setStatusFilter('')
                            setServicoFilterId('')
                          }}
                        >
                          Limpar filtros
                        </Button>
                      ) : null}
                      <Button variant="secondary" onClick={() => void refresh()}>
                        Recarregar
                      </Button>
                    </div>
                  </div>
                ) : null}
                {slots.map((time) => {
                const agStart = findAgendamentoStartInSlot(time)
                const agCover = agStart ?? findAgendamentoAt(time)
                const blockStart = findBloqueioStartInSlot(time)
                const blockCover = blockStart ?? findBloqueioAt(time)

                if (agCover) {
                  const isStart = Boolean(agStart)
                  const visible = isStart && matchesFilters(agCover)
                  const statusUi = resolveStatusUi(agCover.status)
                  const startLabel = normalizeTimeHHMM(agCover.hora_inicio)
                  const timeLabel = isStart && startLabel ? startLabel : time
                  const endereco = visible ? readExtrasEndereco(agCover.extras) : null
                  return (
                    <div key={time} className="p-4 flex items-start justify-between gap-3">
                      <div>
                        <div className="text-sm font-semibold text-slate-900 flex items-center gap-2">
                          {agCover.servico?.cor ? (
                            <span className="h-2.5 w-2.5 rounded-full" style={{ backgroundColor: agCover.servico.cor }}
/>
                          ) : null}
                          <span>
                            {timeLabel} - {visible ? agCover.cliente_nome : 'Ocupado'}
                          </span>
                        </div>
```

```
                        {visible ? <div className="text-sm text-slate-600">📱 {agCover.cliente_telefone}</div> : null}
                        {endereco ? <div className="text-sm text-slate-600">📍 {endereco}</div> : null}
                        {visible ? (
                          <div className="text-sm text-slate-700">
                            ✂️ {agCover.servico?.nome}{' '}
                            {funcionario.pode_ver_financeiro && agCover.servico?.preco ? `-
${formatBRMoney(Number(agCover.servico.preco))}` : ''}
                          </div>
                        ) : null}
                      </div>
                      <div className="flex flex-col items-end gap-2">
                        <Badge tone={statusUi.tone}>{statusUi.label}</Badge>
                        {agCover.status !== 'cancelado' && visible ? (
                          <div className="flex gap-2">
                            {funcionario.pode_criar_agendamentos && String(agCover.status ?? '').trim().toLowerCase()
!== 'confirmado' ? (
                              <Button
                                variant="secondary"
                                onClick={async () => {
                                  setError(null)
                                  const { error: updErr } = await supabase
                                    .from('agendamentos')
                                    .update({ status: 'confirmado' })
                                    .eq('id', agCover.id)
                                    .eq('funcionario_id', funcionario.id)
                                  if (updErr) {
                                    const formatted = formatSupabaseError(updErr)
                                    const lower = formatted.toLowerCase()
                                    if (lower.includes('internal server error') || lower.includes('500')) {
                                      setError(
                                        `${formatted} • Provável trigger/função no Postgres falhando. Reexecute no
Supabase o “SQL do WhatsApp (trigger confirmação imediata)” e o “SQL de Logs de Auditoria”.`
                                      )
                                      return
                                    }
                                    setError(formatted)
                                    return
                                  }
                                  setAgendamentos((prev) => prev.map((x) => (x.id === agCover.id ? { ...x, status:
'confirmado' } : x)))
                                  const sendRes = await sendConfirmacaoWhatsapp(agCover.id)
                                  if (!sendRes.ok) {
                                  if (typeof sendRes.body === 'object' && sendRes.body !== null && (sendRes.body as
Record<string, unknown>).error === 'supabase_gateway_invalid_jwt') {
                                    setError('JWT inválido para chamar a Edge Function. Saia e entre novamente. Se
persistir, reimplante a função com verify_jwt=false (--no-verify-jwt).')
                                    return
                                  }
                                  if (typeof sendRes.body === 'object' && sendRes.body !== null && (sendRes.body as
Record<string, unknown>).error === 'jwt_project_mismatch') {
                                    setError('Sessão do Supabase pertence a outro projeto. Saia e entre novamente no
sistema.')
                                    return
                                  }
                                    if (typeof sendRes.body === 'object' && sendRes.body !== null && (sendRes.body as
Record<string, unknown>).error === 'invalid_jwt') {
                                      setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
                                      return
                                    }
                                    if (typeof sendRes.body === 'object' && sendRes.body !== null) {
                                      const hint = (sendRes.body as Record<string, unknown>).hint
                                      if (typeof hint === 'string' && hint.trim()) {
                                        setError(hint)
                                        return
                                      }
                                      const code = (sendRes.body as Record<string, unknown>).error
                                      if (code === 'instance_not_connected') {
                                        setError('WhatsApp não conectado. Vá em Configurações > WhatsApp e conecte a
instância (QR Code).')
                                        return
                                      }
                                    }
```

```
                                    const details = typeof sendRes.body === 'string' ? sendRes.body :
JSON.stringify(sendRes.body)
                                    setError(`Falha ao enviar confirmação (HTTP ${sendRes.status}): ${details}`)
                                  }
                                }}
                              >
                                ✓ Confirmar
                              </Button>
                            ) : null}
                            {funcionario.pode_cancelar_agendamentos ? (
                              <Button
                                variant="secondary"
                                onClick={async () => {
                                  setError(null)
                                  const ok = window.confirm('Marcar como no-show?')
                                  if (!ok) return
                                  const { error: updErr } = await supabase
                                    .from('agendamentos')
                                    .update({ status: 'nao_compareceu' })
                                    .eq('id', agCover.id)
                                    .eq('funcionario_id', funcionario.id)
                                  if (updErr) {
                                    setError(formatSupabaseError(updErr))
                                    return
                                  }
                                  setAgendamentos((prev) => prev.map((x) => (x.id === agCover.id ? { ...x, status:
'nao_compareceu' } : x)))
                                }}
                              >
                                No-show
                              </Button>
                            ) : null}
                            {funcionario.pode_cancelar_agendamentos ? (
                              <Button
                                variant="danger"
                                onClick={async () => {
                                  setError(null)
                                  setSuccess(null)
                                  const { error: updErr } = await supabase
                                    .from('agendamentos')
                                    .update({ status: 'cancelado', cancelado_em: new Date().toISOString() })
                                    .eq('id', agCover.id)
                                    .eq('funcionario_id', funcionario.id)
                                  if (updErr) {
                                    setError(formatSupabaseError(updErr))
                                    return
                                  }
                                  setAgendamentos((prev) => prev.map((x) => (x.id === agCover.id ? { ...x, status:
'cancelado' } : x)))
                                  setSuccess('Agendamento cancelado.')
                                  const sendRes = await sendCancelamentoWhatsapp(agCover.id)
                                  if (sendRes.ok) {
                                    if (typeof sendRes.body === 'object' && sendRes.body !== null) {
                                      const skipped = (sendRes.body as Record<string, unknown>).skipped
                                      if (skipped === 'not_configured') {
                                        setSuccess('Agendamento cancelado. WhatsApp não configurado.')
                                        return
                                      }
                                      if (skipped === 'disabled') {
                                        setSuccess('Agendamento cancelado. Envio automático desabilitado.')
                                        return
                                      }
                                    }
                                    setSuccess('Agendamento cancelado e aviso enviado.')
                                    return
                                  }
                                  if (typeof sendRes.body === 'object' && sendRes.body !== null && (sendRes.body as
Record<string, unknown>).error === 'supabase_gateway_invalid_jwt') {
                                    setError('JWT inválido para chamar a Edge Function. Saia e entre novamente. Se
persistir, reimplante a função com verify_jwt=false (--no-verify-jwt).')
                                    return
                                  }
```

```
                                  if (typeof sendRes.body === 'object' && sendRes.body !== null && (sendRes.body as
Record<string, unknown>).error === 'jwt_project_mismatch') {
                                    setError('Sessão do Supabase pertence a outro projeto. Saia e entre novamente no
sistema.')
                                    return
                                  }
                                  if (typeof sendRes.body === 'object' && sendRes.body !== null && (sendRes.body as
Record<string, unknown>).error === 'invalid_jwt') {
                                    setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
                                    return
                                  }
                                  if (typeof sendRes.body === 'object' && sendRes.body !== null) {
                                    const hint = (sendRes.body as Record<string, unknown>).hint
                                    if (typeof hint === 'string' && hint.trim()) {
                                      setError(hint)
                                      return
                                    }
                                    const code = (sendRes.body as Record<string, unknown>).error
                                    if (code === 'instance_not_connected') {
                                      setError('WhatsApp não conectado. Vá em Configurações > WhatsApp e conecte a
instância (QR Code).')
                                      return
                                    }
                                  }
                                  const details = typeof sendRes.body === 'string' ? sendRes.body :
JSON.stringify(sendRes.body)
                                  setError(`Falha ao enviar cancelamento (HTTP ${sendRes.status}): ${details}`)
                                }}
                              >
                                ✗ Cancelar
                              </Button>
                            ) : null}
                          </div>
                        ) : null}
                      </div>
                    </div>
                  )
                }
                if (blockCover) {
                  const isStart = Boolean(blockStart)
                  const canDelete = funcionario.pode_bloquear_horarios && blockCover.funcionario_id === funcionarioId &&
isStart
                  const timeLabel = isStart ? blockCover.hora_inicio : time
                  return (
                    <div key={time} className="p-4 flex items-center justify-between">
                      <div className="text-sm font-semibold text-slate-900">
                        {timeLabel} - 🔒 BLOQUEADO {isStart && blockCover.motivo ? `(${blockCover.motivo})` : ''}
                      </div>
                      <div className="flex items-center gap-2">
                        {canDelete ? (
                          <Button variant="danger" onClick={() => removeBloqueio(blockCover)}>
                            Remover
                          </Button>
                        ) : null}
                        <Badge tone="yellow">Bloqueado</Badge>
                      </div>
                    </div>
                  )
                }
                return (
                  <div key={time} className="p-4 flex items-center justify-between">
                    <div className="text-sm text-slate-600">{time} - LIVRE</div>
                  </div>
                )
              })}
              </>
            ) : (
              <div className="p-4">
                {visibleAgendamentos.length === 0 && bloqueios.length === 0 ? (
                  <div className="mb-3 rounded-xl border border-slate-200 bg-slate-50 p-3 flex flex-col gap-2 sm:flex-
row sm:items-center sm:justify-between">
                    <div className="text-sm text-slate-700">Nenhum agendamento encontrado nesta semana.</div>
```

```
                    <div className="flex flex-wrap gap-2">
                      {hasAnyFilter ? (
                        <Button
                          variant="secondary"
                          onClick={() => {
                            setSearch('')
                            setStatusFilter('')
                            setServicoFilterId('')
                          }}
                        >
                          Limpar filtros
                        </Button>
                      ) : null}
                      <Button variant="secondary" onClick={() => void refresh()}>
                        Recarregar
                      </Button>
                    </div>
                  </div>
                ) : null}
                <div className="overflow-x-auto">
                  <div className="grid grid-cols-7 gap-3 min-w-[980px]">
                    {weekDays.map((d) => {
                      const ags = agendamentosByDay[d.key] ?? []
                      const bls = bloqueiosByDay[d.key] ?? []
                      const visibleAgs = ags.filter(matchesFilters)
                      const visibleAgCount = visibleAgs.length
                      const label = d.date.toLocaleDateString('pt-BR', { weekday: 'short', day: '2-digit', month: '2-digit' })
                      return (
                        <div key={d.key} className="rounded-xl border border-slate-200 bg-white overflow-hidden">
                          <div className="px-3 py-2 border-b border-slate-100 bg-slate-50">
                            <div className="text-sm font-semibold text-slate-900 flex items-center justify-between gap-2">
                              <span>{label}</span>
                              <span className="text-xs text-slate-600">{visibleAgCount} ag.</span>
                            </div>
                          </div>
                          <div className="divide-y divide-slate-100">
                            {visibleAgs.length === 0 && bls.length === 0 ? (
                              <div className="px-3 py-3 space-y-2">
                                <div className="text-sm text-slate-600">Sem itens</div>
                                <div>
                                  {hasAnyFilter ? (
                                    <Button
                                      variant="secondary"
                                      onClick={() => {
                                        setSearch('')
                                        setStatusFilter('')
                                        setServicoFilterId('')
                                      }}
                                    >
                                      Limpar filtros
                                    </Button>
                                  ) : (
                                    <Button variant="secondary" onClick={() => void refresh()}>
                                      Recarregar
                                    </Button>
                                  )}
                                </div>
                              </div>
                            ) : (
                              [...visibleAgs.map((a) => ({ kind: 'ag' as const, time: a.hora_inicio, ag: a })), ...bls.map((b) => ({ kind: 'b' as const, time: b.hora_inicio, b }))]
                                .sort((x, y) => parseTimeToMinutes(x.time) - parseTimeToMinutes(y.time))
                                .map((item) => {
                                  if (item.kind === 'b') {
                                    return (
                                      <div key={`b:${item.b.id}`} className="px-3 py-2 flex items-start justify-between gap-2">
                                        <div className="text-sm text-slate-700">
                                          <div className="font-semibold text-slate-900">
                                            {item.b.hora_inicio}–{item.b.hora_fim}
```

```
                     </div>
                     <div className="text-slate-600">🔒 Bloqueado {item.b.motivo ? `•
${item.b.motivo}` : ''}</div>
                    </div>
                    <Badge tone="yellow">Bloqueado</Badge>
                   </div>
                  )
                 }
                 const a = item.ag
                 const visible = true
                 const statusUi = resolveStatusUi(a.status)
                 return (
                  <div key={`ag:${a.id}`} className="px-3 py-2 flex items-start justify-between gap-
2">
                   <div className="text-sm text-slate-700">
                    <div className="font-semibold text-slate-900 flex items-center gap-2">
                     {a.servico?.cor ? (
                      <span className="h-2.5 w-2.5 rounded-full" style={{ backgroundColor:
a.servico.cor }} />
                     ) : null}
                     <span>
                      {a.hora_inicio} • {visible ? a.cliente_nome : 'Ocupado'}
                     </span>
                    </div>
                    {visible ? <div className="text-slate-600">{a.servico?.nome ?? 'Serviço'}</div>
: null}
                    {visible ? (() => {
                     const endereco = readExtrasEndereco(a.extras)
                     return endereco ? <div className="text-slate-600">📍 {endereco}</div> : null
                    })() : null}
                   </div>
                   <Badge tone={statusUi.tone}>{statusUi.label}</Badge>
                  </div>
                 )
                })
              )}
             </div>
            </div>
           )
          })}
         </div>
        </div>
       </div>
      )}
        </div>
       </Card>
      </div>

      {funcionario.permissao !== 'atendente' ? (
       <Card>
        <div className="p-6 flex items-center justify-between">
         <div>
          <div className="text-sm font-semibold text-slate-900">Resumo do Dia</div>
          <div className="text-sm text-slate-600">
           {agendamentos.filter((a) => a.status !== 'cancelado').length} agendamentos
          </div>
         </div>
         <div className="text-lg font-semibold text-slate-900">{funcionario.pode_ver_financeiro ?
formatBRMoney(totalDia) : '—'}</div>
        </div>
       </Card>
      ) : null}

      <TutorialOverlay
       open={tutorialOpen}
       steps={tutorialSteps}
       step={tutorialStep}
       onStepChange={setTutorialStep}
       onClose={closeTutorial}
       titleFallback="Agenda"
      />
     </div>
```

```
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/FuncionariosPage.tsx

```
import { useCallback, useEffect, useMemo, useState } from 'react'
import { AppShell } from '../../components/layout/AppShell'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { toISODate } from '../../lib/dates'
import { supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type Funcionario = {
  id: string
  usuario_master_id: string
  nome_completo: string
  email: string
  telefone: string | null
  permissao: 'admin' | 'funcionario' | 'atendente'
  pode_ver_agenda: boolean
  pode_criar_agendamentos: boolean
  pode_cancelar_agendamentos: boolean
  pode_bloquear_horarios: boolean
  pode_ver_financeiro: boolean
  pode_gerenciar_servicos: boolean
  pode_ver_clientes_de_outros: boolean
  horario_inicio: string | null
  horario_fim: string | null
  dias_trabalho: number[] | null
  intervalo_inicio: string | null
  intervalo_fim: string | null
  capacidade_dia_inteiro?: number
  ativo: boolean
}

type PermissionKey =
  | 'pode_ver_agenda'
  | 'pode_criar_agendamentos'
  | 'pode_cancelar_agendamentos'
  | 'pode_bloquear_horarios'
  | 'pode_ver_financeiro'
  | 'pode_gerenciar_servicos'
  | 'pode_ver_clientes_de_outros'

const weekdayOptions = [
  { value: 1, label: 'S' },
  { value: 2, label: 'T' },
  { value: 3, label: 'Q' },
  { value: 4, label: 'Q' },
  { value: 5, label: 'S' },
  { value: 6, label: 'S' },
  { value: 0, label: 'D' },
]

function daysLabel(days: number[] | null) {
  if (!days || days.length === 0) return '—'
  const map = new Map(weekdayOptions.map((d) => [d.value, d.label]))
  return days
    .slice()
    .sort((a, b) => a - b)
    .map((d) => map.get(d) ?? String(d))
    .join('')
}

type FormState = {
  id?: string
  nome_completo: string
  email: string
  senha: string
  senha_confirm: string
  telefone: string
```

```
  permissao: 'admin' | 'funcionario' | 'atendente'
  horario_inicio: string
  horario_fim: string
  dias_trabalho: number[]
  intervalo_inicio: string
  intervalo_fim: string
  capacidade_dia_inteiro: string
  pode_ver_agenda: boolean
  pode_criar_agendamentos: boolean
  pode_cancelar_agendamentos: boolean
  pode_bloquear_horarios: boolean
  pode_ver_financeiro: boolean
  pode_gerenciar_servicos: boolean
  pode_ver_clientes_de_outros: boolean
  ativo: boolean
}

const permissionFields: Array<{ key: PermissionKey; label: string }> = [
  { key: 'pode_ver_agenda', label: 'Ver agenda' },
  { key: 'pode_criar_agendamentos', label: 'Criar agendamentos' },
  { key: 'pode_cancelar_agendamentos', label: 'Cancelar agendamentos' },
  { key: 'pode_bloquear_horarios', label: 'Bloquear horários próprios' },
  { key: 'pode_ver_financeiro', label: 'Ver financeiro' },
  { key: 'pode_gerenciar_servicos', label: 'Gerenciar serviços' },
  { key: 'pode_ver_clientes_de_outros', label: 'Ver clientes de outros' },
]

function toFormState(f?: Funcionario | null): FormState {
  return {
    id: f?.id,
    nome_completo: f?.nome_completo ?? '',
    email: f?.email ?? '',
    senha: '',
    senha_confirm: '',
    telefone: f?.telefone ?? '',
    permissao: f?.permissao ?? 'funcionario',
    horario_inicio: f?.horario_inicio ?? '09:00',
    horario_fim: f?.horario_fim ?? '18:00',
    dias_trabalho: f?.dias_trabalho ?? [1, 2, 3, 4, 5],
    intervalo_inicio: f?.intervalo_inicio ?? '',
    intervalo_fim: f?.intervalo_fim ?? '',
    capacidade_dia_inteiro: String(typeof f?.capacidade_dia_inteiro === 'number' ? f.capacidade_dia_inteiro : 1),
    pode_ver_agenda: f?.pode_ver_agenda ?? true,
    pode_criar_agendamentos: f?.pode_criar_agendamentos ?? true,
    pode_cancelar_agendamentos: f?.pode_cancelar_agendamentos ?? true,
    pode_bloquear_horarios: f?.pode_bloquear_horarios ?? true,
    pode_ver_financeiro: f?.pode_ver_financeiro ?? false,
    pode_gerenciar_servicos: f?.pode_gerenciar_servicos ?? false,
    pode_ver_clientes_de_outros: f?.pode_ver_clientes_de_outros ?? false,
    ativo: f?.ativo ?? true,
  }
}

export function FuncionariosPage() {
  const { appPrincipal } = useAuth()
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : null

  const canUseCapacidadeDiaInteiro = useMemo(() => {
    const t = String(usuario?.tipo_negocio ?? '').trim().toLowerCase()
    return t === 'faxina' || t === 'diarista'
  }, [usuario?.tipo_negocio])

  const agendamentosIcon = useMemo(() => {
    const t = String(usuario?.tipo_negocio ?? '').trim().toLowerCase()
    return t === 'barbearia' ? '✂️' : '📅'
  }, [usuario?.tipo_negocio])

  const usuarioId = usuario?.id

  const tutorialSteps = useMemo(
    () =>
      [
```

```
        {
          title: 'Adicionar funcionário',
          body: 'Use “+ Novo” para cadastrar sua equipe. Você pode criar perfis ativos/inativos.',
          target: 'create' as const,
        },
        {
          title: 'Permissões e horários',
          body: 'Defina o que cada pessoa pode fazer e o horário/dias de trabalho para organizar a agenda.',
          target: 'form' as const,
        },
        {
          title: 'Gerenciar equipe',
          body: 'Edite, ative/desative e acompanhe quantos agendamentos cada funcionário tem no mês.',
          target: 'list' as const,
        },
      ] as const,
    []
  )

  const [funcionarios, setFuncionarios] = useState<Funcionario[]>([])
  const [counts, setCounts] = useState<Record<string, number>>({})
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [formOpen, setFormOpen] = useState(false)
  const [form, setForm] = useState<FormState>(() => toFormState(null))
  const [saving, setSaving] = useState(false)

  const limiteFuncionariosEfetivo = useMemo<number | null>(() => {
    if (!usuario) return 1
    const plano = String(usuario.plano ?? '').trim().toLowerCase()
    if (plano === 'enterprise') return null
    if (plano === 'pro' || plano === 'team') {
      const raw = usuario.limite_funcionarios
      const n = typeof raw === 'number' && Number.isFinite(raw) && raw > 0 ? Math.floor(raw) : null
      const base = 4
      const max = 6
      if (!n || n < base) return base
      return Math.min(max, n)
    }
    return 1
  }, [usuario])

  const limiteAtingido = useMemo(() => {
    const limite = limiteFuncionariosEfetivo
    if (limite === null) return false
    return funcionarios.filter((f) => f.ativo).length >= limite
  }, [funcionarios, limiteFuncionariosEfetivo])

  const canSubmit = useMemo(() => {
    const baseOk = form.nome_completo.trim() && form.email.trim() && form.horario_inicio && form.horario_fim &&
form.dias_trabalho.length > 0
    if (!baseOk) return false
    if (!form.id) {
      if (form.senha.trim().length < 8) return false
      if (form.senha !== form.senha_confirm) return false
    }
    return true
  }, [form])

  const senhaInvalida = useMemo(() => {
    if (form.id) return null
    if (!form.senha.trim()) return 'Defina uma senha para o funcionário.'
    if (form.senha.trim().length < 8) return 'A senha deve ter no mínimo 8 caracteres.'
    if (form.senha !== form.senha_confirm) return 'As senhas não conferem.'
    return null
  }, [form.id, form.senha, form.senha_confirm])

  type AgendamentoRow = { funcionario_id: string | null; status: string }

  const load = useCallback(async () => {
    if (!usuarioId) return
    setLoading(true)
```

```
    setError(null)

    const selectWithoutCap =
'id,usuario_master_id,nome_completo,email,telefone,permissao,pode_ver_agenda,pode_criar_agendamentos,pode_cancelar_agendamentos,pode_bloquear_horarios,pode_ver_financeiro,pode_gerenciar_servicos,pode_ver_clientes_de_outros,horario_inicio,horario_fim,dias_trabalho,intervalo_inicio,intervalo_fim,ativo'
    const selectWithCap = `${selectWithoutCap},capacidade_dia_inteiro`

    const missingCapacidadeColumn = (message: string) => {
      const lower = message.toLowerCase()
      return lower.includes('capacidade_dia_inteiro') && lower.includes('does not exist')
    }

    const fetchList = async (select: string) => {
      const { data, error: err } = await supabase
        .from('funcionarios')
        .select(select)
        .eq('usuario_master_id', usuarioId)
        .order('criado_em', { ascending: true })
      return { data, err }
    }

    let { data, err } = await fetchList(selectWithCap)
    if (err && missingCapacidadeColumn(err.message)) {
      const fallback = await fetchList(selectWithoutCap)
      data = fallback.data
      err = fallback.err
      setError(
        'Configuração do Supabase incompleta: rode no SQL Editor: alter table public.funcionarios add column if not exists capacidade_dia_inteiro int not null default 1;'
      )
    }
    if (err) {
      setError(err.message)
      setLoading(false)
      return
    }

    const list = (data ?? []) as unknown as Funcionario[]
    setFuncionarios(list)

    const now = new Date()
    const start = new Date(now.getFullYear(), now.getMonth(), 1)
    const end = new Date(now.getFullYear(), now.getMonth() + 1, 0)
    const { data: agData, error: agErr } = await supabase
      .from('agendamentos')
      .select('funcionario_id,status')
      .eq('usuario_id', usuarioId)
      .gte('data', toISODate(start))
      .lte('data', toISODate(end))
      .limit(1000)
    if (agErr) {
      setError(agErr.message)
      setLoading(false)
      return
    }
    const map: Record<string, number> = {}
    for (const a of ((agData ?? []) as unknown as AgendamentoRow[])) {
      if (!a.funcionario_id) continue
      if (a.status === 'cancelado') continue
      map[a.funcionario_id] = (map[a.funcionario_id] ?? 0) + 1
    }
    setCounts(map)
    setLoading(false)
  }, [usuarioId])

  useEffect(() => {
    void (async () => {
      await Promise.resolve()
      await load()
    })().catch((e: unknown) => {
```

```
      setError(e instanceof Error ? e.message : 'Erro ao carregar funcionários')
      setLoading(false)
    })
  }, [load])

  if (!appPrincipal) {
    return (
      <AppShell>
        <div className="text-slate-700">Carregando…</div>
      </AppShell>
    )
  }

  if (!usuario) {
    return (
      <AppShell>
        <div className="text-slate-700">Acesso restrito.</div>
      </AppShell>
    )
  }

  const openCreate = () => {
    setForm(toFormState(null))
    setFormOpen(true)
  }

  const openEdit = (f: Funcionario) => {
    setForm(toFormState(f))
    setFormOpen(true)
  }

  const closeForm = () => {
    setFormOpen(false)
    setForm(toFormState(null))
  }

  const toggleDay = (day: number) => {
    setForm((p) => ({ ...p, dias_trabalho: p.dias_trabalho.includes(day) ? p.dias_trabalho.filter((d) => d !== day) :
[...p.dias_trabalho, day] }))
  }

  const setPermission = (key: PermissionKey, value: boolean) => {
    setForm((p) => ({ ...p, [key]: value }))
  }

  const submit = async () => {
    if (!canSubmit) return
    setSaving(true)
    setError(null)

    const capRaw = Number(form.capacidade_dia_inteiro)
    const cap = canUseCapacidadeDiaInteiro && Number.isFinite(capRaw) ? Math.max(1, Math.min(2, Math.floor(capRaw))) : 1

    const payload = {
      nome_completo: form.nome_completo.trim(),
      email: form.email.trim().toLowerCase(),
      telefone: form.telefone.trim() ? form.telefone.trim() : null,
      permissao: form.permissao,
      horario_inicio: form.horario_inicio || null,
      horario_fim: form.horario_fim || null,
      dias_trabalho: form.dias_trabalho,
      intervalo_inicio: form.intervalo_inicio.trim() ? form.intervalo_inicio : null,
      intervalo_fim: form.intervalo_fim.trim() ? form.intervalo_fim : null,
      capacidade_dia_inteiro: cap,
      pode_ver_agenda: form.pode_ver_agenda,
      pode_criar_agendamentos: form.pode_criar_agendamentos,
      pode_cancelar_agendamentos: form.pode_cancelar_agendamentos,
      pode_bloquear_horarios: form.pode_bloquear_horarios,
      pode_ver_financeiro: form.pode_ver_financeiro,
      pode_gerenciar_servicos: form.pode_gerenciar_servicos,
      pode_ver_clientes_de_outros: form.pode_ver_clientes_de_outros,
      ativo: form.ativo,
```

```
    }

    if (form.id) {
      const missingCapacidadeColumn = (message: string) => {
        const lower = message.toLowerCase()
        return lower.includes('capacidade_dia_inteiro') && lower.includes('does not exist')
      }

      const { error: err } = await supabase.from('funcionarios').update(payload).eq('id',
form.id).eq('usuario_master_id', usuario.id)
      if (err) {
        if (missingCapacidadeColumn(err.message)) {
          const { capacidade_dia_inteiro, ...payloadNoCap } = payload
          void capacidade_dia_inteiro
          const { error: retryErr } = await supabase
            .from('funcionarios')
            .update(payloadNoCap)
            .eq('id', form.id)
            .eq('usuario_master_id', usuario.id)
          if (!retryErr) {
            setError(
              'Seu Supabase ainda não tem a coluna capacidade_dia_inteiro. O funcionário foi salvo sem essa
configuração. Rode no SQL Editor: alter table public.funcionarios add column if not exists capacidade_dia_inteiro int
not null default 1;'
            )
          } else {
            setError(retryErr.message)
            setSaving(false)
            return
          }
        } else {
          setError(err.message)
          setSaving(false)
          return
        }
      }
    } else {
      if (limiteAtingido) {
        setError('Limite de funcionários atingido. Ajuste a quantidade de profissionais em Pagamento.')
        setSaving(false)
        return
      }
      if (!supabaseEnv.ok) {
        setError('Configuração do Supabase ausente (VITE_SUPABASE_URL / VITE_SUPABASE_ANON_KEY).')
        setSaving(false)
        return
      }

      const { data: sessionData } = await supabase.auth.getSession()
      let session = sessionData.session
      const now = Math.floor(Date.now() / 1000)
      const expiresAt = session?.expires_at ?? null

      if (session && expiresAt && expiresAt <= now + 30) {
        const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
        if (!refreshErr) session = refreshed.session
      }

      const accessToken=[REDACTED]
      if (!accessToken) {
        setError('Sessão expirada. Faça login novamente e tente criar o funcionário.')
        setSaving(false)
        return
      }

      const fnUrl = `${supabaseEnv.values.VITE_SUPABASE_URL}/functions/v1/create-funcionario`

      let res: Response
      try {
        res = await fetch(fnUrl, {
          method: 'POST',
          headers: {
```

```
            'Content-Type': 'application/json',
            apikey=[REDACTED]
            Authorization: `Bearer ${accessToken}`,
          },
          body: JSON.stringify({ ...payload, senha: form.senha }),
        })
      } catch (e: unknown) {
        const msg = e instanceof Error ? e.message : 'Falha de rede'
        setError(`Não foi possível conectar à função de criação de funcionário: ${msg}`)
        setSaving(false)
        return
      }

      const fnVersion = res.headers.get('x-smagenda-fn')

      const raw = await res.text().catch(() => null)
      const parsed = raw
        ? (() => {
            try {
              return JSON.parse(raw) as unknown
            } catch {
              return null
            }
          })()
        : null

      const serverMessage = (() => {
        if (!parsed || typeof parsed !== 'object') return null
        const p = parsed as Record<string, unknown>
        if (typeof p.message === 'string' && p.message.trim()) return p.message
        if (typeof p.error === 'string' && p.error.trim()) return p.error
        return null
      })()

      if (!res.ok) {
        const status = res.status
        if (status === 404) {
          setError('Função create-funcionario não encontrada no Supabase (deploy pendente).')
        } else if (status === 401) {
          if (!fnVersion && parsed && typeof parsed === 'object' && (parsed as Record<string, unknown>).message ===
'Invalid JWT' && (parsed as Record<string, unknown>).code === 401) {
            setError('A Edge Function "create-funcionario" está exigindo JWT no Supabase. Refaça o deploy com
verify_jwt=false e tente novamente.')
          } else {
            setError(serverMessage ? `Sem autorização: ${serverMessage}` : 'Sem autorização para chamar a função. Faça o
deploy com verify_jwt=false e tente novamente.')
          }
        } else if (status === 403) {
          setError(serverMessage ? `Sem permissão: ${serverMessage}` : 'Sem permissão para criar funcionário.')
        } else {
          setError(serverMessage ? `Erro ao criar funcionário (HTTP ${status}): ${serverMessage}` : `Erro ao criar
funcionário (HTTP ${status}).`)
        }
        setSaving(false)
        return
      }

      const funcionarioId = (parsed as { id?: string } | null)?.id
      if (!funcionarioId) {
        setError('Falha ao criar login do funcionário.')
        setSaving(false)
        return
      }
    }

    setSaving(false)
    closeForm()
    await load()
  }

  const toggleAtivo = async (f: Funcionario) => {
    setError(null)
```

```
    const { error: err } = await supabase
      .from('funcionarios')
      .update({ ativo: !f.ativo })
      .eq('id', f.id)
      .eq('usuario_master_id', usuario.id)
    if (err) {
      setError(err.message)
      return
    }
    setFuncionarios((prev) => prev.map((x) => (x.id === f.id ? { ...x, ativo: !x.ativo } : x)))
  }

  const remove = async (f: Funcionario) => {
    setError(null)
    const ok = window.confirm(`Excluir funcionário "${f.nome_completo}"?`)
    if (!ok) return
    const { error: err } = await supabase.from('funcionarios').delete().eq('id', f.id).eq('usuario_master_id',
usuario.id)
    if (err) {
      setError(err.message)
      return
    }
    await load()
  }

  return (
    <PageTutorial usuarioId={usuarioId} page="funcionarios">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="space-y-6">
            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'create'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl p-2 -m-2'
                  : ''
              }
            >
              <div className="flex items-center justify-between">
                <div>
                  <div className="text-sm font-semibold text-slate-500">Funcionários</div>
                  <div className="text-xl font-semibold text-slate-900">Gestão da equipe</div>
                </div>
                <div className="flex items-center gap-2">
                  <Button variant="secondary" onClick={resetTutorial}>
                    Rever tutorial
                  </Button>
                  <Button onClick={openCreate} disabled={limiteAtingido}>
                    + Novo
                  </Button>
                </div>
              </div>
            </div>

        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> :
null}

        {limiteAtingido ? (
          <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-800">Limite de
funcionários ativo atingido.</div>
        ) : null}

            {formOpen ? (
              <div
                className={
                  tutorialOpen && tutorialSteps[tutorialStep]?.target === 'form'
                    ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                    : ''
                }
              >
                <Card>
                  <div className="p-6 space-y-4">
              <div className="text-sm font-semibold text-slate-900">{form.id ? 'Editar funcionário' : 'Adicionar
```

```
funcionário'}</div>

              <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                <Input label="Nome completo" value={form.nome_completo} onChange={(e) => setForm((p) => ({ ...p,
nome_completo: e.target.value }))} />
                <Input label="Email" value={form.email} onChange={(e) => setForm((p) => ({ ...p, email: e.target.value
}))} />
              </div>

              {!form.id ? (
                <div className="space-y-2">
                  <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                    <Input
                      label="Senha"
                      type="password"
                      autoComplete="new-password"
                      value={form.senha}
                      onChange={(e) => setForm((p) => ({ ...p, senha: e.target.value }))}
                    />
                    <Input
                      label="Confirmar senha"
                      type="password"
                      autoComplete="new-password"
                      value={form.senha_confirm}
                      onChange={(e) => setForm((p) => ({ ...p, senha_confirm: e.target.value }))}
                    />
                  </div>
                  <div className={senhaInvalida ? 'text-xs text-rose-600' : 'text-xs text-slate-600'}>
                    {senhaInvalida ?? 'Essa senha será usada pelo funcionário para entrar no sistema.'}
                  </div>
                </div>
              ) : null}

              <Input label="Telefone" value={form.telefone} onChange={(e) => setForm((p) => ({ ...p, telefone:
e.target.value }))} />

              <div className="space-y-2">
                <div className="text-sm font-medium text-slate-700">Nível de acesso</div>
                <label className="flex items-center gap-2 text-sm text-slate-700">
                  <input type="radio" checked={form.permissao === 'admin'} onChange={() => setForm((p) => ({ ...p,
permissao: 'admin' }))} />
                  Gerente
                </label>
                <label className="flex items-center gap-2 text-sm text-slate-700">
                  <input
                    type="radio"
                    checked={form.permissao === 'funcionario'}
                    onChange={() => setForm((p) => ({ ...p, permissao: 'funcionario' }))}
                  />
                  Funcionário
                </label>
                <label className="flex items-center gap-2 text-sm text-slate-700">
                  <input
                    type="radio"
                    checked={form.permissao === 'atendente'}
                    onChange={() => setForm((p) => ({ ...p, permissao: 'atendente' }))}
                  />
                  Atendente
                </label>
              </div>

              <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                <Input label="Início" type="time" value={form.horario_inicio} onChange={(e) => setForm((p) => ({ ...p,
horario_inicio: e.target.value }))} />
                <Input label="Fim" type="time" value={form.horario_fim} onChange={(e) => setForm((p) => ({ ...p,
horario_fim: e.target.value }))} />
              </div>

              <div>
                <div className="text-sm font-medium text-slate-700 mb-2">Dias</div>
                <div className="flex flex-wrap gap-2">
                  {weekdayOptions.map((d) => (
```

```
                    <button
                      type="button"
                      key={d.value}
                      onClick={() => toggleDay(d.value)}
                      className={[
                        'h-9 w-9 rounded-lg border text-sm font-semibold',
                        form.dias_trabalho.includes(d.value)
                          ? 'bg-slate-900 text-white border-slate-900'
                          : 'bg-white text-slate-700 border-slate-200',
                      ].join(' ')}
                    >
                      {d.label}
                    </button>
                  ))}
                </div>
              </div>

              <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                <Input
                  label="Intervalo (opcional) início"
                  type="time"
                  value={form.intervalo_inicio}
                  onChange={(e) => setForm((p) => ({ ...p, intervalo_inicio: e.target.value }))}
                />
                <Input
                  label="Intervalo (opcional) fim"
                  type="time"
                  value={form.intervalo_fim}
                  onChange={(e) => setForm((p) => ({ ...p, intervalo_fim: e.target.value }))}
                />
              </div>

              {canUseCapacidadeDiaInteiro ? (
                <Input
                  label="Capacidade de diárias por dia (1 ou 2)"
                  type="number"
                  value={form.capacidade_dia_inteiro}
                  onChange={(e) => setForm((p) => ({ ...p, capacidade_dia_inteiro: e.target.value }))}
                />
              ) : null}

              <div className="space-y-2">
                <div className="text-sm font-medium text-slate-700">Permissões</div>
                {permissionFields.map(({ key, label }) => (
                  <label key={key} className="flex items-center gap-2 text-sm text-slate-700">
                    <input type="checkbox" checked={form[key]} onChange={(e) => setPermission(key, e.target.checked)} />
                    {label}
                  </label>
                ))}
              </div>

              <label className="flex items-center gap-2 text-sm text-slate-700">
                <input type="checkbox" checked={form.ativo} onChange={(e) => setForm((p) => ({ ...p, ativo:
e.target.checked }))} />
                Ativo
              </label>

              <div className="flex flex-wrap justify-end gap-2">
                <Button variant="secondary" onClick={closeForm} disabled={saving}>
                  Cancelar
                </Button>
                <Button onClick={submit} disabled={!canSubmit || saving}>
                  Salvar
                </Button>
              </div>

                </div>
                </Card>
              </div>
            ) : null}

            <div
```

```
       className={
        tutorialOpen && tutorialSteps[tutorialStep]?.target === 'list'
         ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
         : ''
       }
      >
       <Card>
        <div className="divide-y divide-slate-100">
         {loading ? (
          <div className="p-6 text-sm text-slate-600">Carregando funcionários…</div>
         ) : funcionarios.length === 0 ? (
          <div className="p-6 space-y-3">
           <div className="text-sm font-semibold text-slate-900">Nenhum funcionário cadastrado</div>
           <div className="text-sm text-slate-600">Cadastre sua equipe para distribuir os horários e organizar a agenda.</div>
           <div>
            <Button variant="secondary" onClick={openCreate} disabled={limiteAtingido}>
             + Adicionar primeiro funcionário
            </Button>
           </div>
          </div>
         ) : (
          funcionarios.map((f) => (
        <div key={f.id} className="p-4 space-y-3">
         <div className="flex flex-col gap-3 sm:flex-row sm:items-start sm:justify-between">
          <div>
           <div className="text-sm font-semibold text-slate-900">👤 {f.nome_completo}</div>
           <div className="text-sm text-slate-600">{f.email}</div>
           {f.telefone ? <div className="text-sm text-slate-600">📱 {f.telefone}</div> : null}
          </div>
          <div className="flex items-center gap-2 justify-end">
           {f.ativo ? <Badge tone="green">Ativo</Badge> : <Badge tone="red">Inativo</Badge>}
           <Badge tone="slate">{f.permissao === 'admin' ? 'Gerente' : f.permissao === 'atendente' ? 'Atendente' : 'Funcionário'}</Badge>
          </div>
         </div>

         <div className="grid grid-cols-1 gap-2 sm:grid-cols-3">
          <div className="text-sm text-slate-700">
           📅 Atende: {daysLabel(f.dias_trabalho)} {f.horario_inicio ?? '—'}–{f.horario_fim ?? '—'}
          </div>
          <div className="text-sm text-slate-700">
           {agendamentosIcon} {counts[f.id] ?? 0} agendamentos este mês
          </div>
          <div className="text-sm text-slate-700">
           Permissões: {f.pode_ver_agenda ? '✅' : '❌'} agenda • {f.pode_criar_agendamentos ? '✅' : '❌'} criar •{' '}
           {f.pode_ver_financeiro ? '✅' : '❌'} financeiro • {f.pode_gerenciar_servicos ? '✅' : '❌'} serviços
          </div>
         </div>

         <div className="flex flex-wrap justify-end gap-2">
          <Button variant="secondary" onClick={() => openEdit(f)}>
           Editar
          </Button>
          <Button variant="secondary" onClick={() => toggleAtivo(f)}>
           {f.ativo ? 'Desativar' : 'Ativar'}
          </Button>
          <Button variant="danger" onClick={() => remove(f)}>
           Excluir
          </Button>
         </div>
        </div>
          ))
         )}
        </div>
       </Card>
      </div>
     </div>

     <TutorialOverlay
```

```
            open={tutorialOpen}
            steps={tutorialSteps}
            step={tutorialStep}
            onStepChange={setTutorialStep}
            onClose={closeTutorial}
            titleFallback="Funcionários"
          />
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/MensagensSettingsPage.tsx

```tsx
import { useNavigate } from 'react-router-dom'
import { useEffect, useMemo, useRef, useState } from 'react'
import { AppShell } from '../../components/layout/AppShell'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { checkJwtProject, supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

const defaultConfirmacao = `Olá {nome}!\n\nSeu agendamento foi confirmado:\n📅 {data} às {hora}\n✂️ {servico}\n💰 {preco}\n\nLocal: {endereco}\n\nNos vemos em breve!\n{nome_negocio}`
const defaultLembrete = `Oi {nome}!\n\nLembrete: você tem agendamento em {data} às {hora}.\n\nSe não puder comparecer, me avise!\n{telefone_profissional}`
const defaultCancelamento = `Olá {nome}!\n\nSeu agendamento foi cancelado:\n📅 {data} às {hora}\n✂️ {servico}\n\nSe quiser remarcar, é só me chamar.\n{nome_negocio}`

type TemplatePreset = {
  key: string
  title: string
  confirmacao: string
  lembrete: string
  cancelamento: string
}

const templatePresets: TemplatePreset[] = [
  {
    key: 'padrao_servicos',
    title: 'Padrão (serviços)',
    confirmacao: defaultConfirmacao,
    lembrete: defaultLembrete,
    cancelamento: defaultCancelamento,
  },
  {
    key: 'lava_jatos',
    title: 'Lava-jato',
    confirmacao:
      `Olá {nome}!\n\n✅ Sua lavagem está confirmada:\n📅 {data} às {hora}\n🚗 {servico}\n💰 {preco}\n\nLocal: {unidade_nome}\n{unidade_endereco}\n\nDúvidas/alterações: {telefone_profissional}\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete da sua lavagem:\n📅 {data} às {hora}\n🚗 {servico}\n\nLocal: {unidade_nome}\n{unidade_endereco}\n\nSe precisar remarcar: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Sua lavagem foi cancelada:\n📅 {data} às {hora}\n🚗 {servico}\n\nSe quiser remarcar, fale com a gente: {telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'barbearia',
    title: 'Barbearia',
    confirmacao:
      `Olá {nome}!\n\n✅ Seu horário está confirmado:\n📅 {data} às {hora}\n✂️ {servico}\n💰 {preco}\n\nBarbeiro: {profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nAté já!\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete do seu horário:\n📅 {data} às {hora}\n✂️ {servico}\n\nBarbeiro: {profissional_nome}\n\nSe precisar remarcar: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Seu horário foi cancelado:\n📅 {data} às {hora}\n✂️ {servico}\n\nSe quiser remarcar, chama aqui: {telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'salao',
    title: 'Salão de beleza',
    confirmacao:
      `Olá {nome}!\n\n✅ Seu horário está confirmado:\n📅 {data} às {hora}\n💇 {servico}\n\nProfissional: {profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nAté já!\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete do seu horário:\n📅 {data} às {hora}\n💇 {servico}\nProfissional: {profissional_nome}\n\nSe precisar remarcar, fale com a gente: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Seu horário foi cancelado:\n📅 {data} às {hora}\n💇 {servico}\n\nSe quiser remarcar, fale com a gente: {telefone_profissional}\n{nome_negocio}`,
```

```
  },
  {
    key: 'estetica',
    title: 'Estética',
    confirmacao:
      `Olá {nome}!\n\n✅ Seu atendimento está confirmado:\n📅 {data} às {hora}\n✨ {servico}\n\nProfissional: {profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nAté breve!\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n✨ Lembrete do seu atendimento:\n📅 {data} às {hora}\n✨ {servico}\n\nQualquer ajuste: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Seu atendimento foi cancelado:\n📅 {data} às {hora}\n✨ {servico}\n\nSe quiser remarcar, é só me chamar: {telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'odontologia',
    title: 'Odontologia',
    confirmacao:
      `Olá {nome}!\n\n✅ Sua consulta está confirmada:\n📅 {data} às {hora}\n🦷 {servico}\n\nDentista: {profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nQualquer dúvida: {telefone_profissional}\n{nome_negocio}`,
    lembrete:
      `Olá {nome}!\n\n🦷 Lembrete da sua consulta:\n📅 {data} às {hora}\nProcedimento: {servico}\n\nSe precisar remarcar: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Sua consulta foi cancelada:\n📅 {data} às {hora}\n🦷 {servico}\n\nSe quiser remarcar: {telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'manicure',
    title: 'Manicure',
    confirmacao:
      `Olá {nome}!\n\n✅ Seu horário está confirmado:\n📅 {data} às {hora}\n💅 {servico}\n💰 {preco}\n\nProfissional: {profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nAté já!\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete do seu horário:\n📅 {data} às {hora}\n💅 {servico}\n\nSe precisar remarcar: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Seu horário foi cancelado:\n📅 {data} às {hora}\n💅 {servico}\n\nSe quiser remarcar: {telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'pilates',
    title: 'Pilates / Estúdio',
    confirmacao:
      `Olá {nome}!\n\n✅ Sua aula está confirmada:\n📅 {data} às {hora}\n🏋️‍♀️ {servico}\n\nInstrutor: {profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nAté lá!\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete da sua aula:\n📅 {data} às {hora}\n🏋️‍♀️ {servico}\n\nSe precisar remarcar: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Sua aula foi cancelada:\n📅 {data} às {hora}\n🏋️‍♀️ {servico}\n\nSe quiser remarcar: {telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'faxina',
    title: 'Faxina / Diarista',
    confirmacao:
      `Olá {nome}!\n\n✅ Sua diária está confirmada:\n📅 {data} às {hora}\n🧹 {servico}\n💰 {preco}\n\nEndereço do cliente:\n{cliente_endereco}\n\nProfissional: {profissional_nome}\n\nQualquer ajuste: {telefone_profissional}\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete da sua diária:\n📅 {data} às {hora}\n🧹 {servico}\n\nEndereço do cliente:\n{cliente_endereco}\n\nSe precisar remarcar: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Sua diária foi cancelada:\n📅 {data} às {hora}\n🧹 {servico}\n\nSe quiser remarcar, é só me chamar: {telefone_profissional}\n{nome_negocio}`,
  },
]

const templateVars: Array<{ key: string; label: string }> = [
  { key: 'nome', label: 'Cliente' },
```

```
  { key: 'data', label: 'Data' },
  { key: 'hora', label: 'Hora' },
  { key: 'servico', label: 'Serviço' },
  { key: 'preco', label: 'Preço' },
  { key: 'cliente_endereco', label: 'Endereço cliente' },
  { key: 'profissional_nome', label: 'Profissional' },
  { key: 'telefone_profissional', label: 'Telefone' },
  { key: 'unidade_nome', label: 'Unidade' },
  { key: 'unidade_endereco', label: 'Endereço unidade' },
  { key: 'unidade_telefone', label: 'Telefone unidade' },
  { key: 'endereco', label: 'Endereço (fallback)' },
  { key: 'nome_negocio', label: 'Nome do negócio' },
]

type AgendamentoConfirmadoRow = {
  id: string
  data: string
  hora_inicio: string | null
  cliente_nome: string | null
  cliente_telefone: string | null
  status?: string | null
  confirmacao_enviada?: boolean | null
}

async function sendConfirmacaoWhatsapp(agendamentoId: string) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } }
  }

  const supabaseUrl = supabaseEnv.values.VITE_SUPABASE_URL
  const supabaseAnonKey = supabaseEnv.values.VITE_SUPABASE_ANON_KEY

  const tryRefresh = async () => {
    const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
    if (refreshErr) return null
    return refreshed.session ?? null
  }

  const { data: sessionData } = await supabase.auth.getSession()
  let session = sessionData.session
  const now = Math.floor(Date.now() / 1000)
  const expiresAt = session?.expires_at ?? null

  if (session && expiresAt && expiresAt <= now + 60) {
    const refreshed = await tryRefresh()
    if (refreshed) session = refreshed
  }

  const token = session?.access_token ?? null
  if (!token) {
    return { ok: false as const, status: 401, body: { error: 'session_expired' } }
  }

  const tokenProject = checkJwtProject(token, supabaseUrl)
  if (!tokenProject.ok) {
    await supabase.auth.signOut().catch(() => undefined)
    return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: tokenProject.iss, expected:
tokenProject.expectedPrefix } }
  }

  const callFetch = async (jwt: string) => {
    const fnUrl = `${supabaseUrl}/functions/v1/whatsapp`
    let res: Response
    try {
      res = await fetch(fnUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          apikey=[REDACTED]
          Authorization: `Bearer ${jwt}`,
          'x-user-jwt': jwt,
        },
```

```
        body: JSON.stringify({ action: 'send_confirmacao', agendamento_id: agendamentoId }),
      })
    } catch (e: unknown) {
      const msg = e instanceof Error ? e.message : 'Falha de rede'
      return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
    }

    const fnVersion = res.headers.get('x-smagenda-fn')

    const text = await res.text()
    let parsed: unknown = null
    try {
      parsed = text ? JSON.parse(text) : null
    } catch {
      parsed = text
    }

    if (!res.ok && res.status === 401 && !fnVersion) {
      if (
        parsed &&
        typeof parsed === 'object' &&
        (parsed as Record<string, unknown>).message === 'Invalid JWT' &&
        (parsed as Record<string, unknown>).code === 401
      ) {
        return { ok: false as const, status: 401, body: { error: 'supabase_gateway_invalid_jwt' } }
      }
    }

    if (!res.ok) return { ok: false as const, status: res.status, body: parsed }
    return { ok: true as const, status: res.status, body: parsed }
  }

  const isInvalidJwtPayload = (payload: unknown) => {
    if (typeof payload === 'string') return payload.includes('Invalid JWT')
    if (!payload || typeof payload !== 'object') return false
    const obj = payload as Record<string, unknown>
    return obj.message === 'Invalid JWT' || obj.error === 'invalid_jwt'
  }

  const first = await callFetch(token)
  if (
    !first.ok &&
    first.status === 401 &&
    typeof first.body === 'object' &&
    first.body !== null &&
    (first.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
  ) {
    return first
  }

  if (!first.ok && first.status === 401 && isInvalidJwtPayload(first.body)) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    return callFetch(nextToken)
  }

  return first
}

export function MensagensSettingsPage() {
  const { appPrincipal } = useAuth()
  const navigate = useNavigate()
  const usuarioId = appPrincipal?.kind === 'usuario' ? appPrincipal.profile.id : null
  const canEditTemplates = useMemo(() => appPrincipal?.kind === 'usuario' && Boolean(usuarioId), [appPrincipal?.kind,
usuarioId])
  const canTestConfirmacao = Boolean(usuarioId)

  const tutorialSteps = useMemo(
    () =>
      [
        {
```

```
          title: 'Ative as automações',
          body: 'Ligue/desligue confirmação e lembrete automático. Ajuste as horas do lembrete conforme seu fluxo.',
          target: 'prefs' as const,
        },
        {
          title: 'Edite os templates',
          body: 'Personalize as mensagens e use variáveis como {nome}, {data} e {hora}.',
          target: 'templates' as const,
        },
        {
          title: 'Teste com um agendamento',
          body: 'Selecione um agendamento e envie uma confirmação para validar o WhatsApp em produção.',
          target: 'test' as const,
        },
      ] as const,
    []
  )

  const [mensagemConfirmacao, setMensagemConfirmacao] = useState(defaultConfirmacao)
  const [mensagemLembrete, setMensagemLembrete] = useState(defaultLembrete)
  const [mensagemCancelamento, setMensagemCancelamento] = useState(defaultCancelamento)
  const [presetKey, setPresetKey] = useState(templatePresets[0]?.key ?? 'padrao_servicos')
  const [lastField, setLastField] = useState<'confirmacao' | 'lembrete' | 'cancelamento'>('confirmacao')
  const confirmacaoRef = useRef<HTMLTextAreaElement | null>(null)
  const lembreteRef = useRef<HTMLTextAreaElement | null>(null)
  const cancelamentoRef = useRef<HTMLTextAreaElement | null>(null)
  const [enviarConfirmacao, setEnviarConfirmacao] = useState(true)
  const [enviarLembrete, setEnviarLembrete] = useState(false)
  const [enviarCancelamento, setEnviarCancelamento] = useState(true)
  const [lembreteHorasAntes, setLembreteHorasAntes] = useState(24)
  const [loading, setLoading] = useState(true)
  const [saving, setSaving] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [saved, setSaved] = useState(false)
  const [schemaIncompleto, setSchemaIncompleto] = useState(false)

  const [agLoading, setAgLoading] = useState(false)
  const [agendamentosConfirmados, setAgendamentosConfirmados] = useState<AgendamentoConfirmadoRow[]>([])
  const [agendamentoSelecionadoId, setAgendamentoSelecionadoId] = useState('')
  const [sendingConfirmacao, setSendingConfirmacao] = useState(false)
  const [sendConfirmacaoResult, setSendConfirmacaoResult] = useState<string | null>(null)
  const [agError, setAgError] = useState<string | null>(null)

  const isMissingColumnError = (message: string) => message.toLowerCase().includes('does not exist') &&
message.toLowerCase().includes('column')

  useEffect(() => {
    const run = async () => {
      if (!usuarioId) {
        setLoading(false)
        return
      }
      setLoading(true)
      setError(null)
      setSchemaIncompleto(false)

      const baseSelect = 'mensagem_confirmacao,mensagem_lembrete,mensagem_cancelamento'
      const baseFallbackSelect = 'mensagem_confirmacao,mensagem_lembrete'
      const baseFirst = await supabase.from('usuarios').select(baseSelect).eq('id', usuarioId).maybeSingle()

      if (baseFirst.error) {
        if (!isMissingColumnError(baseFirst.error.message)) {
          setError(baseFirst.error.message)
          setLoading(false)
          return
        }
        const baseSecond = await supabase.from('usuarios').select(baseFallbackSelect).eq('id', usuarioId).maybeSingle()
        if (baseSecond.error) {
          setError(baseSecond.error.message)
          setLoading(false)
          return
        }
```

```
        const baseRow = (baseSecond.data ?? null) as unknown as {
          mensagem_confirmacao?: string | null
          mensagem_lembrete?: string | null
          mensagem_cancelamento?: string | null
        } | null
        setMensagemConfirmacao(baseRow?.mensagem_confirmacao ?? defaultConfirmacao)
        setMensagemLembrete(baseRow?.mensagem_lembrete ?? defaultLembrete)
        setMensagemCancelamento(defaultCancelamento)
      } else {
        const baseRow = (baseFirst.data ?? null) as unknown as {
          mensagem_confirmacao?: string | null
          mensagem_lembrete?: string | null
          mensagem_cancelamento?: string | null
        } | null
        setMensagemConfirmacao(baseRow?.mensagem_confirmacao ?? defaultConfirmacao)
        setMensagemLembrete(baseRow?.mensagem_lembrete ?? defaultLembrete)
        setMensagemCancelamento(baseRow?.mensagem_cancelamento ?? defaultCancelamento)
      }

      const { data: extraData, error: extraErr } = await supabase
        .from('usuarios')
        .select('enviar_confirmacao,enviar_lembrete,enviar_cancelamento,lembrete_horas_antes')
        .eq('id', usuarioId)
        .maybeSingle()

      if (extraErr) {
        if (isMissingColumnError(extraErr.message)) {
          setSchemaIncompleto(true)
          setEnviarConfirmacao(true)
          setEnviarLembrete(false)
          setEnviarCancelamento(true)
          setLembreteHorasAntes(24)
          setLoading(false)
          return
        }
        setError(extraErr.message)
        setLoading(false)
        return
      }
      const row =
        (extraData ?? null) as unknown as {
          enviar_confirmacao?: boolean | null
          enviar_lembrete?: boolean | null
          enviar_cancelamento?: boolean | null
          lembrete_horas_antes?: number | null
        } | null
      setEnviarConfirmacao(row?.enviar_confirmacao ?? true)
      setEnviarLembrete(row?.enviar_lembrete ?? false)
      setEnviarCancelamento(row?.enviar_cancelamento ?? true)
      setLembreteHorasAntes(typeof row?.lembrete_horas_antes === 'number' ? row?.lembrete_horas_antes : 24)
      setLoading(false)
    }
    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar mensagens')
      setLoading(false)
    })
  }, [usuarioId])

  useEffect(() => {
    const run = async () => {
      if (!usuarioId) return
      setAgLoading(true)
      setSendConfirmacaoResult(null)
      setAgError(null)

      const baseSelect = 'id,data,hora_inicio,cliente_nome,cliente_telefone,status,confirmacao_enviada'
      const fallbackSelect = 'id,data,hora_inicio,cliente_nome,cliente_telefone,status'
      const isMissingColumnError = (message: string) => message.toLowerCase().includes('does not exist') &&
message.toLowerCase().includes('column')

      const first = await supabase
        .from('agendamentos')
```

```
        .select(baseSelect)
        .eq('usuario_id', usuarioId)
        .order('data', { ascending: false })
        .order('hora_inicio', { ascending: false })
        .limit(30)

      if (first.error) {
        if (!isMissingColumnError(first.error.message)) {
          setAgendamentosConfirmados([])
          setAgError(first.error.message)
          setAgLoading(false)
          return
        }

        const second = await supabase
          .from('agendamentos')
          .select(fallbackSelect)
          .eq('usuario_id', usuarioId)
          .order('data', { ascending: false })
          .order('hora_inicio', { ascending: false })
          .limit(30)

        if (second.error) {
          setAgendamentosConfirmados([])
          setAgError(second.error.message)
          setAgLoading(false)
          return
        }

        const list = (second.data ?? []) as unknown as AgendamentoConfirmadoRow[]
        setAgendamentosConfirmados(list)
        setAgendamentoSelecionadoId((prev) => prev || (list[0]?.id ?? ''))
        setAgLoading(false)
        return
      }

      const list = (first.data ?? []) as unknown as AgendamentoConfirmadoRow[]
      setAgendamentosConfirmados(list)
      setAgendamentoSelecionadoId((prev) => prev || (list[0]?.id ?? ''))
      setAgLoading(false)
    }

    run().catch(() => {
      setAgError('Erro ao carregar agendamentos')
      setAgLoading(false)
    })
  }, [usuarioId])

  const selectedAgendamento = useMemo(() => {
    if (!agendamentoSelecionadoId) return null
    return agendamentosConfirmados.find((a) => a.id === agendamentoSelecionadoId) ?? null
  }, [agendamentoSelecionadoId, agendamentosConfirmados])

  const save = async () => {
    if (!usuarioId) return
    setSaving(true)
    setSaved(false)
    setError(null)

    const baseUpdate = { mensagem_confirmacao: mensagemConfirmacao, mensagem_lembrete: mensagemLembrete,
mensagem_cancelamento: mensagemCancelamento }
    const first = await supabase.from('usuarios').update(baseUpdate).eq('id', usuarioId)

    if (first.error) {
      if (isMissingColumnError(first.error.message)) {
        const fallbackUpdate = { mensagem_confirmacao: mensagemConfirmacao, mensagem_lembrete: mensagemLembrete }
        const second = await supabase.from('usuarios').update(fallbackUpdate).eq('id', usuarioId)
        if (second.error) {
          setError(second.error.message)
          setSaving(false)
          return
        }
```

```
      } else {
        setError(first.error.message)
        setSaving(false)
        return
      }
    }

    const { error: extraErr } = await supabase
      .from('usuarios')
      .update({ enviar_confirmacao: enviarConfirmacao, enviar_lembrete: enviarLembrete, enviar_cancelamento: enviarCancelamento, lembrete_horas_antes: lembreteHorasAntes })
      .eq('id', usuarioId)

    if (extraErr) {
      if (isMissingColumnError(extraErr.message)) {
        setSchemaIncompleto(true)
        setSaving(false)
        setSaved(true)
        setTimeout(() => setSaved(false), 2000)
        return
      }
      setError(extraErr.message)
      setSaving(false)
      return
    }

    setSaving(false)
    setSaved(true)
    setTimeout(() => setSaved(false), 2000)
  }

  const getPreset = (key: string) => templatePresets.find((p) => p.key === key) ?? templatePresets[0]

  const applyPreset = (target: 'confirmacao' | 'lembrete' | 'cancelamento' | 'both' | 'all', keyOverride?: string) => {
    const preset = getPreset(keyOverride ?? presetKey)
    if (!preset) return
    if (target === 'confirmacao' || target === 'both' || target === 'all') setMensagemConfirmacao(preset.confirmacao)
    if (target === 'lembrete' || target === 'both' || target === 'all') setMensagemLembrete(preset.lembrete)
    if (target === 'cancelamento' || target === 'all') setMensagemCancelamento(preset.cancelamento)
  }

  const insertVar = (key: string) => {
    const token = `{${key}}`
    const active = lastField === 'confirmacao' ? confirmacaoRef.current : lastField === 'lembrete' ? lembreteRef.current : cancelamentoRef.current
    const getter = lastField === 'confirmacao' ? mensagemConfirmacao : lastField === 'lembrete' ? mensagemLembrete : mensagemCancelamento
    const setter =
      lastField === 'confirmacao' ? setMensagemConfirmacao : lastField === 'lembrete' ? setMensagemLembrete : setMensagemCancelamento
    if (!active) {
      setter(`${getter}${getter ? ' ' : ''}${token}`)
      return
    }

    const start = active.selectionStart ?? getter.length
    const end = active.selectionEnd ?? getter.length
    const next = `${getter.slice(0, start)}${token}${getter.slice(end)}`
    setter(next)
    requestAnimationFrame(() => {
      active.focus()
      const pos = start + token.length
      active.setSelectionRange(pos, pos)
    })
  }

  if (!usuarioId) {
    return (
      <AppShell>
        <div className="text-slate-700">{appPrincipal ? 'Acesso restrito.' : 'Carregando…'}</div>
      </AppShell>
    )
```

```
  }

  return (
    <PageTutorial usuarioId={usuarioId} page="mensagens">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="mx-auto w-full max-w-3xl space-y-6">
            <div className="flex flex-col items-start justify-between gap-3 sm:flex-row sm:items-center">
              <div>
                <div className="text-sm font-semibold text-slate-500">Configurações</div>
                <div className="text-xl font-semibold text-slate-900">Mensagens automáticas</div>
              </div>
              <Button variant="secondary" onClick={resetTutorial} className="w-full sm:w-auto">
                Rever tutorial
              </Button>
            </div>

            <div className="space-y-3">
              {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}
</div> : null}
              {schemaIncompleto ? (
                <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-800">
                  Seu Supabase ainda não tem as colunas de automação do WhatsApp. Execute o SQL do WhatsApp (automação)
no painel de Admin.
                </div>
              ) : null}
              {saved ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-
800">Salvo.</div> : null}
            </div>

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'prefs'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
              <Card>
                <div className="space-y-4 p-4 sm:p-6">
            <div className="text-sm font-semibold text-slate-900">Preferências de envio</div>
            {loading ? (
              <div className="text-sm text-slate-600">Carregando…</div>
            ) : (
              <div className="space-y-4">
                <label className="flex items-center gap-2 text-sm text-slate-700">
                  <input
                    type="checkbox"
                    checked={enviarConfirmacao}
                    onChange={(e) => setEnviarConfirmacao(e.target.checked)}
                    disabled={!canEditTemplates || saving}
                    className="h-4 w-4"
                  />
                  Enviar confirmação ao confirmar agendamento
                </label>
                <label className="flex items-center gap-2 text-sm text-slate-700">
                  <input
                    type="checkbox"
                    checked={enviarLembrete}
                    onChange={(e) => setEnviarLembrete(e.target.checked)}
                    disabled={!canEditTemplates || saving}
                    className="h-4 w-4"
                  />
                  Enviar lembrete automático
                </label>
                <div className="space-y-2">
                  <div className="text-sm font-medium text-slate-700">Enviar lembrete X horas antes</div>
                  <div className="flex items-center gap-3">
                    <input
                      type="range"
                      min={4}
                      max={72}
                      step={1}
```

```
                      value={lembreteHorasAntes}
                      onChange={(e) => setLembreteHorasAntes(Number(e.target.value))}
                      disabled={!canEditTemplates || saving || !enviarLembrete}
                      className="w-full"
                    />
                    <div className="w-14 text-right text-sm font-semibold text-slate-900">{lembreteHorasAntes}h</div>
                  </div>
                </div>
              </div>
            )}
                </div>
              </Card>
            </div>

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'test'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
              <Card>
                <div className="space-y-4 p-4 sm:p-6">
            <div className="text-sm font-semibold text-slate-900">Validar confirmação automática</div>
            <div className="text-sm text-slate-600">Salve o template antes de testar o envio.</div>

            {agLoading ? <div className="text-sm text-slate-600">Carregando agendamentos…</div> : null}

            {!agLoading && agError ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-
700">{agError}</div> : null}

            {!agLoading && agendamentosConfirmados.length === 0 ? (
              <div className="space-y-3">
                <div className="text-sm text-slate-600">Nenhum agendamento encontrado.</div>
                <div>
                  <Button variant="secondary" onClick={() => navigate('/dashboard')}>
                    Ir para Agenda
                  </Button>
                </div>
              </div>
            ) : null}

            {!agLoading && agendamentosConfirmados.length > 0 ? (
              <div className="space-y-2">
                <div className="text-sm font-medium text-slate-700">Agendamento</div>
                <select
                  className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                  value={agendamentoSelecionadoId}
                  onChange={(e) => setAgendamentoSelecionadoId(e.target.value)}
                  disabled={!canTestConfirmacao || saving || sendingConfirmacao}
                >
                  <option value="">Selecione…</option>
                  {agendamentosConfirmados.map((a) => {
                    const data = a.data ? a.data.split('-').reverse().join('/') : ''
                    const hora = a.hora_inicio ?? ''
                    const nome = a.cliente_nome ?? 'Cliente'
                    const tel = a.cliente_telefone ? ` • ${a.cliente_telefone}` : ''
                    const flag = a.confirmacao_enviada ? ' • já enviada' : ''
                    const status = (a.status ?? '').trim() ? ` • ${(a.status ?? '').trim()}` : ''
                    return (
                      <option key={a.id} value={a.id}>
                        {data} {hora} • {nome}
                        {tel}
                        {status}
                        {flag}
                      </option>
                    )
                  })}
                </select>
              </div>
            ) : null}
```

```
            {!agLoading && selectedAgendamento && (selectedAgendamento.status ?? '').trim().toLowerCase() !== 'confirmado' ? (
              <div className="text-sm text-amber-800 rounded-xl border border-amber-200 bg-amber-50 p-3">
                Esse agendamento ainda não está como “confirmado”. A confirmação automática só envia após confirmar.
              </div>
            ) : null}

            <div className="flex justify-end">
              <Button
                onClick={async () => {
                  if (!agendamentoSelecionadoId) return
                  if ((selectedAgendamento?.status ?? '').trim().toLowerCase() !== 'confirmado') return
                  setSendConfirmacaoResult(null)
                  setSendingConfirmacao(true)
                  const res = await sendConfirmacaoWhatsapp(agendamentoSelecionadoId)
                  if (!res.ok) {
                    if (
                      typeof res.body === 'object' &&
                      res.body !== null &&
                      (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
                    ) {
                      setSendConfirmacaoResult('JWT inválido para chamar a Edge Function. Saia e entre novamente. Se persistir, reimplante a função com verify_jwt=false (--no-verify-jwt).')
                      setSendingConfirmacao(false)
                      return
                    }
                    if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error === 'jwt_project_mismatch') {
                      setSendConfirmacaoResult('Sessão do Supabase pertence a outro projeto. Saia e entre novamente no sistema.')
                      setSendingConfirmacao(false)
                      return
                    }
                    if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error === 'invalid_jwt') {
                      setSendConfirmacaoResult('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
                      setSendingConfirmacao(false)
                      return
                    }
                    if (typeof res.body === 'object' && res.body !== null) {
                      const hint = (res.body as Record<string, unknown>).hint
                      if (typeof hint === 'string' && hint.trim()) {
                        setSendConfirmacaoResult(hint)
                        setSendingConfirmacao(false)
                        return
                      }
                      const code = (res.body as Record<string, unknown>).error
                      if (code === 'instance_not_connected') {
                        setSendConfirmacaoResult('WhatsApp não conectado. Vá em Configurações > WhatsApp e conecte a instância (QR Code).')
                        setSendingConfirmacao(false)
                        return
                      }
                    }
                    const details = typeof res.body === 'string' ? res.body : JSON.stringify(res.body)
                    setSendConfirmacaoResult(`Falha ao enviar confirmação (HTTP ${res.status}): ${details}`)
                    setSendingConfirmacao(false)
                    return
                  }

                  const details = typeof res.body === 'string' ? res.body : JSON.stringify(res.body)
                  setSendConfirmacaoResult(details || 'Enviado.')
                  setSendingConfirmacao(false)
                }}
                disabled={
                  !canTestConfirmacao ||
                  saving ||
                  loading ||
                  sendingConfirmacao ||
                  !agendamentoSelecionadoId ||
                  (selectedAgendamento?.status ?? '').trim().toLowerCase() !== 'confirmado'
```

```
                }
                className="w-full sm:w-auto"
              >
                {sendingConfirmacao ? 'Enviando…' : 'Enviar confirmação agora'}
              </Button>
            </div>

            {sendConfirmacaoResult ? (
              <pre className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-xs text-slate-800 overflow-auto
whitespace-pre-wrap">{sendConfirmacaoResult}</pre>
            ) : null}
                </div>
              </Card>
            </div>

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'templates'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
              <div className="space-y-6">
                <Card>
                  <div className="space-y-4 p-4 sm:p-6">
                    <div className="text-sm font-semibold text-slate-900">Modelos prontos</div>
                    <div className="text-sm text-slate-600">
                      Escolha um modelo e ajuste se quiser. As variáveis são preenchidas automaticamente com dados reais
do agendamento.
                    </div>

                    <div className="grid grid-cols-1 gap-3 sm:grid-cols-2">
                      <label className="block">
                        <div className="text-sm font-medium text-slate-700 mb-1">Modelo</div>
                        <select
                          className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                          value={presetKey}
                          onChange={(e) => {
                            const nextKey = e.target.value
                            setPresetKey(nextKey)
                            applyPreset('all', nextKey)
                          }}
                          disabled={!canEditTemplates || saving || loading}
                        >
                          {templatePresets.map((p) => (
                            <option key={p.key} value={p.key}>
                              {p.title}
                            </option>
                          ))}
                        </select>
                      </label>

                      <div className="grid grid-cols-2 gap-2 sm:self-end md:grid-cols-4">
                        <Button
                          variant="secondary"
                          onClick={() => applyPreset('confirmacao')}
                          disabled={!canEditTemplates || saving || loading}
                          className="w-full h-10"
                        >
                          Aplicar confirmação
                        </Button>
                        <Button
                          variant="secondary"
                          onClick={() => applyPreset('lembrete')}
                          disabled={!canEditTemplates || saving || loading}
                          className="w-full h-10"
                        >
                          Aplicar lembrete
                        </Button>
                        <Button
                          variant="secondary"
```

```
                          onClick={() => applyPreset('cancelamento')}
                          disabled={!canEditTemplates || saving || loading}
                          className="w-full h-10"
                        >
                          Aplicar cancelamento
                        </Button>
                        <Button
                          variant="secondary"
                          onClick={() => applyPreset('all')}
                          disabled={!canEditTemplates || saving || loading}
                          className="w-full h-10"
                        >
                          Aplicar tudo
                        </Button>
                      </div>
                    </div>
                  </div>
                </Card>

                <Card>
                  <div className="space-y-4 p-4 sm:p-6">
                    <div className="text-sm font-semibold text-slate-900">Variáveis</div>
                    <div className="text-sm text-slate-600">Clique para inserir no campo selecionado
(confirmação/lembrete/cancelamento).</div>

                    <div className="grid grid-cols-2 gap-2 sm:flex sm:flex-wrap">
                      {templateVars.map((v) => (
                        <Button
                          key={v.key}
                          variant="secondary"
                          onClick={() => insertVar(v.key)}
                          disabled={!canEditTemplates || saving || loading}
                          className="w-full sm:w-auto"
                        >
                          {v.label}
                        </Button>
                      ))}
                    </div>
                  </div>
                </Card>

                <Card>
                  <div className="space-y-4 p-4 sm:p-6">
                    <div className="text-sm font-semibold text-slate-900">Mensagem de confirmação</div>
                    {loading ? (
                      <div className="text-sm text-slate-600">Carregando…</div>
                    ) : (
                      <textarea
                        className="w-full min-h-[160px] rounded-lg border border-slate-200 bg-white px-3 py-2 text-base
text-slate-900 outline-none focus:ring-2 focus:ring-slate-300 sm:text-sm"
                        value={mensagemConfirmacao}
                        onChange={(e) => setMensagemConfirmacao(e.target.value)}
                        disabled={!canEditTemplates || saving}
                        ref={confirmacaoRef}
                        onFocus={() => setLastField('confirmacao')}
                      />
                    )}
                  </div>
                </Card>

                <Card>
                  <div className="space-y-4 p-4 sm:p-6">
                    <div className="text-sm font-semibold text-slate-900">Mensagem de lembrete</div>
                    {loading ? (
                      <div className="text-sm text-slate-600">Carregando…</div>
                    ) : (
                      <textarea
                        className="w-full min-h-[160px] rounded-lg border border-slate-200 bg-white px-3 py-2 text-base
text-slate-900 outline-none focus:ring-2 focus:ring-slate-300 sm:text-sm"
                        value={mensagemLembrete}
                        onChange={(e) => setMensagemLembrete(e.target.value)}
                        disabled={!canEditTemplates || saving}
```

```
                        ref={lembreteRef}
                        onFocus={() => setLastField('lembrete')}
                      />
                    )}
                  </div>
                </Card>

                <Card>
                  <div className="space-y-4 p-4 sm:p-6">
                    <div className="text-sm font-semibold text-slate-900">Mensagem de cancelamento</div>
                    {loading ? (
                      <div className="text-sm text-slate-600">Carregando…</div>
                    ) : (
                      <textarea
                        className="w-full min-h-[160px] rounded-lg border border-slate-200 bg-white px-3 py-2 text-base text-slate-900 outline-none focus:ring-2 focus:ring-slate-300 sm:text-sm"
                        value={mensagemCancelamento}
                        onChange={(e) => setMensagemCancelamento(e.target.value)}
                        disabled={!canEditTemplates || saving}
                        ref={cancelamentoRef}
                        onFocus={() => setLastField('cancelamento')}
                      />
                    )}
                  </div>
                </Card>

                {canEditTemplates ? (
                  <div className="flex justify-end">
                    <Button onClick={save} disabled={saving || loading} className="w-full sm:w-auto">
                      Salvar
                    </Button>
                  </div>
                ) : (
                  <div className="text-sm text-slate-600">Edição disponível apenas para a conta master.</div>
                )}
              </div>
            </div>
          </div>

          <TutorialOverlay
            open={tutorialOpen}
            steps={tutorialSteps}
            step={tutorialStep}
            onStepChange={setTutorialStep}
            onClose={closeTutorial}
            titleFallback="Mensagens"
          />
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/PagamentoPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { useLocation, useNavigate } from 'react-router-dom'
import { AppShell } from '../../components/layout/AppShell'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { checkJwtProject, supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type FnResult = { ok: true; status: number; body: unknown } | { ok: false; status: number; body: unknown }

function safeJson(value: unknown) {
  try {
    return JSON.stringify(value)
  } catch {
    return null
  }
}

async function callPaymentsFn(body: Record<string, unknown>): Promise<FnResult> {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } }
  }

  const supabaseUrl = String(supabaseEnv.values.VITE_SUPABASE_URL ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
    .replace(/\/+$/g, '')
  const supabaseAnonKey = String(supabaseEnv.values.VITE_SUPABASE_ANON_KEY ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
  const fnUrl = `${supabaseUrl}/functions/v1/payments`

  const tryRefresh = async () => {
    const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
    if (refreshErr) return null
    return refreshed.session ?? null
  }

  const { data: sessionData } = await supabase.auth.getSession()
  let session = sessionData.session
  const now = Math.floor(Date.now() / 1000)
  const expiresAt = session?.expires_at ?? null

  if (session && (!expiresAt || expiresAt <= now + 60)) {
    const refreshed = await tryRefresh()
    if (refreshed) session = refreshed
  }

  if (session) {
    const { error: userErr } = await supabase.auth.getUser()
    const userErrMsg = typeof userErr?.message === 'string' ? userErr.message : ''
    if (userErr && /invalid\s+jwt/i.test(userErrMsg)) {
      const refreshed = await tryRefresh()
      if (refreshed) session = refreshed
    }
  }

  const token = session?.access_token ?? null
  if (!token) {
    return { ok: false as const, status: 401, body: { error: 'session_expired' } }
  }

  const tokenProject = checkJwtProject(token, supabaseUrl)
  if (!tokenProject.ok) {
    await supabase.auth.signOut().catch(() => undefined)
    return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: tokenProject.iss, expected: tokenProject.expectedPrefix } }
```

```
  }

  const callFetch = async (jwt: string) => {
    let res: Response
    try {
      res = await fetch(fnUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          apikey=[REDACTED]
          Authorization: `Bearer ${jwt}`,
        },
        body: JSON.stringify(body),
      })
    } catch (e: unknown) {
      const msg = e instanceof Error ? e.message : 'Falha de rede'
      return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
    }

    const text = await res.text()
    let parsed: unknown = null
    try {
      parsed = text ? JSON.parse(text) : null
    } catch {
      parsed = text
    }

    if (!res.ok && res.status === 404) {
      const raw = typeof parsed === 'string' ? parsed : text
      if (typeof raw === 'string' && raw.includes('Requested function was not found')) {
        return {
          ok: false as const,
          status: 404,
          body: {
            error: 'function_not_deployed',
            message: 'A função payments não está publicada no Supabase. Faça deploy da Edge Function `payments` no seu
projeto.',
          },
        }
      }
    }

    if (
      !res.ok &&
      res.status === 401 &&
      parsed &&
      typeof parsed === 'object' &&
      (parsed as Record<string, unknown>).message === 'Invalid JWT' &&
      (parsed as Record<string, unknown>).code === 401
    ) {
      return { ok: false as const, status: 401, body: { error: 'supabase_gateway_invalid_jwt' } }
    }

    if (!res.ok) console.error('payments error', { status: res.status, body: parsed, body_json: safeJson(parsed) })

    if (!res.ok) return { ok: false as const, status: res.status, body: parsed }
    return { ok: true as const, status: res.status, body: parsed }
  }

  const first = await callFetch(token)
  if (!first.ok && first.status === 401) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) {
      await supabase.auth.signOut().catch(() => undefined)
      return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    }
    const nextProject = checkJwtProject(nextToken, supabaseUrl)
    if (!nextProject.ok) {
      await supabase.auth.signOut().catch(() => undefined)
      return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: nextProject.iss, expected:
nextProject.expectedPrefix } }
```

```
    }
    return callFetch(nextToken)
  }

  return first
}

async function createCheckoutPagamento(
  usuarioId: string,
  item: string,
  metodo: 'card' | 'pix',
  funcionariosTotal?: number | null
): Promise<FnResult> {
  const payload: Record<string, unknown> = { action: 'create_checkout', usuario_id: usuarioId, plano: item, metodo }
  if (typeof funcionariosTotal === 'number' && Number.isFinite(funcionariosTotal)) payload.funcionarios_total =
funcionariosTotal
  return callPaymentsFn(payload)
}

async function createBillingPortalPagamento(usuarioId: string): Promise<FnResult> {
  const payload: Record<string, unknown> = { action: 'create_billing_portal', usuario_id: usuarioId }
  return callPaymentsFn(payload)
}

async function syncCheckoutSessionPagamento(sessionId: string, usuarioId: string | null): Promise<FnResult> {
  const payload: Record<string, unknown> = { action: 'sync_checkout_session', session_id: sessionId }
  if (usuarioId) payload.usuario_id = usuarioId
  return callPaymentsFn(payload)
}

type PlanKey = 'free' | 'basic' | 'pro' | 'team' | 'enterprise'

type PlanCard = {
  key: PlanKey
  title: string
  priceLabel: string
  subtitle: string
  bullets: string[]
}

type ServiceCard = {
  key: 'setup_completo' | 'consultoria_hora'
  title: string
  priceLabel: string
  bullets: string[]
}

export function PagamentoPage() {
  const { appPrincipal, refresh } = useAuth()
  const location = useLocation()
  const navigate = useNavigate()
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : null
  const usuarioId = usuario?.id ?? null

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Escolha um plano',
          body: 'Selecione o plano ideal para seu negócio e veja o que está incluído.',
          target: 'plans' as const,
        },
        {
          title: 'Iniciar pagamento',
          body: 'Escolha PIX (30 dias) ou cartão (assinatura) para abrir o checkout em produção.',
          target: 'checkout' as const,
        },
        {
          title: 'Serviços avulsos',
          body: 'Use esta área para contratar setup completo ou consultoria, quando precisar.',
          target: 'services' as const,
        },
```

```
      ] as const,
    []
  )

  const [checkoutNotice, setCheckoutNotice] = useState<null | { kind: 'success' | 'cancel'; item: string | null }>(null)
  const [userSelectedPlan, setUserSelectedPlan] = useState<PlanKey | null>(null)
  const [selectedService, setSelectedService] = useState<ServiceCard['key'] | null>(null)
  const [creatingCheckout, setCreatingCheckout] = useState(false)
  const [openingPortal, setOpeningPortal] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [funcionariosTotal, setFuncionariosTotal] = useState<number | null>(null)

  const formatCheckoutError = (status: number, body: unknown) => {
    if (typeof body === 'string' && body.trim()) return body
    if (body && typeof body === 'object') {
      const obj = body as Record<string, unknown>
      const err = typeof obj.error === 'string' ? obj.error : null
      const message = typeof obj.message === 'string' && obj.message.trim() ? obj.message.trim() : null
      if (message) {
        const stripeStatus = typeof obj.stripe_status === 'number' && Number.isFinite(obj.stripe_status) ?
obj.stripe_status : null
        if (err === 'stripe_error') {
          const mode = typeof obj.stripe_key_mode === 'string' && obj.stripe_key_mode.trim() ?
obj.stripe_key_mode.trim() : 'unknown'
          if (stripeStatus) return `Stripe (${mode}, HTTP ${stripeStatus}): ${message}`
          return `Stripe (${mode}): ${message}`
        }
        return message
      }
      if (err === 'missing_supabase_env') return 'Configuração do Supabase ausente no ambiente.'
      if (err === 'invalid_action') {
        return 'A função payments do Supabase está desatualizada (não reconhece a ação solicitada). Faça deploy da Edge
Function payments e tente novamente.'
      }
      if (err === 'network_error') return typeof obj.message === 'string' && obj.message.trim() ? obj.message : 'Falha
de rede ao iniciar pagamento.'
      if (err === 'session_expired' || err === 'invalid_jwt') return 'Sessão expirada no Supabase. Saia e entre
novamente.'
      if (err === 'jwt_project_mismatch') return 'Sessão do Supabase pertence a outro projeto. Saia e entre novamente.'
      if (err === 'supabase_gateway_invalid_jwt') return 'A Edge Function está exigindo JWT no gateway. Faça deploy com
verify_jwt=false.'
      if (err === 'function_not_deployed') return 'A função payments não está publicada no Supabase.'
      const asJson = safeJson(body)
      if (err && asJson) return `Erro ao iniciar pagamento: ${err} (HTTP ${status}): ${asJson}`
      if (err) return `Erro ao iniciar pagamento: ${err} (HTTP ${status}).`
      if (asJson) return `Erro ao iniciar pagamento (HTTP ${status}): ${asJson}`
    }
    return `Erro ao iniciar pagamento (HTTP ${status}).`
  }

  const canShowFree = Boolean(usuario && usuario.plano === 'free' && usuario.status_pagamento === 'trial' &&
usuario.free_trial_consumido !== true)

  const plans = useMemo<PlanCard[]>(
    () =>
      [
        ...(canShowFree
          ? [
              {
                key: 'free' as const,
                title: 'FREE',
                priceLabel: 'R$ 0/mês',
                subtitle: 'Para testar',
                bullets: ['Até 30 agendamentos por mês', '1 profissional', 'Lembretes manuais (link do WhatsApp)',
'Suporte por email'],
              },
            ]
          : []),
        {
          key: 'basic',
          title: 'BASIC',
          priceLabel: 'R$ 34,99/mês',
```

```
          subtitle: '',
          bullets: ['Agendamentos 60 por mês', '1 profissional incluído', 'Lembretes automáticos via WhatsApp', 'Até 3
serviços', 'Página pública personalizável', 'Suporte por email'],
        },
        {
          key: 'pro',
          title: 'PRO',
          priceLabel: 'R$ 59,99/mês',
          subtitle: 'Até 6 profissionais (4 inclusos + até 2 adicionais)',
          bullets: ['4 profissionais incluídos', 'Serviços ilimitados', 'Logo e fotos de serviços', 'Relatórios',
'Bloqueios recorrentes', 'Suporte via WhatsApp'],
        },
        {
          key: 'enterprise',
          title: 'EMPRESA',
          priceLabel: 'R$ 98,99/mês',
          subtitle: 'Até 10 profissionais',
          bullets: ['Até 10 profissionais', 'Multi-unidades', 'Agendamentos ilimitados', 'Serviços ilimitados', 'Logo e
fotos de serviços', 'Para mais profissionais, fale com o suporte'],
        },
      ] satisfies PlanCard[],
    [canShowFree]
  )

  const services = useMemo<ServiceCard[]>(
    () =>
      [
        { key: 'setup_completo', title: 'Setup Completo', priceLabel: 'R$ 150 (uma vez)', bullets: ['Configuramos tudo
para você', 'Cadastramos serviços, fotos, horários', 'Testamos envios do WhatsApp apos conexão', 'Treinamos o cliente em
15 minutos'] },
        { key: 'consultoria_hora', title: 'Consultoria por Hora', priceLabel: 'R$ 80/hora', bullets: ['Ajuda com
configurações avançadas', 'Sugestões de otimização', 'Dúvidas gerais'] },
      ] satisfies ServiceCard[],
    []
  )

  const currentPlan = useMemo<PlanKey>(() => {
    const current = (usuario?.plano ?? '').trim().toLowerCase() as PlanKey
    if (current === 'free') return canShowFree ? 'free' : 'basic'
    if (current === 'enterprise') return 'enterprise'
    if (current === 'team') return 'pro'
    if (current === 'basic' || current === 'pro') return current
    return 'basic'
  }, [canShowFree, usuario?.plano])

  const selectedPlan = userSelectedPlan ?? currentPlan

  const defaultFuncionariosTotal = useMemo(() => {
    const raw = usuario?.limite_funcionarios
    const n = typeof raw === 'number' && Number.isFinite(raw) && raw > 0 ? Math.floor(raw) : 1
    return Math.max(1, Math.min(200, n))
  }, [usuario?.limite_funcionarios])

  const includedPro = 4
  const maxPro = 6
  const defaultProFuncionariosTotal = useMemo(() => {
    return Math.min(maxPro, Math.max(includedPro, defaultFuncionariosTotal))
  }, [defaultFuncionariosTotal, includedPro, maxPro])

  const effectiveFuncionariosTotal = funcionariosTotal ?? (selectedPlan === 'pro' ? defaultProFuncionariosTotal :
defaultFuncionariosTotal)

  useEffect(() => {
    const run = async () => {
      const search = location.search ?? ''
      if (!search) return
      const params = new URLSearchParams(search)
      const checkout = (params.get('checkout') ?? '').trim().toLowerCase()
      if (checkout !== 'success' && checkout !== 'cancel') return

      const item = (params.get('item') ?? params.get('plano') ?? '').trim().toLowerCase() || null
      setCheckoutNotice({ kind: checkout, item })
```

```
      const sessionId = (params.get('session_id') ?? '').trim()
      const usuarioIdFromParams = (params.get('usuario_id') ?? '').trim() || null

      const first = await refresh()
      const refreshedUsuarioId = first?.kind === 'usuario' ? first.profile.id : null
      const effectiveUsuarioId = refreshedUsuarioId ?? usuarioIdFromParams

      if (checkout === 'success') {
        if (sessionId) {
          const sync = await syncCheckoutSessionPagamento(sessionId, effectiveUsuarioId)
          if (!sync.ok) {
            setError(formatCheckoutError(sync.status, sync.body))
          } else {
            await refresh()
          }
        }

        let tries = 0
        while (tries < 10) {
          await new Promise((r) => setTimeout(r, 2000))
          const next = await refresh()
          if (next?.kind === 'usuario' && next.profile.status_pagamento === 'ativo') break
          tries += 1
        }
      }

      navigate('/pagamento', { replace: true })
    }
    run().catch(() => undefined)
  }, [location.search, navigate, refresh])

  const startPlanCheckout = async (metodo: 'card' | 'pix') => {
    if (!usuarioId) return
    const plan = selectedPlan
    if (!plan || plan === 'free') {
      setError('Selecione um plano válido.')
      return
    }

    const requested = plan === 'pro' ? Math.floor(effectiveFuncionariosTotal || includedPro) : 1
    if (plan === 'pro' && requested > maxPro) {
      setUserSelectedPlan('enterprise')
      setFuncionariosTotal(null)
      setError('Para mais de 6 profissionais, selecione o plano EMPRESA.')
      return
    }

    const total = plan === 'pro' ? Math.max(includedPro, Math.min(maxPro, requested)) : 1

    setCreatingCheckout(true)
    setError(null)
    const res = await createCheckoutPagamento(usuarioId, plan, metodo, total)
    if (!res.ok) {
      setError(formatCheckoutError(res.status, res.body))
      setCreatingCheckout(false)
      return
    }
    const body = res.body as Record<string, unknown>
    const url = typeof body.url === 'string' ? body.url : null
    if (!url) {
      setError('A função não retornou o link de checkout.')
      setCreatingCheckout(false)
      return
    }
    window.location.href = url
  }

  const openBillingPortal = async () => {
    if (!usuarioId) return
    setOpeningPortal(true)
    setError(null)
```

```
    const res = await createBillingPortalPagamento(usuarioId)
    if (!res.ok) {
      setError(formatCheckoutError(res.status, res.body))
      setOpeningPortal(false)
      return
    }
    const body = res.body as Record<string, unknown>
    const url = typeof body.url === 'string' ? body.url : null
    if (!url) {
      setError('A função não retornou o link do portal.')
      setOpeningPortal(false)
      return
    }
    window.location.href = url
  }

  const startServiceCheckout = async (metodo: 'card' | 'pix') => {
    if (!usuarioId || !selectedService) return
    setCreatingCheckout(true)
    setError(null)
    const res = await createCheckoutPagamento(usuarioId, selectedService, metodo)
    if (!res.ok) {
      setError(formatCheckoutError(res.status, res.body))
      setCreatingCheckout(false)
      return
    }
    const body = res.body as Record<string, unknown>
    const url = typeof body.url === 'string' ? body.url : null
    if (!url) {
      setError('A função não retornou o link de checkout.')
      setCreatingCheckout(false)
      return
    }
    window.location.href = url
  }

  const formatStatusPagamento = (value: string) => {
    const v = String(value ?? '').trim().toLowerCase()
    if (!v) return '—'
    if (v === 'ativo') return 'Ativo'
    if (v === 'trial') return 'Trial'
    if (v === 'inadimplente') return 'Inadimplente'
    if (v === 'suspenso') return 'Suspenso'
    if (v === 'cancelado') return 'Cancelado'
    return value
  }

  if (!usuarioId || !usuario) {
    return (
      <AppShell>
        <div className="text-slate-700">{appPrincipal ? 'Acesso restrito.' : 'Carregando…'}</div>
      </AppShell>
    )
  }

  const statusTone = usuario.status_pagamento === 'inadimplente' ? 'red' : usuario.status_pagamento === 'ativo' ?
'green' : 'slate'

  const planoLabel = (planoRaw: string) => {
    const p = String(planoRaw ?? '').trim().toLowerCase()
    if (p === 'enterprise') return 'EMPRESA'
    if (p === 'team') return 'PRO'
    if (p === 'pro') return 'PRO'
    if (p === 'basic') return 'BASIC'
    if (p === 'free') return 'FREE'
    return planoRaw
  }

  return (
    <PageTutorial usuarioId={usuarioId} page="pagamento">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
```

```
          <div className="space-y-6">
            <div className="flex items-center justify-between gap-3">
              <div>
                <div className="text-sm font-semibold text-slate-500">Configurações</div>
                <div className="text-xl font-semibold text-slate-900">Pagamento e planos</div>
              </div>
              <Button variant="secondary" onClick={resetTutorial}>
                Rever tutorial
              </Button>
            </div>

        {checkoutNotice ? (
          <div
            className={[
              'rounded-xl border p-4 text-sm',
              checkoutNotice.kind === 'success' ? 'border-emerald-200 bg-emerald-50 text-emerald-800' : 'border-amber-200 bg-amber-50 text-amber-900',
            ].join(' ')}
          >
            <div className="flex flex-wrap items-center justify-between gap-3">
              <div className="space-y-1">
                <div className="font-semibold">{checkoutNotice.kind === 'success' ? 'Pagamento concluído' : 'Pagamento cancelado'}</div>
                <div>
                  {checkoutNotice.item ? `Item: ${checkoutNotice.item.toUpperCase()}. ` : ''}Status: {formatStatusPagamento(usuario.status_pagamento)}.
                </div>
              </div>
              <div className="flex items-center gap-2">
                <Button variant="secondary" onClick={() => void refresh()}>
                  Atualizar
                </Button>
                <Button variant="secondary" onClick={() => setCheckoutNotice(null)}>
                  Fechar
                </Button>
              </div>
            </div>
          </div>
        ) : null}

        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'plans'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
              <Card>
                <div className="p-6 space-y-4">
                  <div className="flex flex-wrap items-start justify-between gap-3">
                    <div>
                      <div className="text-sm font-semibold text-slate-900">Pagamento</div>
                      <div className="text-sm text-slate-600">Plano atual: {planoLabel(usuario.plano)}</div>
                    </div>
                    <div className="flex items-center gap-2">
                      <Badge tone={usuario.ativo ? 'green' : 'red'}>{usuario.ativo ? 'Conta: Ativa' : 'Conta: Inativa'}</Badge>
                      <Badge tone={statusTone}>Pagamento: {formatStatusPagamento(usuario.status_pagamento)}</Badge>
                      <Button variant="secondary" onClick={() => void refresh()}>
                        Atualizar status
                      </Button>
                    </div>
                  </div>

                  <div className="grid grid-cols-1 gap-3 sm:grid-cols-2 lg:grid-cols-3">
                    {plans.map((p) => {
                      const selected = selectedPlan === p.key
                      const clickable = p.key !== 'free'
                      const best = p.key === 'pro'
```

```
                      return (
                        <button
                          key={p.key}
                          type="button"
                          onClick={() => {
                            if (!clickable) return
                            setUserSelectedPlan(p.key)
                            setFuncionariosTotal(null)
                          }}
                          className={[
                            'text-left rounded-xl border bg-white p-4 transition',
                            best
                              ? selected
                                ? 'border-slate-900 ring-2 ring-slate-900/10 bg-amber-50'
                                : 'border-amber-300 hover:bg-amber-50'
                              : selected
                                ? 'border-slate-900 ring-2 ring-slate-900/10'
                                : 'border-slate-200 hover:bg-slate-50',
                            clickable ? '' : 'cursor-default',
                          ].join(' ')}
                        >
                          <div className="flex items-start justify-between gap-3">
                            <div>
                              <div className="text-sm font-semibold text-slate-900">{p.title}</div>
                              <div className="text-xs text-slate-600">
                                {p.priceLabel}
                                {p.subtitle ? ` • ${p.subtitle}` : ''}
                              </div>
                            </div>
                            <div className="flex items-center gap-2">
                              {best ? <Badge tone="yellow">Melhor opção</Badge> : null}
                              {selected ? <Badge tone="slate">Selecionado</Badge> : null}
                            </div>
                          </div>
                          <div className="mt-3 space-y-1">
                            {p.bullets.map((b) => (
                              <div key={b} className="text-xs text-slate-700">
                                - {b}
                              </div>
                            ))}
                          </div>
                          <div className="mt-3 text-[11px] text-slate-500">Valores de pré-venda • válido até 08/02/2026.
</div>
                        </button>
                      )
                    })}
                  </div>

            {selectedPlan === 'pro' ? (
              <div className="grid grid-cols-1 gap-3 sm:grid-cols-2">
                <Input
                  label="Profissionais"
                  type="number"
                  min={includedPro}
                  max={maxPro}
                  value={String(effectiveFuncionariosTotal)}
                  onChange={(e) => {
                    const raw = e.target.value
                    const n = raw.trim() === '' ? includedPro : Number(raw)
                    if (!Number.isFinite(n)) return
                    const i = Math.floor(n)
                    if (i > maxPro) {
                      setUserSelectedPlan('enterprise')
                      setFuncionariosTotal(null)
                      setError('Para mais de 6 profissionais, selecione o plano EMPRESA.')
                      return
                    }
                    const clamped = Math.max(includedPro, Math.min(maxPro, i))
                    setFuncionariosTotal(clamped)
                  }}
                />
                <div className="rounded-xl border border-slate-200 bg-slate-50 p-4 text-xs text-slate-700 flex items-
```

```
center">
         O checkout calcula 4 profissionais inclusos + adicional por profissional acima de 4 (máximo 6 no PRO).
        </div>
       </div>
      ) : null}

         <div
          className={
           tutorialOpen && tutorialSteps[tutorialStep]?.target === 'checkout'
            ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl p-2 -m-2'
            : ''
          }
         >
          <div className="flex flex-wrap justify-end gap-2">
           <Button variant="secondary" onClick={() => void startPlanCheckout('pix')} disabled=
{creatingCheckout || selectedPlan === 'free'}>
            {creatingCheckout ? 'Abrindo…' : 'PIX (30 dias)'}
           </Button>
           <Button onClick={() => void startPlanCheckout('card')} disabled={creatingCheckout || selectedPlan
=== 'free'}>
            {creatingCheckout ? 'Abrindo…' : 'Cartão (assinatura)'}
           </Button>
          </div>
         </div>

         {usuario.stripe_customer_id ? (
          <div className="mt-4 rounded-xl border border-slate-200 bg-slate-50 p-4">
           <div className="flex flex-col gap-3 sm:flex-row sm:items-center sm:justify-between">
            <div>
             <div className="text-sm font-semibold text-slate-900">Assinatura</div>
             <div className="text-xs text-slate-600">Cancelar ou trocar forma de pagamento pelo Stripe.
</div>
            </div>
            <Button variant="secondary" onClick={() => void openBillingPortal()} disabled={openingPortal}>
             {openingPortal ? 'Abrindo…' : 'Gerenciar / Cancelar'}
            </Button>
           </div>
          </div>
         ) : usuario.status_pagamento === 'ativo' ? (
          <div className="mt-4 rounded-xl border border-slate-200 bg-slate-50 p-4 text-xs text-slate-700">
           Se você pagou por PIX (30 dias), não existe assinatura para cancelar. Ao vencer, basta não
renovar.
          </div>
         ) : null}
        </div>
       </Card>
      </div>

      <div
       className={
        tutorialOpen && tutorialSteps[tutorialStep]?.target === 'services'
         ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
         : ''
       }
      >
       <Card>
        <div className="p-6 space-y-4">
         <div>
          <div className="text-sm font-semibold text-slate-900">Serviços</div>
          <div className="text-sm text-slate-600">Contrate serviços avulsos para configuração e suporte.</div>
         </div>

      <div className="grid grid-cols-1 gap-3 sm:grid-cols-2">
       {services.map((s) => {
        const selected = selectedService === s.key
        return (
         <button
          key={s.key}
          type="button"
          onClick={() => setSelectedService(s.key)}
          className={[
           'text-left rounded-xl border bg-white p-4 transition',
```

```
                      selected ? 'border-slate-900 ring-2 ring-slate-900/10' : 'border-slate-200 hover:bg-slate-50',
                    ].join(' ')}
                  >
                    <div className="flex items-start justify-between gap-3">
                      <div>
                        <div className="text-sm font-semibold text-slate-900">{s.title}</div>
                        <div className="text-xs text-slate-600">{s.priceLabel}</div>
                      </div>
                      {selected ? <Badge tone="slate">Selecionado</Badge> : null}
                    </div>
                    <div className="mt-3 space-y-1">
                      {s.bullets.map((b) => (
                        <div key={b} className="text-xs text-slate-700">
                          - {b}
                        </div>
                      ))}
                    </div>
                  </button>
                )
              })}
            </div>

                  <div className="flex flex-wrap justify-end gap-2">
                    <Button variant="secondary" onClick={() => void startServiceCheckout('pix')} disabled=
{creatingCheckout || !selectedService}>
                      {creatingCheckout ? 'Abrindo…' : 'Pagar com PIX'}
                    </Button>
                    <Button onClick={() => void startServiceCheckout('card')} disabled={creatingCheckout ||
!selectedService}>
                      {creatingCheckout ? 'Abrindo…' : 'Pagar com cartão'}
                    </Button>
                  </div>
                </div>
              </Card>
            </div>

            <TutorialOverlay
              open={tutorialOpen}
              steps={tutorialSteps}
              step={tutorialStep}
              onStepChange={setTutorialStep}
              onClose={closeTutorial}
              titleFallback="Pagamento"
            />
          </div>
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/PaginaPublicaSettingsPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { AppShell } from '../../components/layout/AppShell'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { formatBRMoney } from '../../lib/dates'
import { slugify } from '../../lib/slug'
import { supabase } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

function isValidSlug(s: string) {
  if (!s.trim()) return false
  if (s.length < 2) return false
  if (s.length > 60) return false
  return /^[a-z0-9]+(?:-[a-z0-9]+)*$/.test(s)
}

function normalizeSlug(s: string) {
  return slugify(s).slice(0, 60)
}

export function PaginaPublicaSettingsPage() {
  const { appPrincipal, refresh } = useAuth()
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : null
  const usuarioId = usuario?.id ?? null
  const nomeNegocio = usuario?.nome_negocio ?? null

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Seu link público',
          body: 'Copie/abra sua URL de agendamento. Esse link é o que você divulga aos clientes.',
          target: 'link' as const,
        },
        {
          title: 'Personalizar aparência',
          body: 'Ajuste cores, logo e imagem de fundo para deixar a página com a cara do seu negócio.',
          target: 'appearance' as const,
        },
        {
          title: 'Salvar alterações',
          body: 'Finalize clicando em “Salvar”. Depois, abra o link para conferir o resultado.',
          target: 'save' as const,
        },
      ] as const,
    []
  )

  const [loading, setLoading] = useState(true)
  const [saving, setSaving] = useState(false)
  const [saved, setSaved] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [info, setInfo] = useState<string | null>(null)

  const [slug, setSlug] = useState('')
  const [logoUrl, setLogoUrl] = useState('')
  const [logoFile, setLogoFile] = useState<File | null>(null)
  const [instagramUrl, setInstagramUrl] = useState('')

  const [tipoNegocio, setTipoNegocio] = useState('')
  const [tipoNegocioInicial, setTipoNegocioInicial] = useState('')

  const [primaryColor, setPrimaryColor] = useState('#0f172a')
  const [backgroundColor, setBackgroundColor] = useState('#f8fafc')
  const [useBackgroundImage, setUseBackgroundImage] = useState(false)
  const [backgroundImageUrl, setBackgroundImageUrl] = useState('')
  const [bgFile, setBgFile] = useState<File | null>(null)
```

```
  type Unidade = {
    id: string
    nome: string
    slug: string
    endereco: string | null
    telefone: string | null
    ativo: boolean
  }

  const canUseMultiUnits = useMemo(() => {
    const p = String(usuario?.plano ?? '').trim().toLowerCase()
    return p === 'enterprise'
  }, [usuario?.plano])
  const [unidades, setUnidades] = useState<Unidade[]>([])
  const [unidadesLoading, setUnidadesLoading] = useState(false)
  const [savingUnidade, setSavingUnidade] = useState(false)
  const [unidadeNome, setUnidadeNome] = useState('')
  const [unidadeSlug, setUnidadeSlug] = useState('')
  const [unidadeEndereco, setUnidadeEndereco] = useState('')
  const [unidadeTelefone, setUnidadeTelefone] = useState('')

  type ServicoPreview = {
    id: string
    nome: string
    duracao_minutos: number
    preco: number
    cor: string | null
    foto_url: string | null
    ativo?: boolean | null
    ordem?: number | null
  }

  const [servicosPreview, setServicosPreview] = useState<ServicoPreview[]>([])
  const [servicosPreviewLoading, setServicosPreviewLoading] = useState(false)
  const [servicosPreviewError, setServicosPreviewError] = useState<string | null>(null)

  const logoObjectUrl = useMemo(() => (logoFile ? URL.createObjectURL(logoFile) : null), [logoFile])
  const bgObjectUrl = useMemo(() => (bgFile ? URL.createObjectURL(bgFile) : null), [bgFile])

  const canUseMedia = useMemo(() => {
    const p = String(usuario?.plano ?? '').trim().toLowerCase()
    return p === 'pro' || p === 'team' || p === 'enterprise'
  }, [usuario?.plano])

  useEffect(() => {
    if (!logoObjectUrl) return
    return () => {
      URL.revokeObjectURL(logoObjectUrl)
    }
  }, [logoObjectUrl])

  useEffect(() => {
    if (!bgObjectUrl) return
    return () => {
      URL.revokeObjectURL(bgObjectUrl)
    }
  }, [bgObjectUrl])

  const publicLink = useMemo(() => {
    if (!slug.trim()) return ''
    return `${window.location.origin}/agendar/${encodeURIComponent(slug.trim())}`
  }, [slug])

  const unidadeLink = useMemo(() => {
    return (unidadeSlugValue: string) => {
      if (!slug.trim() || !unidadeSlugValue.trim()) return ''
      return
`${window.location.origin}/agendar/${encodeURIComponent(slug.trim())}/${encodeURIComponent(unidadeSlugValue.trim())}`
    }
  }, [slug])

  useEffect(() => {
```

```
    const run = async () => {
      if (!usuarioId) {
        setLoading(false)
        return
      }
      setLoading(true)
      setError(null)
      setInfo(null)
      setSaved(false)
      const { data, error: err } = await supabase.from('usuarios').select('*').eq('id', usuarioId).maybeSingle()
      if (err) {
        setError(err.message)
        setLoading(false)
        return
      }
      const row = (data ?? {}) as Record<string, unknown>
      const nextSlug = typeof row.slug === 'string' ? row.slug : ''
      const nextLogo = typeof row.logo_url === 'string' ? row.logo_url : ''
      const nextInstagram = typeof row.instagram_url === 'string' ? row.instagram_url : ''
      const nextTipoNegocio = typeof row.tipo_negocio === 'string' ? row.tipo_negocio : ''
      const nextPrimary = typeof row.public_primary_color === 'string' ? row.public_primary_color : '#0f172a'
      const nextBgColor = typeof row.public_background_color === 'string' ? row.public_background_color : '#f8fafc'
      const nextUseBgImage = typeof row.public_use_background_image === 'boolean' ? row.public_use_background_image :
false
      const nextBgUrl = typeof row.public_background_image_url === 'string' ? row.public_background_image_url : ''

      setSlug(nextSlug)
      setLogoUrl(nextLogo)
      setInstagramUrl(nextInstagram)
      setTipoNegocio(nextTipoNegocio)
      setTipoNegocioInicial(nextTipoNegocio)
      setPrimaryColor(nextPrimary)
      setBackgroundColor(nextBgColor)
      setUseBackgroundImage(canUseMedia ? nextUseBgImage : false)
      setBackgroundImageUrl(nextBgUrl)
      setLoading(false)
    }
    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar')
      setLoading(false)
    })
  }, [usuarioId, canUseMedia])

  useEffect(() => {
    const run = async () => {
      if (!canUseMultiUnits || !usuarioId) {
        setUnidades([])
        return
      }
      setUnidadesLoading(true)
      const { data, error: err } = await supabase
        .from('unidades')
        .select('id,nome,slug,endereco,telefone,ativo')
        .eq('usuario_id', usuarioId)
        .order('criado_em', { ascending: true })

      if (err) {
        setError(err.message)
        setUnidadesLoading(false)
        return
      }
      setUnidades((data ?? []) as unknown as Unidade[])
      setUnidadesLoading(false)
    }
    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar unidades')
      setUnidadesLoading(false)
    })
  }, [canUseMultiUnits, usuarioId])

  useEffect(() => {
    const run = async () => {
```

```
      if (!usuarioId) {
        setServicosPreview([])
        return
      }
      setServicosPreviewLoading(true)
      setServicosPreviewError(null)

      const cols = 'id,nome,duracao_minutos,preco,cor,foto_url,ativo,ordem'
      const res = await supabase
        .from('servicos')
        .select(cols)
        .eq('usuario_id', usuarioId)
        .eq('ativo', true)
        .order('ordem', { ascending: true })
        .order('criado_em', { ascending: true })
        .limit(6)

      if (res.error) {
        setServicosPreview([])
        setServicosPreviewError(res.error.message)
        setServicosPreviewLoading(false)
        return
      }

      setServicosPreview((res.data ?? []) as unknown as ServicoPreview[])
      setServicosPreviewLoading(false)
    }

    run().catch((e: unknown) => {
      setServicosPreview([])
      setServicosPreviewError(e instanceof Error ? e.message : 'Erro ao carregar serviços')
      setServicosPreviewLoading(false)
    })
  }, [usuarioId])

  const uploadBackground = async (file: File) => {
    if (!usuarioId) throw new Error('Sessão inválida')
    const safeName = file.name
      .trim()
      .replace(/\s+/g, '-')
      .replace(/[^a-zA-Z0-9._-]/g, '')
      .slice(0, 80)
    const key = `${usuarioId}/bg-${Date.now()}-${safeName || 'bg'}`
    const { error: uploadErr } = await supabase.storage.from('logos').upload(key, file, {
      upsert: true,
      contentType: file.type || undefined,
    })
    if (uploadErr) throw uploadErr
    const { data } = supabase.storage.from('logos').getPublicUrl(key)
    const publicUrl = data?.publicUrl
    if (!publicUrl) throw new Error('Não foi possível obter a URL pública do fundo')
    return publicUrl
  }

  const uploadLogo = async (file: File) => {
    if (!usuarioId) throw new Error('Sessão inválida')
    const safeName = file.name
      .trim()
      .replace(/\s+/g, '-')
      .replace(/[^a-zA-Z0-9._-]/g, '')
      .slice(0, 80)
    const key = `${usuarioId}/logo-public-${Date.now()}-${safeName || 'logo'}`
    const { error: uploadErr } = await supabase.storage.from('logos').upload(key, file, {
      upsert: true,
      contentType: file.type || undefined,
    })
    if (uploadErr) throw uploadErr
    const { data } = supabase.storage.from('logos').getPublicUrl(key)
    const publicUrl = data?.publicUrl
    if (!publicUrl) throw new Error('Não foi possível obter a URL pública do logo')
    return publicUrl
  }
```

```
  const save = async () => {
    if (!usuarioId) return
    setSaving(true)
    setSaved(false)
    setError(null)
    setInfo(null)

    const lockedTipoBase = tipoNegocioInicial.trim()
    const nextTipo = tipoNegocio.trim()
    if (lockedTipoBase && nextTipo !== lockedTipoBase) {
      setError('Tipo de negócio já definido. Para alterar, solicite mudança no suporte.')
      setSaving(false)
      return
    }

    const slugSource = slug.trim() ? slug : ((nomeNegocio ?? '').trim() || `user-${usuarioId}`)
    const nextSlug = normalizeSlug(slugSource)
    if (!isValidSlug(nextSlug)) {
      setError('Slug inválido. Use letras minúsculas, números e hífen.')
      setSaving(false)
      return
    }

    let nextBgUrl: string | null = backgroundImageUrl.trim() ? backgroundImageUrl.trim() : null
    if (bgFile && !canUseMedia) {
      setError('Imagem de fundo disponível apenas no plano PRO ou EMPRESA.')
      setSaving(false)
      return
    }
    if (bgFile) {
      try {
        const url = await uploadBackground(bgFile)
        nextBgUrl = url
        setBackgroundImageUrl(url)
        setBgFile(null)
      } catch (e: unknown) {
        const msg = e instanceof Error ? e.message : 'Falha ao enviar imagem'
        const lower = msg.toLowerCase()
        const missingBucket = lower.includes('bucket') || lower.includes('not found')
        const rls = lower.includes('row-level security') || lower.includes('row level security')
        setError(
          missingBucket
            ? 'Configuração do Supabase incompleta: crie o bucket "logos" no Storage e habilite leitura pública + upload
do próprio usuário.'
            : rls
              ? 'Sem permissão para enviar ao Storage. Execute o SQL do Storage (logos) em /admin/configuracoes.'
              : msg
        )
        setSaving(false)
        return
      }
    }

    if (useBackgroundImage && !canUseMedia) {
      setError('Imagem de fundo disponível apenas no plano PRO ou EMPRESA.')
      setSaving(false)
      return
    }

    if (useBackgroundImage && !nextBgUrl) {
      setError('Envie uma imagem de fundo para ativar o fundo com imagem.')
      setSaving(false)
      return
    }

    let nextLogoUrl: string | null = logoUrl.trim() ? logoUrl.trim() : null
    if (logoFile && !canUseMedia) {
      setError('Logo disponível apenas no plano PRO ou EMPRESA.')
      setSaving(false)
      return
    }
```

```
    if (logoFile) {
      try {
        const url = await uploadLogo(logoFile)
        nextLogoUrl = url
        setLogoUrl(url)
        setLogoFile(null)
      } catch (e: unknown) {
        const msg = e instanceof Error ? e.message : 'Falha ao enviar logo'
        const lower = msg.toLowerCase()
        const missingBucket = lower.includes('bucket') || lower.includes('not found')
        const rls = lower.includes('row-level security') || lower.includes('row level security')
        setError(
          missingBucket
            ? 'Configuração do Supabase incompleta: crie o bucket "logos" no Storage e habilite leitura pública + upload do próprio usuário.'
            : rls
              ? 'Sem permissão para enviar ao Storage. Execute o SQL do Storage (logos) em /admin/configuracoes.'
              : msg
        )
        setSaving(false)
        return
      }
    }

    const payload: Record<string, unknown> = {
      slug: nextSlug,
      logo_url: nextLogoUrl,
      instagram_url: instagramUrl.trim() ? instagramUrl.trim() : null,
      tipo_negocio: tipoNegocio.trim() ? tipoNegocio.trim() : null,
      public_primary_color: primaryColor.trim() ? primaryColor.trim() : null,
      public_background_color: backgroundColor.trim() ? backgroundColor.trim() : null,
      public_use_background_image: canUseMedia ? useBackgroundImage : false,
      public_background_image_url: nextBgUrl,
    }

    const { error: updateErr } = await supabase.from('usuarios').update(payload).eq('id', usuarioId)
    if (updateErr) {
      const msg = updateErr.message
      const lower = msg.toLowerCase()
      const duplicate = lower.includes('duplicate') || lower.includes('unique')
      const missingColumn = (lower.includes('schema cache') && lower.includes('column')) || (lower.includes('does not exist') && lower.includes('column'))
      const rls = lower.includes('row-level security') || lower.includes('row level security')
      setError(
        duplicate
          ? 'Nome já está em uso. Escolha outro para o seu link público.'
          : missingColumn
            ? 'Configuração do Supabase incompleta: atualize o SQL do link público (listar + agendar).'
            : rls
              ? 'Sem permissão para atualizar seu perfil (RLS). Execute o SQL de políticas (Usuário / Funcionário) em /admin/configuracoes.'
              : msg
      )
      setSaving(false)
      return
    }

    await refresh()
    setSlug(nextSlug)
    if (!tipoNegocioInicial.trim() && nextTipo) setTipoNegocioInicial(nextTipo)
    setSaved(true)
    setSaving(false)
  }

  const createUnidade = async () => {
    if (!canUseMultiUnits || !usuarioId) return
    const nome = unidadeNome.trim()
    if (!nome) {
      setError('Informe o nome da unidade.')
      return
    }
    const slugRaw = (unidadeSlug.trim() || nome).trim()
```

```
    const nextSlug = normalizeSlug(slugRaw)
    if (!isValidSlug(nextSlug)) {
      setError('Slug da unidade inválido. Use letras minúsculas, números e hífen.')
      return
    }

    setSavingUnidade(true)
    setError(null)
    setInfo(null)

    const payload: Record<string, unknown> = {
      usuario_id: usuarioId,
      nome,
      slug: nextSlug,
      endereco: unidadeEndereco.trim() ? unidadeEndereco.trim() : null,
      telefone: unidadeTelefone.trim() ? unidadeTelefone.trim() : null,
    }

    const { error: insertErr } = await supabase.from('unidades').insert(payload)
    if (insertErr) {
      const msg = insertErr.message
      const lower = msg.toLowerCase()
      const duplicate = lower.includes('duplicate') || lower.includes('unique')
      setError(duplicate ? 'Já existe uma unidade com esse slug.' : msg)
      setSavingUnidade(false)
      return
    }

    setUnidadeNome('')
    setUnidadeSlug('')
    setUnidadeEndereco('')
    setUnidadeTelefone('')
    setInfo('Unidade criada.')
    setSavingUnidade(false)

    const { data, error: reloadErr } = await supabase
      .from('unidades')
      .select('id,nome,slug,endereco,telefone,ativo')
      .eq('usuario_id', usuarioId)
      .order('criado_em', { ascending: true })
    if (!reloadErr) setUnidades((data ?? []) as unknown as Unidade[])
  }

  const toggleUnidadeAtiva = async (id: string, ativo: boolean) => {
    if (!canUseMultiUnits || !usuarioId) return
    setSavingUnidade(true)
    setError(null)
    setInfo(null)
    const { error: updateErr } = await supabase.from('unidades').update({ ativo: !ativo }).eq('id', id).eq('usuario_id',
usuarioId)
    if (updateErr) {
      setError(updateErr.message)
      setSavingUnidade(false)
      return
    }
    setUnidades((prev) => prev.map((u) => (u.id === id ? { ...u, ativo: !ativo } : u)))
    setSavingUnidade(false)
  }

  if (!usuarioId) {
    return (
      <AppShell>
        <div className="text-slate-700">{appPrincipal ? 'Acesso restrito.' : 'Carregando…'}</div>
      </AppShell>
    )
  }

  return (
    <PageTutorial usuarioId={usuarioId} page="pagina_publica">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="mx-auto w-full max-w-3xl space-y-6">
```

```
            <div className="flex flex-col items-start justify-between gap-3 sm:flex-row sm:items-center">
              <div>
                <div className="text-sm font-semibold text-slate-500">Configurações</div>
                <div className="text-xl font-semibold text-slate-900">Página pública de agendamento</div>
              </div>
              <Button variant="secondary" onClick={resetTutorial} className="w-full sm:w-auto">
                Rever tutorial
              </Button>
            </div>

            {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}
            {info ? <div className="rounded-xl border border-slate-200 bg-white p-3 text-sm text-slate-700">{info}</div> : null}
            {saved ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">Salvo.</div> : null}

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'link'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
              <Card>
                  <div className="p-6 space-y-4">
                  <div className="text-sm font-semibold text-slate-900">Link público</div>

                  <label className="block">
                    <div className="text-sm font-medium text-slate-700 mb-1">Tipo de negócio</div>
                    <select
                      className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900 outline-none focus:ring-2 focus:ring-slate-300"
                      value={tipoNegocio}
                      onChange={(e) => setTipoNegocio(e.target.value)}
                      disabled={Boolean(tipoNegocioInicial.trim())}
                    >
                      <option value="">Geral</option>
                      <option value="lava_jatos">Lava-jato</option>
                      <option value="barbearia">Barbearia</option>
                      <option value="salao">Salão de beleza</option>
                      <option value="estetica">Estética</option>
                      <option value="odontologia">Odontologia</option>
                      <option value="manicure">Manicure</option>
                      <option value="pilates">Pilates / Estúdio</option>
                      <option value="faxina">Faxina / Diarista</option>
                    </select>
                    <div className="mt-1 text-xs text-slate-600">
                      {tipoNegocioInicial.trim()
                        ? 'Tipo de negócio bloqueado. Para alterar, solicite mudança no suporte.'
                        : 'Após salvar um tipo específico, ele fica bloqueado para evitar mudanças futuras.'}
                    </div>
                  </label>

                  <Input label="URL pública" value={publicLink} readOnly />

                  <Input
                    label="Instagram (opcional)"
                    value={instagramUrl}
                    onChange={(e) => setInstagramUrl(e.target.value)}
                    placeholder="@seuusuario ou https://instagram.com/seuusuario"
                  />

                  <div className="flex gap-3">
                    <Button
                      variant="secondary"
                      onClick={async () => {
                        if (!publicLink) return
                        await navigator.clipboard.writeText(publicLink)
                        setSaved(true)
                      }}
                      disabled={!publicLink}
```

```
                    >
                      Copiar link
                    </Button>
                    <Button
                      variant="secondary"
                      onClick={() => {
                        if (!publicLink) return
                        window.open(publicLink, '_blank', 'noopener,noreferrer')
                      }}
                      disabled={!publicLink}
                    >
                      Abrir
                    </Button>
                  </div>
                </div>
              </Card>
            </div>

        {canUseMultiUnits ? (
          <Card>
            <div className="p-6 space-y-4">
              <div>
                <div className="text-sm font-semibold text-slate-900">Unidades</div>
                <div className="text-sm text-slate-600">Crie filiais com link público próprio.</div>
              </div>

              <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                <Input label="Nome" value={unidadeNome} onChange={(e) => setUnidadeNome(e.target.value)}
placeholder="Unidade Centro" />
                <Input label="Slug" value={unidadeSlug} onChange={(e) => setUnidadeSlug(normalizeSlug(e.target.value))}
placeholder="centro" />
              </div>

              <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                <Input label="Endereço (opcional)" value={unidadeEndereco} onChange={(e) =>
setUnidadeEndereco(e.target.value)} placeholder="Rua X, 123" />
                <Input label="Telefone (opcional)" value={unidadeTelefone} onChange={(e) =>
setUnidadeTelefone(e.target.value)} placeholder="(11) 99999-9999" />
              </div>

              <div className="flex justify-end">
                <Button onClick={createUnidade} disabled={savingUnidade || unidadesLoading || loading || saving}>
                  {savingUnidade ? 'Salvando…' : 'Adicionar unidade'}
                </Button>
              </div>

              {unidadesLoading ? <div className="text-sm text-slate-600">Carregando unidades…</div> : null}
              {!unidadesLoading && unidades.length === 0 ? <div className="text-sm text-slate-600">Nenhuma unidade
cadastrada.</div> : null}

              {unidades.length > 0 ? (
                <div className="space-y-3">
                  {unidades.map((u) => {
                    const link = unidadeLink(u.slug)
                    return (
                      <div key={u.id} className="rounded-xl border border-slate-200 bg-white p-4 space-y-2">
                        <div className="flex flex-wrap items-start justify-between gap-3">
                          <div>
                            <div className="text-sm font-semibold text-slate-900">{u.nome}</div>
                            <div className="text-xs text-slate-600">/{u.slug}</div>
                          </div>
                          <div className="flex items-center gap-2">
                            <Button variant="secondary" onClick={() => toggleUnidadeAtiva(u.id, u.ativo)} disabled=
{savingUnidade}>
                              {u.ativo ? 'Desativar' : 'Ativar'}
                            </Button>
                          </div>
                        </div>

                        <Input label="URL da unidade" value={link} readOnly />

                        <div className="flex flex-wrap gap-3">
```

```
                          <Button
                            variant="secondary"
                            onClick={async () => {
                              if (!link) return
                              await navigator.clipboard.writeText(link)
                              setInfo('Link da unidade copiado.')
                            }}
                            disabled={!link}
                          >
                            Copiar link
                          </Button>
                          <Button
                            variant="secondary"
                            onClick={() => {
                              if (!link) return
                              window.open(link, '_blank', 'noopener,noreferrer')
                            }}
                            disabled={!link}
                          >
                            Abrir
                          </Button>
                        </div>
                      </div>
                    )
                  })}
                </div>
              ) : null}
            </div>
          </Card>
        ) : null}

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'appearance'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
          <Card>
                <div className="p-6 space-y-4">
                  <div className="text-sm font-semibold text-slate-900">Aparência</div>

                  {!canUseMedia ? (
                    <div className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-sm text-slate-700">
                      Logo e imagem de fundo estão disponíveis apenas no plano PRO ou EMPRESA. As cores ficam
disponíveis em qualquer plano.
                    </div>
                  ) : null}

            <div className="grid grid-cols-1 gap-4 sm:grid-cols-3">
              <Input label="Cor principal" type="color" value={primaryColor} onChange={(e) =>
setPrimaryColor(e.target.value)} />
              <Input
                label="Cor de fundo"
                type="color"
                value={backgroundColor}
                onChange={(e) => setBackgroundColor(e.target.value)}
                disabled={useBackgroundImage}
              />
              <div className="flex items-end">
                <button
                  type="button"
                  onClick={() => {
                    if (!canUseMedia) return
                    setUseBackgroundImage((v) => !v)
                  }}
                  disabled={!canUseMedia}
                  className={[
                    'h-10 w-full rounded-lg border px-3 text-sm font-medium',
                    useBackgroundImage ? 'border-slate-900 bg-slate-900 text-white' : 'border-slate-200 bg-white text-
slate-900',
                    !canUseMedia ? 'opacity-60 cursor-not-allowed' : '',
```

```
                  ].join(' ')}
                >
                  {useBackgroundImage ? 'Usar imagem de fundo: Sim' : 'Usar imagem de fundo: Não'}
                </button>
              </div>
            </div>

            <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
              <Input
                label="Ou enviar logo"
                type="file"
                accept="image/*"
                disabled={!canUseMedia}
                onChange={(e) => {
                  const file = e.target.files?.[0] ?? null
                  if (!file) {
                    setLogoFile(null)
                    return
                  }
                  if (!file.type.startsWith('image/')) {
                    setError('Selecione um arquivo de imagem (PNG/JPG/WebP).')
                    setLogoFile(null)
                    return
                  }
                  const maxBytes = 2 * 1024 * 1024
                  if (file.size > maxBytes) {
                    setError('Imagem muito grande (máx 2MB).')
                    setLogoFile(null)
                    return
                  }
                  setError(null)
                  setLogoFile(file)
                }}
              />
              {canUseMedia && (logoObjectUrl || logoUrl.trim()) && (
                <div className="flex items-end">
                  <div className="rounded-xl border border-slate-200 bg-white p-3">
                    <img src={logoObjectUrl ?? logoUrl.trim()} alt="Logo" className="h-14 w-14 rounded-xl object-cover
border border-slate-200" />
                  </div>
                </div>
              )}
            </div>

            <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
              <Input
                label="Enviar imagem de fundo"
                type="file"
                accept="image/*"
                disabled={!canUseMedia}
                onChange={(e) => {
                  const file = e.target.files?.[0] ?? null
                  if (!file) {
                    setBgFile(null)
                    return
                  }
                  if (!file.type.startsWith('image/')) {
                    setError('Selecione um arquivo de imagem (PNG/JPG/WebP).')
                    setBgFile(null)
                    return
                  }
                  const maxBytes = 3 * 1024 * 1024
                  if (file.size > maxBytes) {
                    setError('Imagem muito grande (máx 3MB).')
                    setBgFile(null)
                    return
                  }
                  setError(null)
                  setBgFile(file)
                }}
              />
              <div />
```

```
          </div>

          <div
            className="rounded-xl border border-slate-200 overflow-hidden"
            style={
              canUseMedia && useBackgroundImage && (bgObjectUrl || backgroundImageUrl.trim())
                ? {
                    backgroundColor,
                    backgroundImage: `url(${bgObjectUrl ?? backgroundImageUrl.trim()})`,
                    backgroundSize: 'cover',
                    backgroundPosition: 'center',
                  }
                : { backgroundColor }
            }
          >
            <div className="p-6" style={{ backgroundColor: canUseMedia && useBackgroundImage ?
'rgba(255,255,255,0.78)' : 'rgba(255,255,255,0.9)' }}>
              <div className="text-sm font-semibold text-slate-900">Prévia (com serviços reais)</div>
              <div className="mt-3 flex items-center gap-3">
                {(logoObjectUrl || logoUrl.trim()) && (
                  <img
                    src={logoObjectUrl ?? logoUrl.trim()}
                    alt="Logo"
                    className="h-12 w-12 rounded-xl object-cover border border-slate-200 bg-white"
                  />
                )}
                <div className="min-w-0">
                  <div className="truncate text-base font-semibold text-slate-900">{nomeNegocio ?? 'Seu negócio'}
</div>
                  <div className="truncate text-sm text-slate-600">Agendamento online</div>
                </div>
              </div>

              {instagramUrl.trim() ? (
                <div className="mt-3 text-sm">
                  <a
                    className="font-semibold"
                    href={instagramUrl.trim()}
                    target="_blank"
                    rel="noopener noreferrer"
                    style={{ color: primaryColor }}
                  >
                    Instagram
                  </a>
                </div>
              ) : null}

              <div className="mt-4">
                {servicosPreviewLoading ? <div className="text-sm text-slate-600">Carregando serviços…</div> : null}
                {!servicosPreviewLoading && servicosPreviewError ? (
                  <div className="text-sm text-rose-700">Não foi possível carregar seus serviços para a prévia.</div>
                ) : null}
                {!servicosPreviewLoading && !servicosPreviewError && servicosPreview.length === 0 ? (
                  <div className="text-sm text-slate-600">Cadastre ao menos 1 serviço para ver a prévia completa.
</div>
                ) : null}

                {!servicosPreviewLoading && !servicosPreviewError && servicosPreview.length > 0 ? (
                  <div className="mt-3 grid grid-cols-1 gap-2 sm:grid-cols-2">
                    {servicosPreview.slice(0, 4).map((s) => (
                      <div key={s.id} className="rounded-xl border border-slate-200 bg-white p-3">
                        <div className="flex items-start justify-between gap-3">
                          <div className="min-w-0">
                            <div className="truncate text-sm font-semibold text-slate-900">{s.nome}</div>
                            <div className="text-xs text-slate-600">
                              {s.duracao_minutos}min · {formatBRMoney(s.preco)}
                            </div>
                          </div>
                          <div
                            className="h-5 w-5 rounded-full border border-slate-200"
                            style={{ backgroundColor: (s.cor ?? '').trim() ? s.cor ?? undefined : '#e2e8f0' }}
                          />
```

```
                          </div>
                        </div>
                      ))}
                    </div>
                  ) : null}
                </div>

                <div className="mt-4">
                  <button
                    type="button"
                    className="h-10 w-full rounded-xl px-4 text-sm font-semibold text-white"
                    style={{ backgroundColor: primaryColor }}
                  >
                    Continuar
                  </button>
                </div>
              </div>
            </div>

                  <div
                    className={
                      tutorialOpen && tutorialSteps[tutorialStep]?.target === 'save'
                        ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl p-2 -m-2'
                        : ''
                    }
                  >
                    <div className="flex justify-end">
                      <Button onClick={save} disabled={saving || loading} style={{ backgroundColor: primaryColor,
borderColor: primaryColor }}>
                        {saving ? 'Salvando…' : 'Salvar'}
                      </Button>
                    </div>
                  </div>
                </div>
              </Card>
            </div>

            {loading ? <div className="text-sm text-slate-600">Carregando…</div> : null}

            <TutorialOverlay
              open={tutorialOpen}
              steps={tutorialSteps}
              step={tutorialStep}
              onStepChange={setTutorialStep}
              onClose={closeTutorial}
              titleFallback="Página pública"
            />
          </div>
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/RelatoriosPage.tsx

```
import { useCallback, useEffect, useMemo, useState } from 'react'
import { useNavigate } from 'react-router-dom'
import { AppShell } from '../../components/layout/AppShell'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { formatBRMoney, parseTimeToMinutes, toISODate } from '../../lib/dates'
import { supabase } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type AgendamentoRow = {
  id: string
  data: string
  hora_inicio: string
  status: string
  cliente_telefone: string | null
  servico: { id: string; nome: string; preco: number | null } | null
  funcionario: { id: string; nome_completo: string } | null
}

type ExportAgendamentoRow = {
  data: string
  hora_inicio: string
  hora_fim: string | null
  status: string
  cliente_nome: string | null
  cliente_telefone: string | null
  servico: { nome: string; preco: number | null } | null
  funcionario: { nome_completo: string } | null
}

function startOfMonth(d: Date) {
  const x = new Date(d)
  x.setDate(1)
  x.setHours(0, 0, 0, 0)
  return x
}

function endOfMonth(d: Date) {
  const x = new Date(d)
  x.setMonth(x.getMonth() + 1)
  x.setDate(0)
  x.setHours(0, 0, 0, 0)
  return x
}

function toCsv(rows: Array<Record<string, string | number | null | undefined>>) {
  const headers = Array.from(
    rows.reduce((set, r) => {
      for (const k of Object.keys(r)) set.add(k)
      return set
    }, new Set<string>())
  )
  const escape = (v: unknown) => {
    const s = v === null || v === undefined ? '' : String(v)
    if (/[\n\r",;]/.test(s)) return `"${s.replace(/"/g, '""')}"`
    return s
  }
  const lines = [headers.map(escape).join(';')]
  for (const r of rows) {
    lines.push(headers.map((h) => escape(r[h])).join(';'))
  }
  return `\ufeff${lines.join('\n')}`
}

export function RelatoriosPage() {
  const { appPrincipal, masterUsuario, masterUsuarioLoading } = useAuth()
  const funcionario = appPrincipal?.kind === 'funcionario' ? appPrincipal.profile : null
```

```
  const isGerente = appPrincipal?.kind === 'funcionario' && appPrincipal.profile.permissao === 'admin'
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : isGerente ? masterUsuario : null
  const usuarioId = usuario?.id ?? null
  const navigate = useNavigate()

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Escolha o período',
          body: 'Defina início e fim para calcular métricas e exportar apenas o que interessa.',
          target: 'filters' as const,
        },
        {
          title: 'Atualize e exporte',
          body: 'Clique em “Atualizar” para recarregar e use “Exportar CSV” para baixar seus dados reais.',
          target: 'filters' as const,
        },
        {
          title: 'Leia as métricas',
          body: 'No-show, receita prevista/realizada e novos vs recorrentes ajudam a tomar decisões.',
          target: 'cards' as const,
        },
      ] as const,
    []
  )

  const canUseRelatorios = useMemo(() => {
    const p = String(usuario?.plano ?? '').trim().toLowerCase()
    return p === 'pro' || p === 'team' || p === 'enterprise'
  }, [usuario?.plano])

  const today = useMemo(() => {
    const d = new Date()
    d.setHours(0, 0, 0, 0)
    return d
  }, [])

  const [start, setStart] = useState(() => toISODate(startOfMonth(today)))
  const [end, setEnd] = useState(() => toISODate(endOfMonth(today)))

  const [loading, setLoading] = useState(true)
  const [exporting, setExporting] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [agendamentos, setAgendamentos] = useState<AgendamentoRow[]>([])
  const [allUntilEnd, setAllUntilEnd] = useState<AgendamentoRow[]>([])

  const load = useCallback(async () => {
    if (!usuarioId) return
    setLoading(true)
    setError(null)

    const baseSelect =
'id,data,hora_inicio,status,cliente_telefone,servico:servico_id(id,nome,preco),funcionario:funcionario_id(id,nome_comple
to)'

    const { data: periodData, error: periodErr } = await supabase
      .from('agendamentos')
      .select(baseSelect)
      .eq('usuario_id', usuarioId)
      .gte('data', start)
      .lte('data', end)
      .order('data', { ascending: true })
      .order('hora_inicio', { ascending: true })
      .limit(10000)

    if (periodErr) {
      setError(periodErr.message)
      setLoading(false)
      return
    }
```

```
    const { data: untilEndData, error: untilEndErr } = await supabase
      .from('agendamentos')
      .select(baseSelect)
      .eq('usuario_id', usuarioId)
      .lte('data', end)
      .order('data', { ascending: true })
      .order('hora_inicio', { ascending: true })
      .limit(15000)

    if (untilEndErr) {
      setError(untilEndErr.message)
      setLoading(false)
      return
    }

    setAgendamentos((periodData ?? []) as unknown as AgendamentoRow[])
    setAllUntilEnd((untilEndData ?? []) as unknown as AgendamentoRow[])
    setLoading(false)
  }, [end, start, usuarioId])

  useEffect(() => {
    if (!canUseRelatorios) return
    void (async () => {
      await Promise.resolve()
      await load()
    })().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar relatórios')
      setLoading(false)
    })
  }, [canUseRelatorios, load])

  const metrics = useMemo(() => {
    const isCancelado = (s: string) => s.trim().toLowerCase() === 'cancelado'
    const isNoShow = (s: string) => {
      const x = s.trim().toLowerCase()
      return x === 'nao_compareceu' || x === 'não_compareceu' || x === 'no_show'
    }
    const isConcluido = (s: string) => {
      const x = s.trim().toLowerCase()
      return x === 'concluido' || x === 'concluído'
    }

    const valid = agendamentos.filter((a) => !isCancelado(a.status))
    const total = valid.length
    const noShows = valid.filter((a) => isNoShow(a.status)).length
    const concluidos = valid.filter((a) => isConcluido(a.status)).length
    const noShowRate = total > 0 ? (noShows / total) * 100 : 0

    const prevista = valid.reduce((sum, a) => sum + (a.servico?.preco ? Number(a.servico.preco) : 0), 0)
    const realizada = valid
      .filter((a) => isConcluido(a.status))
      .reduce((sum, a) => sum + (a.servico?.preco ? Number(a.servico.preco) : 0), 0)

    const byService = new Map<string, { nome: string; count: number }>()
    for (const a of valid) {
      if (!a.servico?.id) continue
      const prev = byService.get(a.servico.id)
      byService.set(a.servico.id, { nome: a.servico.nome, count: (prev?.count ?? 0) + 1 })
    }
    const topServices = Array.from(byService.values())
      .sort((x, y) => y.count - x.count)
      .slice(0, 5)

    const byHour = new Map<string, number>()
    for (const a of valid) {
      const h = (a.hora_inicio ?? '').slice(0, 2)
      if (!h) continue
      byHour.set(h, (byHour.get(h) ?? 0) + 1)
    }
    const peakHours = Array.from(byHour.entries())
      .map(([hh, count]) => ({ hh, count }))
      .sort((x, y) => y.count - x.count)
```

```
      .slice(0, 5)

    const firstSeenByPhone = new Map<string, string>()
    for (const a of allUntilEnd) {
      const telefone = (a.cliente_telefone ?? '').trim()
      if (!telefone) continue
      if (!firstSeenByPhone.has(telefone)) firstSeenByPhone.set(telefone, a.data)
    }
    const phonesInPeriod = new Set<string>()
    for (const a of agendamentos) {
      const telefone = (a.cliente_telefone ?? '').trim()
      if (!telefone) continue
      if (isCancelado(a.status)) continue
      phonesInPeriod.add(telefone)
    }
    let novos = 0
    let recorrentes = 0
    for (const telefone of phonesInPeriod) {
      const first = firstSeenByPhone.get(telefone) ?? null
      if (!first) continue
      if (first >= start) novos += 1
      else recorrentes += 1
    }

    return { total, concluidos, noShows, noShowRate, prevista, realizada, topServices, peakHours, novos, recorrentes }
  }, [agendamentos, allUntilEnd, start])

  const byFuncionario = useMemo(() => {
    const isCancelado = (s: string) => s.trim().toLowerCase() === 'cancelado'
    const isNoShow = (s: string) => {
      const x = s.trim().toLowerCase()
      return x === 'nao_compareceu' || x === 'não_compareceu' || x === 'no_show'
    }
    const isConcluido = (s: string) => {
      const x = s.trim().toLowerCase()
      return x === 'concluido' || x === 'concluído'
    }

    const map = new Map<
      string,
      {
        id: string
        nome: string
        total: number
        concluidos: number
        noShows: number
        prevista: number
        realizada: number
      }
    >()

    for (const a of agendamentos) {
      if (isCancelado(a.status)) continue
      const id = a.funcionario?.id ?? 'geral'
      const nome = a.funcionario?.nome_completo ?? 'Geral'
      const prev = map.get(id)
      const next = prev
        ? { ...prev }
        : {
            id,
            nome,
            total: 0,
            concluidos: 0,
            noShows: 0,
            prevista: 0,
            realizada: 0,
          }

      next.total += 1
      if (isConcluido(a.status)) next.concluidos += 1
      if (isNoShow(a.status)) next.noShows += 1
```

```
      const preco = a.servico?.preco ? Number(a.servico.preco) : 0
      next.prevista += preco
      if (isConcluido(a.status)) next.realizada += preco

      map.set(id, next)
    }

    return Array.from(map.values()).sort((x, y) => y.total - x.total)
  }, [agendamentos])

  const exportCsv = async () => {
    if (!usuarioId) return
    setExporting(true)
    setError(null)

    const { data, error: err } = await supabase
      .from('agendamentos')
      .select(

'data,hora_inicio,hora_fim,status,cliente_nome,cliente_telefone,servico:servico_id(nome,preco),funcionario:funcionario_i
d(nome_completo)'
      )
      .eq('usuario_id', usuarioId)
      .gte('data', start)
      .lte('data', end)
      .order('data', { ascending: true })
      .order('hora_inicio', { ascending: true })
      .limit(20000)

    if (err) {
      setError(err.message)
      setExporting(false)
      return
    }

    const exportData = (data ?? []) as unknown as ExportAgendamentoRow[]
    const rows = exportData.map((a) => ({
      Data: a.data,
      Inicio: a.hora_inicio,
      Fim: a.hora_fim,
      Cliente: a.cliente_nome,
      Telefone: a.cliente_telefone,
      Servico: a.servico?.nome ?? '',
      Preco: a.servico?.preco ?? '',
      Profissional: a.funcionario?.nome_completo ?? 'Geral',
      Status: a.status,
    }))

    const content = toCsv(rows)
    const blob = new Blob([content], { type: 'text/csv;charset=utf-8' })
    const url = URL.createObjectURL(blob)
    const a = document.createElement('a')
    a.href = url
    a.download = `agenda_${start}_a_${end}.csv`
    document.body.appendChild(a)
    a.click()
    a.remove()
    URL.revokeObjectURL(url)

    setExporting(false)
  }

  const sortedByHour = useMemo(() => {
    return agendamentos
      .slice()
      .filter((a) => a.status !== 'cancelado')
      .sort((x, y) => parseTimeToMinutes(x.hora_inicio) - parseTimeToMinutes(y.hora_inicio))
  }, [agendamentos])

  if (!usuario) {
    return (
      <AppShell>
```

```
        <div className="text-slate-700">{isGerente && masterUsuarioLoading ? 'Carregando…' : 'Acesso restrito.'}</div>
      </AppShell>
    )
  }

  if (funcionario && !funcionario.pode_ver_financeiro) {
    return (
      <AppShell>
        <div className="text-slate-700">Acesso restrito.</div>
      </AppShell>
    )
  }

  if (!canUseRelatorios) {
    return (
      <AppShell>
        <Card>
          <div className="p-6 space-y-3">
            <div className="text-sm font-semibold text-slate-900">Relatórios</div>
            <div className="text-sm text-slate-600">Disponível apenas nos planos PRO e EMPRESA.</div>
            <div className="flex justify-end">
              <Button onClick={() => navigate('/pagamento')}>Ver planos</Button>
            </div>
          </div>
        </Card>
      </AppShell>
    )
  }

  return (
    <PageTutorial usuarioId={usuarioId} page="relatorios">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="space-y-6">
            {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}
</div> : null}

            <Card>
              <div
                className={
                  tutorialOpen && tutorialSteps[tutorialStep]?.target === 'filters'
                    ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                    : ''
                }
              >
                <div className="p-6 space-y-4">
                  <div className="flex flex-col gap-3 sm:flex-row sm:items-end sm:justify-between">
                    <div>
                      <div className="text-sm font-semibold text-slate-900">Relatórios</div>
                      <div className="text-xs text-slate-600">Métricas baseadas nos seus agendamentos.</div>
                    </div>
                    <div className="flex gap-2">
                      <Button variant="secondary" onClick={resetTutorial}>
                        Rever tutorial
                      </Button>
                      <Button variant="secondary" onClick={exportCsv} disabled={loading || exporting}>
                        {exporting ? 'Exportando…' : 'Exportar CSV'}
                      </Button>
                    </div>
                  </div>

                  <div className="grid grid-cols-1 gap-4 sm:grid-cols-3">
                    <Input label="Início" type="date" value={start} onChange={(e) => setStart(e.target.value)} />
                    <Input label="Fim" type="date" value={end} onChange={(e) => setEnd(e.target.value)} />
                    <div className="flex items-end">
                      <Button variant="secondary" onClick={load} disabled={loading}>
                        Atualizar
                      </Button>
                    </div>
                  </div>
                </div>
              </div>
```

```
            </Card>

            {!loading && agendamentos.length === 0 ? (
              <Card>
                <div className="p-6 space-y-3">
                  <div className="text-sm font-semibold text-slate-900">Sem dados no período</div>
                  <div className="text-sm text-slate-600">Crie agendamentos na agenda para ver métricas e exportar.
</div>
                  <div>
                    <Button variant="secondary" onClick={() => navigate('/dashboard')}>
                      Ir para Agenda
                    </Button>
                  </div>
                </div>
              </Card>
            ) : null}

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'cards'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
              <div className="grid grid-cols-1 gap-4 md:grid-cols-3">
                <Card>
                  <div className="p-6 space-y-1">
                    <div className="text-xs font-semibold text-slate-500">Taxa de no-show</div>
                    <div className="text-2xl font-semibold text-slate-900">{metrics.noShowRate.toFixed(1)}%</div>
                    <div className="text-sm text-slate-600">
                      {metrics.noShows} no-show • {metrics.total} agendamentos
                    </div>
                  </div>
                </Card>
                <Card>
                  <div className="p-6 space-y-1">
                    <div className="text-xs font-semibold text-slate-500">Receita prevista</div>
                    <div className="text-2xl font-semibold text-slate-900">{formatBRMoney(metrics.prevista)}</div>
                    <div className="text-sm text-slate-600">Realizada: {formatBRMoney(metrics.realizada)}</div>
                  </div>
                </Card>
                <Card>
                  <div className="p-6 space-y-1">
                    <div className="text-xs font-semibold text-slate-500">Novos vs recorrentes</div>
                    <div className="flex items-center gap-2">
                      <Badge tone="green">Novos {metrics.novos}</Badge>
                      <Badge tone="slate">Recorrentes {metrics.recorrentes}</Badge>
                    </div>
                    <div className="text-sm text-slate-600">Até {new Date(end).toLocaleDateString('pt-BR')}</div>
                  </div>
                </Card>
              </div>
            </div>

        <div className="grid grid-cols-1 gap-4 md:grid-cols-2">
          <Card>
            <div className="p-6 space-y-3">
              <div className="text-sm font-semibold text-slate-900">Serviços mais agendados</div>
              {loading ? (
                <div className="text-sm text-slate-600">Carregando…</div>
              ) : metrics.topServices.length === 0 ? (
                <div className="text-sm text-slate-600">Sem dados no período.</div>
              ) : (
                <div className="space-y-2">
                  {metrics.topServices.map((s) => (
                    <div key={s.nome} className="flex items-center justify-between">
                      <div className="text-sm text-slate-700">{s.nome}</div>
                      <Badge tone="slate">{s.count}</Badge>
                    </div>
                  ))}
                </div>
              )}
```

```
            </div>
          </Card>
          <Card>
            <div className="p-6 space-y-3">
              <div className="text-sm font-semibold text-slate-900">Horários de pico</div>
              {loading ? (
                <div className="text-sm text-slate-600">Carregando…</div>
              ) : metrics.peakHours.length === 0 ? (
                <div className="text-sm text-slate-600">Sem dados no período.</div>
              ) : (
                <div className="space-y-2">
                  {metrics.peakHours.map((h) => (
                    <div key={h.hh} className="flex items-center justify-between">
                      <div className="text-sm text-slate-700">{h.hh}:00</div>
                      <Badge tone="slate">{h.count}</Badge>
                    </div>
                  ))}
                </div>
              )}
            </div>
          </Card>
        </div>

        <Card>
          <div className="p-6 space-y-3">
            <div className="text-sm font-semibold text-slate-900">Por profissional</div>
            {loading ? (
              <div className="text-sm text-slate-600">Carregando…</div>
            ) : byFuncionario.length === 0 ? (
              <div className="text-sm text-slate-600">Sem dados no período.</div>
            ) : (
              <div className="rounded-xl border border-slate-200 divide-y divide-slate-100">
                {byFuncionario.map((f) => (
                  <div key={f.id} className="p-3 flex flex-col gap-2 sm:flex-row sm:items-center sm:justify-between">
                    <div className="min-w-0">
                      <div className="text-sm font-semibold text-slate-900 truncate">{f.nome}</div>
                      <div className="text-xs text-slate-600">
                        {f.concluidos} concluídos • {f.noShows} no-show
                      </div>
                    </div>
                    <div className="flex flex-wrap items-center gap-2 justify-end">
                      <Badge tone="slate">{f.total} agendamentos</Badge>
                      <Badge tone="slate">Prevista {formatBRMoney(f.prevista)}</Badge>
                      <Badge tone="green">Realizada {formatBRMoney(f.realizada)}</Badge>
                    </div>
                  </div>
                ))}
              </div>
            )}
          </div>
        </Card>

        <Card>
          <div className="p-6 space-y-3">
            <div className="text-sm font-semibold text-slate-900">Agendamentos do período</div>
            {loading ? (
              <div className="text-sm text-slate-600">Carregando…</div>
            ) : sortedByHour.length === 0 ? (
              <div className="text-sm text-slate-600">Sem agendamentos.</div>
            ) : (
              <div className="rounded-xl border border-slate-200 divide-y divide-slate-100">
                {sortedByHour.slice(0, 80).map((a) => {
                  const status = a.status?.trim() ?? ''
                  const lower = status.toLowerCase()
                  const tone: 'slate' | 'green' | 'yellow' | 'red' =
                    lower === 'cancelado' ? 'red' : lower === 'pendente' ? 'yellow' : lower === 'nao_compareceu' ? 'red'
: 'green'
                  return (
                    <div key={a.id} className="p-3 flex items-start justify-between gap-3">
                      <div className="text-sm text-slate-700">
                        <div className="font-semibold text-slate-900">
                          {new Date(a.data).toLocaleDateString('pt-BR')} • {a.hora_inicio} • {a.servico?.nome ??
```

```
'Serviço'}
                        </div>
                        <div className="text-slate-600">{a.funcionario?.nome_completo ?? 'Geral'}</div>
                      </div>
                      <Badge tone={tone}>{status || 'Status'}</Badge>
                    </div>
                  )
                })}
              </div>
            )}
          </div>
        </Card>
          </div>

          <TutorialOverlay
            open={tutorialOpen}
            steps={tutorialSteps}
            step={tutorialStep}
            onStepChange={setTutorialStep}
            onClose={closeTutorial}
            titleFallback="Relatórios"
          />
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/ServicosPage.tsx

```
import { useCallback, useEffect, useMemo, useState } from 'react'
import { AppShell } from '../../components/layout/AppShell'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { formatBRMoney } from '../../lib/dates'
import { supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type Servico = {
  id: string
  usuario_id: string
  nome: string
  descricao: string | null
  duracao_minutos: number
  preco: number
  taxa_agendamento: number
  buffer_antes_min?: number
  buffer_depois_min?: number
  antecedencia_minutos?: number
  janela_max_dias?: number
  dia_inteiro?: boolean
  cor: string | null
  foto_url: string | null
  ativo: boolean
  ordem: number
}

type FormState = {
  id?: string
  nome: string
  descricao: string
  duracao_minutos: string
  preco: string
  taxa_agendamento: string
  buffer_antes_min: string
  buffer_depois_min: string
  antecedencia_minutos: string
  janela_max_dias: string
  dia_inteiro: boolean
  cor: string
  foto_url: string
  ativo: boolean
}

function toFormState(servico?: Servico | null): FormState {
  return {
    id: servico?.id,
    nome: servico?.nome ?? '',
    descricao: servico?.descricao ?? '',
    duracao_minutos: String(servico?.duracao_minutos ?? 45),
    preco: String(servico?.preco ?? 0),
    taxa_agendamento: String(servico?.taxa_agendamento ?? 0),
    buffer_antes_min: String(typeof servico?.buffer_antes_min === 'number' ? servico.buffer_antes_min : 0),
    buffer_depois_min: String(typeof servico?.buffer_depois_min === 'number' ? servico.buffer_depois_min : 0),
    antecedencia_minutos: String(typeof servico?.antecedencia_minutos === 'number' ? servico.antecedencia_minutos :
120),
    janela_max_dias: String(typeof servico?.janela_max_dias === 'number' ? servico.janela_max_dias : 15),
    dia_inteiro: Boolean(servico?.dia_inteiro),
    cor: servico?.cor ?? '#0f172a',
    foto_url: servico?.foto_url ?? '',
    ativo: servico?.ativo ?? true,
  }
}

export function ServicosPage() {
  const enableTaxaAgendamento = (import.meta.env.VITE_ENABLE_TAXA_AGENDAMENTO as string | undefined) === '1'
  const { appPrincipal, masterUsuario, masterUsuarioLoading } = useAuth()
  const funcionario = appPrincipal?.kind === 'funcionario' ? appPrincipal.profile : null
```

```
  const isGerente = appPrincipal?.kind === 'funcionario' && appPrincipal.profile.permissao === 'admin'
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : isGerente ? masterUsuario : null

  const usuarioId = usuario?.id

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Adicionar serviço',
          body: 'Clique em "+ Adicionar Serviço" para criar o que você atende (nome, duração e preço).',
          target: 'create' as const,
        },
        {
          title: 'Detalhes do serviço',
          body: 'Defina descrição e cor para facilitar a identificação na agenda. Serviços inativos não aparecem no
agendamento público.',
          target: 'form' as const,
        },
        {
          title: 'Foto (PRO)',
          body: 'No plano PRO você pode enviar foto do serviço. Isso melhora a apresentação na página pública.',
          target: 'form' as const,
        },
        {
          title: 'Organizar e ativar',
          body: 'Use as setas para ordenar a lista. Marque "Ativo" para liberar no agendamento.',
          target: 'list' as const,
        },
      ] as const,
    []
  )

  const [servicos, setServicos] = useState<Servico[]>([])
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [saving, setSaving] = useState(false)
  const [uploadingFoto, setUploadingFoto] = useState(false)
  const [regrasPorServicoDisponivel, setRegrasPorServicoDisponivel] = useState(true)
  const [formOpen, setFormOpen] = useState(false)
  const [form, setForm] = useState<FormState>(() => toFormState(null))

  const isPro = useMemo(() => {
    const p = String(usuario?.plano ?? '').trim().toLowerCase()
    return p === 'pro' || p === 'team' || p === 'enterprise'
  }, [usuario?.plano])

  const servicosLimite = isPro ? null : 3
  const limiteServicosAtingido = useMemo(() => {
    if (!servicosLimite) return false
    return servicos.length >= servicosLimite
  }, [servicos.length, servicosLimite])

  const canSubmit = useMemo(() => {
    const duracao = Number(form.duracao_minutos)
    const preco = Number(form.preco)
    const taxa = Number(form.taxa_agendamento)
    return (
      form.nome.trim() &&
      Number.isFinite(duracao) &&
      duracao > 0 &&
      Number.isFinite(preco) &&
      preco >= 0 &&
      Number.isFinite(taxa) &&
      taxa >= 0
    )
  }, [form.nome, form.duracao_minutos, form.preco, form.taxa_agendamento])

  const load = useCallback(async () => {
    if (!usuarioId) return
    setLoading(true)
    setError(null)
```

```
    const baseCols = enableTaxaAgendamento
      ? 'id,usuario_id,nome,descricao,duracao_minutos,preco,taxa_agendamento,cor,foto_url,ativo,ordem'
      : 'id,usuario_id,nome,descricao,duracao_minutos,preco,cor,foto_url,ativo,ordem'
    const colsWithRules =
`${baseCols},buffer_antes_min,buffer_depois_min,antecedencia_minutos,janela_max_dias,dia_inteiro`

    const first = await supabase
      .from('servicos')
      .select(colsWithRules)
      .eq('usuario_id', usuarioId)
      .order('ordem', { ascending: true })
      .order('criado_em', { ascending: true })

    if (first.error) {
      const msg = first.error.message
      const lower = msg.toLowerCase()
      const missingColumn = lower.includes('column') && lower.includes('does not exist')
      if (!missingColumn) {
        setError(msg)
        setLoading(false)
        return
      }

      setRegrasPorServicoDisponivel(false)
      const second = await supabase
        .from('servicos')
        .select(baseCols)
        .eq('usuario_id', usuarioId)
        .order('ordem', { ascending: true })
        .order('criado_em', { ascending: true })

      if (second.error) {
        setError(second.error.message)
        setLoading(false)
        return
      }
      setServicos((second.data ?? []) as unknown as Servico[])
      setLoading(false)
      return
    }

    setRegrasPorServicoDisponivel(true)
    setServicos((first.data ?? []) as unknown as Servico[])
    setLoading(false)
  }, [enableTaxaAgendamento, usuarioId])

  useEffect(() => {
    void (async () => {
      await Promise.resolve()
      await load()
    })().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar serviços')
      setLoading(false)
    })
  }, [load])

  if (!usuario) {
    return (
      <AppShell>
        <div className="text-slate-700">{isGerente && masterUsuarioLoading ? 'Carregando…' : 'Acesso restrito.'}</div>
      </AppShell>
    )
  }

  if (funcionario && !funcionario.pode_gerenciar_servicos) {
    return (
      <AppShell>
        <div className="text-slate-700">Acesso restrito.</div>
      </AppShell>
    )
  }
```

```
  const openCreate = () => {
    if (limiteServicosAtingido) {
      setError('Limite de serviços atingido. Para criar mais, faça upgrade no plano em Pagamento.')
      return
    }
    setForm(toFormState(null))
    setFormOpen(true)
  }

  const openEdit = (servico: Servico) => {
    setForm(toFormState(servico))
    setFormOpen(true)
  }

  const closeForm = () => {
    setFormOpen(false)
    setForm(toFormState(null))
  }

  const uploadFotoServico = async (file: File) => {
    if (!usuarioId) throw new Error('Sessão inválida')

    const safeName = file.name
      .trim()
      .replace(/\s+/g, '-')
      .replace(/[^a-zA-Z0-9._-]/g, '')
      .slice(0, 80)
    const key = `${usuarioId}/servicos/${Date.now()}-${safeName || 'foto'}`

    const tryUpload = async (bucket: string) => {
      const { error: uploadErr } = await supabase.storage.from(bucket).upload(key, file, {
        upsert: true,
        contentType: file.type || undefined,
      })
      if (uploadErr) throw uploadErr
      const { data } = supabase.storage.from(bucket).getPublicUrl(key)
      const publicUrl = data?.publicUrl
      if (!publicUrl) throw new Error('Não foi possível obter a URL pública da foto')
      return publicUrl
    }

    try {
      return await tryUpload('servicos')
    } catch (err: unknown) {
      const msg = err instanceof Error ? err.message : String(err)
      const lower = msg.toLowerCase()
      const missingBucket = lower.includes('bucket') && lower.includes('not found')
      if (!missingBucket) throw err
      return await tryUpload('logos')
    }
  }

  const submit = async () => {
    if (!canSubmit) return
    setSaving(true)
    setError(null)

    const canUseFotos = isPro

    const payload: Record<string, unknown> = {
      nome: form.nome.trim(),
      descricao: form.descricao.trim() ? form.descricao.trim() : null,
      duracao_minutos: Number(form.duracao_minutos),
      preco: Number(form.preco),
      cor: form.cor || null,
      ativo: Boolean(form.ativo),
    }

    if (enableTaxaAgendamento) {
      payload.taxa_agendamento = Number(form.taxa_agendamento)
    }
```

```
    if (regrasPorServicoDisponivel) {
      const bufferAntes = Number(form.buffer_antes_min)
      const bufferDepois = Number(form.buffer_depois_min)
      const antecedencia = Number(form.antecedencia_minutos)
      const janela = Number(form.janela_max_dias)

      payload.buffer_antes_min = Number.isFinite(bufferAntes) ? Math.max(0, Math.floor(bufferAntes)) : 0
      payload.buffer_depois_min = Number.isFinite(bufferDepois) ? Math.max(0, Math.floor(bufferDepois)) : 0
      payload.antecedencia_minutos = Number.isFinite(antecedencia) ? Math.max(0, Math.floor(antecedencia)) : 120
      payload.janela_max_dias = Number.isFinite(janela) ? Math.max(1, Math.floor(janela)) : 15
      payload.dia_inteiro = Boolean(form.dia_inteiro)
    }

    if (canUseFotos) {
      const nextFoto = form.foto_url.trim()
      payload.foto_url = nextFoto ? nextFoto : null
    }

    if (form.id) {
      const { error: err } = await supabase.from('servicos').update(payload).eq('id', form.id).eq('usuario_id',
usuario.id)
      if (err) {
        setError(err.message)
        setSaving(false)
        return
      }
    } else {
      if (limiteServicosAtingido) {
        setError('Limite de serviços atingido. Para criar mais, faça upgrade no plano em Pagamento.')
        setSaving(false)
        return
      }
      const nextOrder = (servicos.at(-1)?.ordem ?? -1) + 1
      const { error: err } = await supabase.from('servicos').insert({ usuario_id: usuario.id, ordem: nextOrder,
...payload })
      if (err) {
        const msg = err.message.toLowerCase().includes('limite_servicos_atingido')
          ? 'Limite de serviços atingido. Para criar mais, faça upgrade no plano em Pagamento.'
          : err.message
        setError(msg)
        setSaving(false)
        return
      }
    }

    setSaving(false)
    closeForm()
    await load()
  }

  const remove = async (id: string) => {
    setError(null)
    const { error: err } = await supabase.from('servicos').delete().eq('id', id).eq('usuario_id', usuario.id)
    if (err) {
      setError(err.message)
      return
    }
    await load()
  }

  const toggleAtivo = async (servico: Servico) => {
    setError(null)
    const { error: err } = await supabase
      .from('servicos')
      .update({ ativo: !servico.ativo })
      .eq('id', servico.id)
      .eq('usuario_id', usuario.id)
    if (err) {
      setError(err.message)
      return
    }
```

```
    setServicos((prev) => prev.map((s) => (s.id === servico.id ? { ...s, ativo: !s.ativo } : s)))
  }

  const move = async (id: string, dir: -1 | 1) => {
    const index = servicos.findIndex((s) => s.id === id)
    const otherIndex = index + dir
    if (index < 0 || otherIndex < 0 || otherIndex >= servicos.length) return
    const a = servicos[index]
    const b = servicos[otherIndex]

    setServicos((prev) => {
      const copy = [...prev]
      copy[index] = b
      copy[otherIndex] = a
      return copy
    })

    const updates = [
      supabase.from('servicos').update({ ordem: b.ordem }).eq('id', a.id).eq('usuario_id', usuario.id),
      supabase.from('servicos').update({ ordem: a.ordem }).eq('id', b.id).eq('usuario_id', usuario.id),
    ]
    const res = await Promise.all(updates)
    const firstErr = res.find((r) => r.error)?.error
    if (firstErr) {
      setError(firstErr.message)
      await load()
    }
  }

  return (
    <PageTutorial usuarioId={usuarioId} page="servicos">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="space-y-6">
            <div
              className={[
                'flex items-center justify-between',
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'create'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl p-2 -m-2'
                  : '',
              ].join(' ')}
            >
          <div>
            <div className="text-sm font-semibold text-slate-500">Meus Serviços</div>
            <div className="text-xl font-semibold text-slate-900">Gerencie serviços e preços</div>
          </div>
          <div className="flex items-center gap-2">
            <Button variant="secondary" onClick={resetTutorial}>
              Rever tutorial
            </Button>
            <Button onClick={openCreate} disabled={limiteServicosAtingido}>
              + Adicionar Serviço
            </Button>
          </div>
        </div>

        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}
        {!isPro && servicosLimite ? (
          <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-800">
            Plano BASIC: até {servicosLimite} serviços.
          </div>
        ) : null}

        {formOpen ? (
          <div
            className={
              tutorialOpen && tutorialSteps[tutorialStep]?.target === 'form'
                ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                : ''
            }
          >
```

```
          <Card>
            <div className="p-6 space-y-4">
            <div className="text-sm font-semibold text-slate-900">{form.id ? 'Editar serviço' : 'Adicionar serviço'}
</div>
            <Input label="Nome" value={form.nome} onChange={(e) => setForm((p) => ({ ...p, nome: e.target.value }))}
/>
            <Input
              label="Descrição (opcional)"
              value={form.descricao}
              onChange={(e) => setForm((p) => ({ ...p, descricao: e.target.value }))}
            />
            <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
              <Input
                label="Duração (min)"
                type="number"
                value={form.duracao_minutos}
                onChange={(e) => setForm((p) => ({ ...p, duracao_minutos: e.target.value }))}
              />
              <Input
                label="Preço"
                type="number"
                value={form.preco}
                onChange={(e) => setForm((p) => ({ ...p, preco: e.target.value }))}
              />
            </div>

            {regrasPorServicoDisponivel ? (
              <label className="flex items-center gap-2 text-sm text-slate-700">
                <input type="checkbox" checked={form.dia_inteiro} onChange={(e) => setForm((p) => ({ ...p,
dia_inteiro: e.target.checked }))} />
                Dura o dia inteiro (ex.: faxina)
              </label>
            ) : null}

            {regrasPorServicoDisponivel ? (
              <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                <Input
                  label="Buffer antes (min)"
                  type="number"
                  value={form.buffer_antes_min}
                  onChange={(e) => setForm((p) => ({ ...p, buffer_antes_min: e.target.value }))}
                />
                <Input
                  label="Buffer depois (min)"
                  type="number"
                  value={form.buffer_depois_min}
                  onChange={(e) => setForm((p) => ({ ...p, buffer_depois_min: e.target.value }))}
                />
                <Input
                  label="Antecedência mínima (min)"
                  type="number"
                  value={form.antecedencia_minutos}
                  onChange={(e) => setForm((p) => ({ ...p, antecedencia_minutos: e.target.value }))}
                />
                <Input
                  label="Janela de agendamento (dias)"
                  type="number"
                  value={form.janela_max_dias}
                  onChange={(e) => setForm((p) => ({ ...p, janela_max_dias: e.target.value }))}
                />
              </div>
            ) : (
              <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-800">
                Regras por serviço (buffer/antecedência/janela) ainda não estão habilitadas no seu Supabase. Aplique o
SQL do link público atualizado em /admin/configuracoes.
              </div>
            )}
            {enableTaxaAgendamento ? (
              <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                <Input
                  label="Taxa de agendamento"
                  type="number"
```

```
                    value={form.taxa_agendamento}
                    onChange={(e) => setForm((p) => ({ ...p, taxa_agendamento: e.target.value }))}
                  />
                </div>
              ) : null}
              <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                <Input label="Cor" type="color" value={form.cor} onChange={(e) => setForm((p) => ({ ...p, cor:
e.target.value }))} />
                <label className="flex items-end gap-2">
                  <input
                    type="checkbox"
                    checked={form.ativo}
                    onChange={(e) => setForm((p) => ({ ...p, ativo: e.target.checked }))}
                    className="h-4 w-4"
                  />
                  <span className="text-sm text-slate-700">Ativo</span>
                </label>
              </div>

              {usuario.plano === 'pro' || usuario.plano === 'team' || usuario.plano === 'enterprise' ? (
                <div className="space-y-2">
                  <Input
                    label="Foto do serviço (opcional)"
                    type="file"
                    accept="image/*"
                    disabled={uploadingFoto || saving}
                    onChange={(e) => {
                      const file = e.target.files?.[0]
                      if (!file || !usuarioId) return

                      const maxBytes = 6 * 1024 * 1024
                      if (file.size > maxBytes) {
                        setError('A foto deve ter no máximo 6MB.')
                        e.target.value = ''
                        return
                      }

                      void (async () => {
                        setUploadingFoto(true)
                        setError(null)
                        const publicUrl = await uploadFotoServico(file)
                        setForm((p) => ({ ...p, foto_url: publicUrl }))
                        e.target.value = ''
                        setUploadingFoto(false)
                      })().catch((err: unknown) => {
                        const msg = err instanceof Error ? err.message : 'Erro ao enviar foto'
                        const lower = msg.toLowerCase()
                        const missingBucket = lower.includes('bucket') && lower.includes('not found')
                        if (missingBucket) {
                          const host = supabaseEnv.ok
                            ? (() => {
                                try {
                                  return new URL(supabaseEnv.values.VITE_SUPABASE_URL).host
                                } catch {
                                  return supabaseEnv.values.VITE_SUPABASE_URL
                                }
                              })()
                            : 'supabase_desconfigurado'
                          setError(
                            `Bucket não encontrado no Storage. Projeto: ${host}. Crie o bucket "servicos" (público) ou
use o SQL "Storage (fotos de serviços)" em /admin/configuracoes.`
                          )
                        } else {
                          setError(msg)
                        }
                        setUploadingFoto(false)
                        e.target.value = ''
                      })
                    }}
                  />

                  {form.foto_url.trim() ? (
```

```
          <div className="flex items-center justify-between gap-3 rounded-xl border border-slate-200 bg-white
p-3">
           <div className="flex items-center gap-3 min-w-0">
            <img src={form.foto_url} alt="Foto do serviço" className="h-12 w-12 rounded-lg object-cover
border border-slate-200" />
            <div className="text-sm text-slate-700 truncate">{form.foto_url}</div>
           </div>
           <Button
            type="button"
            variant="secondary"
            disabled={saving || uploadingFoto}
            onClick={() => setForm((p) => ({ ...p, foto_url: '' }))}
           >
            Remover
           </Button>
          </div>
         ) : null}
        </div>
       ) : (
        <div className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-sm text-slate-700">
         Foto do serviço disponível a partir do plano PRO.
        </div>
       )}

       <div className="flex flex-wrap justify-end gap-2">
        <Button variant="secondary" onClick={closeForm} disabled={saving || uploadingFoto}>
         Cancelar
        </Button>
        <Button onClick={submit} disabled={!canSubmit || saving || uploadingFoto}>
         Salvar
        </Button>
       </div>
       </div>
      </Card>
     </div>
    ) : null}

    <div
     className={
      tutorialOpen && tutorialSteps[tutorialStep]?.target === 'list'
       ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
       : ''
     }
    >
     <Card>
      <div className="divide-y divide-slate-100">
      {loading ? (
       <div className="p-6 text-sm text-slate-600">Carregando serviços…</div>
      ) : servicos.length === 0 ? (
       <div className="p-6 space-y-3">
        <div className="text-sm font-semibold text-slate-900">Nenhum serviço cadastrado</div>
        <div className="text-sm text-slate-600">Crie seus serviços para que eles apareçam na agenda e na página
pública.</div>
        <div>
         <Button variant="secondary" onClick={openCreate} disabled={limiteServicosAtingido}>
          + Adicionar primeiro serviço
         </Button>
        </div>
       </div>
      ) : (
       servicos.map((s, idx) => (
        <div key={s.id} className="p-4 flex flex-col gap-3 sm:flex-row sm:items-center sm:justify-between">
         <div className="flex items-start gap-3">
          {s.foto_url ? (
           <img src={s.foto_url} alt={s.nome} className="h-12 w-12 rounded-lg object-cover border border-
slate-200" />
          ) : (
           <div className="mt-1 h-4 w-4 rounded" style={{ backgroundColor: s.cor ?? '#0f172a' }} />
          )}
          <div>
           <div className="text-sm font-semibold text-slate-900">{s.nome}</div>
           <div className="text-sm text-slate-600">
```

```
                        {s.duracao_minutos} min • {formatBRMoney(Number(s.preco ?? 0))}
                      </div>
                      {enableTaxaAgendamento && typeof s.taxa_agendamento === 'number' && Number(s.taxa_agendamento) > 0
? (
                        <div className="text-xs text-slate-600">Taxa: {formatBRMoney(Number(s.taxa_agendamento))}</div>
                      ) : null}
                      {s.descricao ? <div className="text-sm text-slate-600">{s.descricao}</div> : null}
                    </div>
                  </div>

                  <div className="flex flex-wrap items-center gap-2 justify-end">
                    <Button variant="ghost" onClick={() => move(s.id, -1)} disabled={idx === 0}>
                      ↑
                    </Button>
                    <Button variant="ghost" onClick={() => move(s.id, 1)} disabled={idx === servicos.length - 1}>
                      ↓
                    </Button>
                    <label className="flex items-center gap-2 rounded-lg border border-slate-200 px-3 py-2 text-sm text-
slate-700">
                      <input type="checkbox" checked={s.ativo} onChange={() => toggleAtivo(s)} />
                      Ativo
                    </label>
                    <Button variant="secondary" onClick={() => openEdit(s)}>
                      Editar
                    </Button>
                    <Button
                      variant="danger"
                      onClick={() => {
                        const ok = window.confirm(`Excluir o serviço "${s.nome}"?`)
                        if (ok) void remove(s.id)
                      }}
                    >
                      Excluir
                    </Button>
                  </div>
                </div>
              ))
            )}
            </div>
          </Card>
        </div>

            <TutorialOverlay open={tutorialOpen} steps={tutorialSteps} step={tutorialStep} onStepChange=
{setTutorialStep} onClose={closeTutorial} />
          </div>
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/WhatsappSettingsPage.tsx

```
import { useEffect, useMemo, useRef, useState } from 'react'
import { AppShell } from '../../components/layout/AppShell'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { checkJwtProject, supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

function getOptionalString(payload: unknown, key: string) {
  if (!payload || typeof payload !== 'object') return null
  const value = (payload as Record<string, unknown>)[key]
  return typeof value === 'string' ? value : null
}

function formatDetails(payload: unknown) {
  if (typeof payload === 'string') {
    const s = payload.trim()
    const lower = s.toLowerCase()
    const isHtml = lower.includes('<!doctype html') || lower.includes('<html') || lower.includes('<head')
    const isCloudflare =
      lower.includes('cloudflare tunnel error') ||
      (lower.includes('cloudflare') && (lower.includes('cf-ray') || lower.includes('tunnel') ||
lower.includes('cloudflareapps')))
    if (isCloudflare) return 'Cloudflare Tunnel indisponível. Verifique Cloudflare Tunnel ativo e URL da Evolution API.'
    if (isHtml) return 'Resposta HTML inesperada ao chamar o WhatsApp. Verifique proxy/túnel e a URL configurada.'
    if (s.length > 900) return `${s.slice(0, 900)}…`
    return s
  }
  if (payload && typeof payload === 'object') {
    const msg = getOptionalString(payload, 'message') ?? getOptionalString(payload, 'details') ??
getOptionalString(payload, 'error')
    if (msg) return formatDetails(msg)
  }
  try {
    const s = JSON.stringify(payload)
    if (s.length > 900) return `${s.slice(0, 900)}…`
    return s
  } catch {
    return String(payload)
  }
}

function isWorkerLimitResponse(args: { status: number; body: unknown }) {
  if (args.status === 546) return true
  if (!args.body || typeof args.body !== 'object') return false
  return (args.body as Record<string, unknown>).code === 'WORKER_LIMIT'
}

async function callWhatsappFunction(body: unknown, opts?: { timeoutMs?: number }) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } }
  }

  const supabaseUrl = supabaseEnv.values.VITE_SUPABASE_URL
  const supabaseAnonKey = supabaseEnv.values.VITE_SUPABASE_ANON_KEY

  const tryRefresh = async () => {
    const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
    if (refreshErr) return null
    return refreshed.session ?? null
  }

  const { data: sessionData } = await supabase.auth.getSession()
  let session = sessionData.session
  const now = Math.floor(Date.now() / 1000)
  const expiresAt = session?.expires_at ?? null

  if (session && expiresAt && expiresAt <= now + 60) {
    const refreshed = await tryRefresh()
```

```
    if (refreshed) session = refreshed
  }

  const accessToken=[REDACTED]
  if (!accessToken) {
    return { ok: false as const, status: 401, body: { error: 'session_expired' } }
  }

  const tokenProject = checkJwtProject(accessToken, supabaseUrl)
  if (!tokenProject.ok) {
    await supabase.auth.signOut().catch(() => undefined)
    return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: tokenProject.iss, expected: tokenProject.expectedPrefix } }
  }

  const callFetch = async (token: string) => {
    const fnUrl = `${supabaseUrl}/functions/v1/whatsapp`
    const controller = new AbortController()
    const timeoutId = setTimeout(() => controller.abort(), opts?.timeoutMs ?? 45000)
    try {
      const headers: Record<string, string> = {
        'Content-Type': 'application/json',
        apikey=[REDACTED]
        Authorization: `Bearer ${token}`,
        'x-user-jwt': token,
      }
      const res = await fetch(fnUrl, {
        method: 'POST',
        headers,
        body: JSON.stringify(body ?? {}),
        signal: controller.signal,
      })

      const fnVersion = res.headers.get('x-smagenda-fn')

      let text = ''
      try {
        text = await res.text()
      } catch (e: unknown) {
        const msg = e instanceof Error ? e.message : 'Falha de rede'
        return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
      }

      let parsed: unknown = null
      try {
        parsed = text ? JSON.parse(text) : null
      } catch {
        parsed = text
      }

      if (!res.ok && res.status === 401 && !fnVersion) {
        if (
          parsed &&
          typeof parsed === 'object' &&
          (parsed as Record<string, unknown>).message === 'Invalid JWT' &&
          (parsed as Record<string, unknown>).code === 401
        ) {
          return { ok: false as const, status: 401, body: { error: 'supabase_gateway_invalid_jwt' } }
        }
      }

      if (!res.ok) return { ok: false as const, status: res.status, body: parsed }
      return { ok: true as const, status: res.status, body: parsed }
    } catch (e: unknown) {
      const errAny = e as { name?: unknown; message?: unknown }
      const name = typeof errAny.name === 'string' ? errAny.name : ''
      const msg = typeof errAny.message === 'string' ? errAny.message : 'Falha de rede'
      const lower = msg.toLowerCase()
      const isAbort = name === 'AbortError' || lower.includes('aborted') || lower.includes('abort')
      if (isAbort) return { ok: false as const, status: 0, body: { error: 'timeout' } }
      return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
    } finally {
```

```
      clearTimeout(timeoutId)
    }
  }

  const isInvalidJwtPayload = (payload: unknown) => {
    if (typeof payload === 'string') return payload.includes('Invalid JWT')
    if (!payload || typeof payload !== 'object') return false
    const obj = payload as Record<string, unknown>
    return obj.message === 'Invalid JWT' || obj.error === 'invalid_jwt'
  }

  const first = await callFetch(accessToken)

  const action = (() => {
    if (!body || typeof body !== 'object') return null
    const raw = (body as Record<string, unknown>).action
    return typeof raw === 'string' ? raw : null
  })()

  if (!first.ok && first.status === 0 && action === 'status') {
    await new Promise((r) => setTimeout(r, 300))
    return callFetch(accessToken)
  }

  if (
    !first.ok &&
    first.status === 401 &&
    typeof first.body === 'object' &&
    first.body !== null &&
    (first.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
  ) {
    return first
  }

  if (!first.ok && first.status === 401 && isInvalidJwtPayload(first.body)) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }

    const second = await callFetch(nextToken)
    if (!second.ok && second.status === 401 && isInvalidJwtPayload(second.body)) {
      return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    }
    return second
  }

  return first
}

export function WhatsappSettingsPage() {
  const { appPrincipal } = useAuth()
  const usuarioId = appPrincipal?.kind === 'usuario' ? appPrincipal.profile.id : null

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Status e pré-requisitos',
          body: 'Verifique se o recurso está habilitado e se a configuração do WhatsApp está OK. Se estiver pendente, o 
Super Admin precisa concluir a configuração.',
          target: 'status' as const,
        },
        {
          title: 'Conectar com QR Code',
          body: 'Clique em "Conectar (QR Code)" e escaneie o QR Code no WhatsApp para vincular a instância.',
          target: 'qr' as const,
        },
        {
          title: 'Testar envio',
          body: 'Envie uma mensagem de teste para validar que está tudo funcionando em produção.',
          target: 'test' as const,
        },
```

```
      ] as const,
    []
  )

  const [configurado, setConfigurado] = useState(false)
  const [whatsappHabilitado, setWhatsappHabilitado] = useState<boolean | null>(null)
  const [loading, setLoading] = useState(true)
  const [connecting, setConnecting] = useState(false)
  const [disconnecting, setDisconnecting] = useState(false)
  const [checkingStatus, setCheckingStatus] = useState(false)

  const [sendingTest, setSendingTest] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [saved, setSaved] = useState(false)

  const [instanceState, setInstanceState] = useState<string | null>(null)
  const [qrBase64, setQrBase64] = useState<string | null>(null)
  const [testNumber, setTestNumber] = useState('')
  const [testText, setTestText] = useState('Olá! Teste de envio do SMagenda.')

  const aliveRef = useRef(true)
  const allowStatusRef = useRef(false)

  useEffect(() => {
    aliveRef.current = true
    return () => {
      aliveRef.current = false
    }
  }, [])

  const habilitado = useMemo(() => (whatsappHabilitado === null ? true : Boolean(whatsappHabilitado)),
[whatsappHabilitado])

  const callWhatsapp = async (body: unknown) => {
    const action = (() => {
      if (!body || typeof body !== 'object') return null
      const raw = (body as Record<string, unknown>).action
      return typeof raw === 'string' ? raw : null
    })()

    if (action === 'status' && !allowStatusRef.current) {
      return { ok: false as const, status: 0, body: { error: 'status_not_allowed' } }
    }

    return callWhatsappFunction(body)
  }

  useEffect(() => {
    const run = async () => {
      if (!usuarioId) return
      setLoading(true)
      setError(null)
      setInstanceState(null)
      const isMissingColumnError = (message: string) => message.toLowerCase().includes('does not exist') &&
message.toLowerCase().includes('column')

      let enabled: boolean | null = null

      const first = await supabase.from('usuarios').select('whatsapp_habilitado').eq('id', usuarioId).maybeSingle()
      if (first.error) {
        if (isMissingColumnError(first.error.message)) {
          enabled = null
          setWhatsappHabilitado(null)
        } else {
          setError(first.error.message)
          setConfigurado(false)
          setLoading(false)
          return
        }
      } else {
        const row = (first.data ?? null) as unknown as { whatsapp_habilitado?: boolean | null } | null
        enabled = typeof row?.whatsapp_habilitado === 'boolean' ? row.whatsapp_habilitado : null
```

```
        setWhatsappHabilitado(enabled)
      }
      const cfg = await callWhatsappFunction({ action: 'config_status' })
      if (!cfg.ok) {
        const hint = cfg.body && typeof cfg.body === 'object' ? getOptionalString(cfg.body, 'hint') : null
        const details = typeof cfg.body === 'string' ? cfg.body : JSON.stringify(cfg.body)
        setError(hint ? `Falha ao carregar configuração do WhatsApp: ${details} | Dica: ${hint}` : `Falha ao carregar
configuração do WhatsApp: ${details}`)
        setConfigurado(false)
        setLoading(false)
        return
      }

      const configured = cfg.body && typeof cfg.body === 'object' ? Boolean((cfg.body as Record<string,
unknown>).configured) : false
      setConfigurado(configured)

      const enabledSafe = enabled === null ? true : Boolean(enabled)
      if (enabledSafe && configured) {
        const quick = await callWhatsappFunction({ action: 'status' }, { timeoutMs: 6000 })
        if (!aliveRef.current) return
        if (quick.ok && quick.body && typeof quick.body === 'object') {
          const s = getOptionalString(quick.body, 'state')
          setInstanceState(s)
          if (s === 'open') {
            setQrBase64(null)
          }
        }
      }
      setLoading(false)
    }
    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar configurações')
      setConfigurado(false)
      setLoading(false)
      setCheckingStatus(false)
    })
  }, [usuarioId])

  if (!usuarioId) {
    return (
      <AppShell>
        <div className="text-slate-700">{appPrincipal ? 'Acesso restrito.' : 'Carregando…'}</div>
      </AppShell>
    )
  }

  const connect = async () => {
    setError(null)
    setQrBase64(null)
    if (!habilitado) {
      setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
      return
    }
    if (!configurado) {
      setError('WhatsApp ainda não foi configurado no painel do Super Admin.')
      return
    }
    setConnecting(true)
    allowStatusRef.current = true
    const res = await callWhatsapp({ action: 'connect' })
    if (!res.ok) {
      if (isWorkerLimitResponse({ status: res.status, body: res.body })) {
        setError('O Supabase está sem recursos para executar a Edge Function agora (WORKER_LIMIT). Aguarde 1–2 minutos e
tente novamente.')
        setConnecting(false)
        return
      }
      if (res.status === 0 && typeof res.body === 'object' && res.body !== null && (res.body as Record<string,
unknown>).error === 'timeout') {
        setError('Tempo esgotado ao gerar QR Code. A Edge Function pode estar temporariamente sem recursos ou
indisponível. Aguarde 1–2 minutos e tente novamente.')
```

```
        setConnecting(false)
        return
      }
      if (
        typeof res.body === 'object' &&
        res.body !== null &&
        (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
      ) {
        setError('JWT inválido para chamar a Edge Function. Saia e entre novamente. Se persistir, reimplante a função
com verify_jwt=false (--no-verify-jwt).')
        setConnecting(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'jwt_project_mismatch') {
        setError('Sessão do Supabase pertence a outro projeto. Saia e entre novamente no sistema.')
        setConnecting(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'invalid_jwt') {
        setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
        setConnecting(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'whatsapp_disabled') {
        setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
        setConnecting(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'whatsapp_not_configured') {
        setError('WhatsApp ainda não foi configurado no painel do Super Admin.')
        setConnecting(false)
        return
      }
      const hint = getOptionalString(res.body, 'hint')
      const details = formatDetails(res.body)
      setError(hint ? `Falha ao gerar QR Code (HTTP ${res.status}): ${details}\n\nDica: ${hint}` : `Falha ao gerar QR
Code (HTTP ${res.status}): ${details}`)
      setConnecting(false)
      return
    }

    const initialState = getOptionalString(res.body, 'state')
    const initialQr = getOptionalString(res.body, 'qrBase64')
    let currentState: string | null = initialState
    if (initialState) setInstanceState(initialState)
    if (initialQr && initialQr.trim()) {
      setQrBase64(initialQr)
    }

    for (let i = 0; i < 30; i += 1) {
      if (currentState === 'open') {
        setQrBase64(null)
        setConnecting(false)
        return
      }
      await new Promise((r) => setTimeout(r, 4000))
      if (!aliveRef.current) return
      const next = await callWhatsapp({ action: 'status' })
      if (!next.ok) {
        if (isWorkerLimitResponse({ status: next.status, body: next.body })) {
          setError('O Supabase está sem recursos para executar a Edge Function agora (WORKER_LIMIT). Aguarde 1–2 minutos
e tente novamente.')
          setConnecting(false)
          return
        }
        if (next.status === 0 && typeof next.body === 'object' && next.body !== null && (next.body as Record<string,
unknown>).error === 'timeout') {
          setError('Tempo esgotado ao verificar status do WhatsApp. A Edge Function pode estar temporariamente sem
```

```
recursos ou indisponível. Aguarde 1–2 minutos e tente novamente.')
          setConnecting(false)
          return
        }
        const hint = getOptionalString(next.body, 'hint')
        const details = formatDetails(next.body)
        setError(hint ? `Falha ao verificar status (HTTP ${next.status}): ${details}\n\nDica: ${hint}` : `Falha ao
verificar status (HTTP ${next.status}): ${details}`)
        setConnecting(false)
        return
      }
      const nextState = getOptionalString(next.body, 'state')
      if (nextState) {
        currentState = nextState
        setInstanceState(nextState)
      }
      if (currentState === 'open') {
        setQrBase64(null)
        setConnecting(false)
        return
      }
    }

    const hint = getOptionalString(res.body, 'hint')
    if (hint) {
      setError(hint)
    } else {
      setError('A instância ainda não conectou. Se o QR Code expirou, gere um novo e tente novamente.')
    }
    setConnecting(false)
  }

  const disconnect = async () => {
    setError(null)
    if (!habilitado) return
    if (!configurado) return
    setDisconnecting(true)
    const res = await callWhatsappFunction({ action: 'disconnect' })
    if (!res.ok) {
      if (
        typeof res.body === 'object' &&
        res.body !== null &&
        (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
      ) {
        setError('A Edge Function "whatsapp" está exigindo JWT no Supabase. Refaça o deploy com verify_jwt=false e tente
novamente.')
        setDisconnecting(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'invalid_jwt') {
        setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
        setDisconnecting(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'whatsapp_disabled') {
        setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
        setDisconnecting(false)
        return
      }
      const details = typeof res.body === 'string' ? res.body : JSON.stringify(res.body)
      setError(`Falha ao desconectar (HTTP ${res.status}): ${details}`)
      setDisconnecting(false)
      return
    }
    setInstanceState(null)
    setQrBase64(null)
    setDisconnecting(false)
  }

  const checkStatus = async () => {
```

```
    setError(null)
    if (!habilitado) {
      setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
      return
    }
    if (!configurado) {
      setError('WhatsApp ainda não foi configurado no painel do Super Admin.')
      return
    }
    setCheckingStatus(true)
    allowStatusRef.current = true
    const res = await callWhatsapp({ action: 'status' })
    if (!res.ok) {
      if (isWorkerLimitResponse({ status: res.status, body: res.body })) {
        setError('O Supabase está sem recursos para executar a Edge Function agora (WORKER_LIMIT). Aguarde 1–2 minutos e 
tente novamente.')
        setCheckingStatus(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error === 
'timeout') {
        setError('Tempo esgotado ao verificar status do WhatsApp. A Edge Function pode estar temporariamente sem 
recursos ou indisponível. Aguarde 1–2 minutos e tente novamente.')
        setCheckingStatus(false)
        return
      }
      if (
        typeof res.body === 'object' &&
        res.body !== null &&
        (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
      ) {
        setError('JWT inválido para chamar a Edge Function. Saia e entre novamente. Se persistir, reimplante a função 
com verify_jwt=false (--no-verify-jwt).')
        setCheckingStatus(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error === 
'jwt_project_mismatch') {
        setError('Sessão do Supabase pertence a outro projeto. Saia e entre novamente no sistema.')
        setCheckingStatus(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error === 
'invalid_jwt') {
        setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
        setCheckingStatus(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error === 
'whatsapp_disabled') {
        setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
        setCheckingStatus(false)
        return
      }
      const hint = getOptionalString(res.body, 'hint')
      const details = formatDetails(res.body)
      setError(hint ? `Falha ao atualizar status (HTTP ${res.status}): ${details}\n\nDica: ${hint}` : `Falha ao 
atualizar status (HTTP ${res.status}): ${details}`)
      setCheckingStatus(false)
      return
    }
    const nextState = getOptionalString(res.body, 'state')
    setInstanceState(nextState)
    if (nextState === 'open') {
      setQrBase64(null)
    }
    setCheckingStatus(false)
  }

  const sendTest = async () => {
    setError(null)
    if (!habilitado) {
```

```
      setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
      return
    }
    if (!configurado) {
      setError('WhatsApp ainda não foi configurado no painel do Super Admin.')
      return
    }
    if (!testNumber.trim() || !testText.trim()) {
      setError('Informe número e mensagem.')
      return
    }
    setSendingTest(true)
    const res = await callWhatsappFunction({ action: 'send_test', number: testNumber, text: testText })
    if (!res.ok) {
      if (
        typeof res.body === 'object' &&
        res.body !== null &&
        (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
      ) {
        setError('A Edge Function "whatsapp" está exigindo JWT no Supabase. Refaça o deploy com verify_jwt=false e tente
novamente.')
        setSendingTest(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'invalid_jwt') {
        setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
        setSendingTest(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'whatsapp_disabled') {
        setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
        setSendingTest(false)
        return
      }
      const hint = getOptionalString(res.body, 'hint')
      const details = formatDetails(res.body)
      const attemptsText = (() => {
        const raw = (res.body as Record<string, unknown> | null)?.attempts
        if (!raw) return null
        try {
          return JSON.stringify(raw)
        } catch {
          return null
        }
      })()
      setError(
        hint
          ? `Falha ao enviar teste (HTTP ${res.status}): ${details}\n\nDica: ${hint}${attemptsText ? `\n\nTentativas:
${attemptsText}` : ''}`
          : `Falha ao enviar teste (HTTP ${res.status}): ${details}${attemptsText ? `\n\nTentativas: ${attemptsText}` :
''}`
      )
      setSendingTest(false)
      return
    }
    setSendingTest(false)
    setSaved(true)
    setTimeout(() => setSaved(false), 2000)
  }

  const isConnected = instanceState?.toLowerCase() === 'open'
  const connectionLabel = (() => {
    if (!habilitado || !configurado) return '—'
    if (checkingStatus) return 'verificando…'
    if (isConnected) return 'Conectado'
    if (!instanceState) return 'Desconectado'
    const s = instanceState.toLowerCase()
    if (s === 'close' || s === 'closed') return 'Desconectado'
    if (s === 'connecting') return 'Conectando…'
    return instanceState
```

```
  })()

  return (
    <PageTutorial usuarioId={usuarioId} page="whatsapp">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="space-y-6">
            <div className="flex items-center justify-between">
              <div>
                <div className="text-sm font-semibold text-slate-500">Configurações</div>
                <div className="text-xl font-semibold text-slate-900">WhatsApp</div>
              </div>
              <Button variant="secondary" onClick={resetTutorial}>
                Rever tutorial
              </Button>
            </div>

        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> :
null}
        {saved ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-
800">Enviado.</div> : null}

        <div
          className={
            tutorialOpen && tutorialSteps[tutorialStep]?.target === 'status'
              ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
              : ''
          }
        >
          <Card>
            <div className="p-6 space-y-4">
              <div className="text-sm font-semibold text-slate-900">Status</div>
              {loading ? (
                <div className="text-sm text-slate-600">Carregando…</div>
              ) : (
                <div className="space-y-2">
                  <div className="text-sm text-slate-700">Recurso: {habilitado ? '🟢 Habilitado' : '🔴 Desabilitado'}
</div>
                  <div className="text-sm text-slate-700">Configuração: {configurado ? '🟢 OK' : '🟡 Pendente'}</div>
                  <div className="text-sm text-slate-700">Conexão: {connectionLabel}</div>
                  <div className="flex flex-wrap gap-2">
                    <Button variant="secondary" onClick={checkStatus} disabled={!habilitado || !configurado ||
checkingStatus || connecting || disconnecting}>
                      {isConnected ? 'Atualizar status' : 'Verificar status'}
                    </Button>
                    {isConnected ? (
                      <Button onClick={checkStatus} disabled={!habilitado || !configurado || checkingStatus ||
connecting || disconnecting}>
                        Testar conexão
                      </Button>
                    ) : (
                      <Button onClick={() => void connect()} disabled={!habilitado || !configurado || checkingStatus ||
connecting || disconnecting}>
                        Conectar (QR Code)
                      </Button>
                    )}
                    <Button variant="danger" onClick={disconnect} disabled={!habilitado || !configurado || disconnecting
|| connecting}>
                      Desconectar
                    </Button>
                  </div>
                </div>
              )}
            </div>
          </Card>
        </div>

        {qrBase64 ? (
          <div
            className={
              tutorialOpen && tutorialSteps[tutorialStep]?.target === 'qr'
                ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
```

```
                : ''
            }
          >
            <Card>
              <div className="p-6 space-y-4">
                <div className="text-sm font-semibold text-slate-900">QR Code</div>
                <div className="text-sm text-slate-600">Escaneie no WhatsApp para conectar a instância.</div>
                <div className="flex justify-center">
                  <img
                    className="h-64 w-64 rounded-xl border border-slate-200 bg-white"
                    src={qrBase64.trim().toLowerCase().startsWith('data:') ? qrBase64 :
`data:image/png;base64,${qrBase64}`}
                    alt="QR Code"
                  />
                </div>
              </div>
            </Card>
          </div>
        ) : null}

        <div
          className={
            tutorialOpen && tutorialSteps[tutorialStep]?.target === 'test'
              ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
              : ''
          }
        >
          <Card>
            <div className="p-6 space-y-4">
              <div className="text-sm font-semibold text-slate-900">Testar envio</div>
              <Input label="Número (com DDD)" value={testNumber} onChange={(e) => setTestNumber(e.target.value)}
placeholder="11 99999-9999" />
              <div className="space-y-2">
                <div className="text-sm font-medium text-slate-700">Mensagem</div>
                <textarea
                  className="w-full min-h-[120px] rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-
slate-900 outline-none focus:ring-2 focus:ring-slate-300"
                  value={testText}
                  onChange={(e) => setTestText(e.target.value)}
                  disabled={sendingTest}
                />
              </div>
              <div className="flex justify-end">
                <Button onClick={sendTest} disabled={!habilitado || !configurado || sendingTest || connecting ||
disconnecting}>
                  Enviar teste
                </Button>
              </div>
            </div>
          </Card>
        </div>

            <TutorialOverlay open={tutorialOpen} steps={tutorialSteps} step={tutorialStep} onStepChange=
{setTutorialStep} onClose={closeTutorial} />
          </div>
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/auth/CadastroPage.tsx

```
import { useMemo, useState } from 'react'
import { Link, useNavigate } from 'react-router-dom'
import { supabase } from '../../lib/supabase'
import { supabaseEnv } from '../../lib/supabase'
import { slugify } from '../../lib/slug'
import { Button } from '../../components/ui/Button'
import { Input } from '../../components/ui/Input'
import { useAuth } from '../../state/auth/useAuth'

const MIN_PASSWORD_LENGTH = 11
const TERMS_VERSION = '2026-01-11'
const PRIVACY_VERSION = '2026-01-11'

export function CadastroPage() {
  const navigate = useNavigate()
  const { refresh } = useAuth()

  const [nomeCompleto, setNomeCompleto] = useState('')
  const [nomeNegocio, setNomeNegocio] = useState('')
  const [telefone, setTelefone] = useState('')
  const [email, setEmail] = useState('')
  const [senha, setSenha] = useState('')
  const [slug, setSlug] = useState('')
  const [acceptedLegal, setAcceptedLegal] = useState(false)
  const [submitting, setSubmitting] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [info, setInfo] = useState<string | null>(null)
  const [diagnostico, setDiagnostico] = useState<string | null>(null)

  const derivedSlug = useMemo(() => (slug ? slugify(slug) : slugify(nomeNegocio)), [slug, nomeNegocio])
  const canSubmit = useMemo(
    () =>
      nomeCompleto.trim() &&
      nomeNegocio.trim() &&
      email.trim() &&
      senha.trim().length >= MIN_PASSWORD_LENGTH &&
      derivedSlug.trim() &&
      acceptedLegal,
    [nomeCompleto, nomeNegocio, email, senha, derivedSlug, acceptedLegal]
  )

  const onSubmit = async (e: React.FormEvent) => {
    e.preventDefault()
    setSubmitting(true)
    setError(null)
    setInfo(null)
    setDiagnostico(null)

    if (!acceptedLegal) {
      setError('Você precisa aceitar os Termos de Uso e a Política de Privacidade para criar a conta.')
      setSubmitting(false)
      return
    }

    const cleanEmail = email.trim().toLowerCase()
    const cleanPassword = senha
    const cleanNomeCompleto = nomeCompleto.trim()
    const cleanNomeNegocio = nomeNegocio.trim()
    const cleanTelefone = telefone.trim() || null
    const acceptedAt = new Date().toISOString()
    const userAgent = typeof navigator !== 'undefined' ? navigator.userAgent : ''

    if (!supabaseEnv.ok) {
      setError(`Supabase não configurado. Faltando: ${supabaseEnv.missing.join(', ')}`)
      setSubmitting(false)
      return
    }

    const redirectTo = `${window.location.origin}/login?type=signup`
```

```
    try {
      const res = await fetch(`${supabaseEnv.values.VITE_SUPABASE_URL.replace(/\/$/, '')}/auth/v1/settings`, {
        headers: {
          apikey=[REDACTED]
          Authorization: `Bearer ${supabaseEnv.values.VITE_SUPABASE_ANON_KEY}`,
        },
      })
      if (!res.ok) {
        setError(`Falha ao conectar no Supabase (HTTP ${res.status}). Verifique URL e chave.`)
        setSubmitting(false)
        return
      }
      const json = (await res.json().catch(() => null)) as
        | { disable_signup?: boolean; external?: { email?: boolean }; mailer_autoconfirm?: boolean }
        | null
      if (json) {
        const host = (() => {
          try {
            return new URL(supabaseEnv.values.VITE_SUPABASE_URL).host
          } catch {
            return supabaseEnv.values.VITE_SUPABASE_URL
          }
        })()
        const emailOn = json.external?.email === true
        const signupOff = json.disable_signup === true
        const autoConfirm = json.mailer_autoconfirm === true
        setDiagnostico(
          `Supabase=${host} • Conexão OK • Email=${emailOn ? 'on' : 'off'} • Signup=${signupOff ? 'off' : 'on'} •
AutoConfirm=${autoConfirm ? 'on' : 'off'}${
            emailOn && !autoConfirm
              ? ' • Dica: se o email não chega, configure SMTP (Resend) em Authentication → SMTP Settings e valide em
/admin/configuracoes.'
              : ''
          }`
        )
        if (signupOff) {
          setError('O cadastro está desabilitado no Supabase (disable_signup=true).')
          setSubmitting(false)
          return
        }
        if (!emailOn && !autoConfirm) {
          setError(
            'O provedor de Email está desativado no Supabase. Ative em Authentication → Providers → Email. Depois, tente
novamente.'
          )
          setSubmitting(false)
          return
        }
      }
    } catch {
      setError('Falha de rede ao conectar no Supabase. Verifique sua conexão e a URL do Supabase.')
      setSubmitting(false)
      return
    }
    let signUpRes: Awaited<ReturnType<typeof supabase.auth.signUp>> | null = null
    try {
      signUpRes = await supabase.auth.signUp({
        email: cleanEmail,
        password: cleanPassword,
        options: {
          emailRedirectTo: redirectTo,
          data: {
            nome_completo: cleanNomeCompleto,
            nome_negocio: cleanNomeNegocio,
            telefone: cleanTelefone,
            slug: derivedSlug,
            termos_versao: TERMS_VERSION,
            privacidade_versao: PRIVACY_VERSION,
            legal_aceite_em: acceptedAt,
            legal_user_agent: userAgent,
          },
        },
```

```
      })
    } catch {
      setError('Falha ao comunicar com o Supabase. Verifique as variáveis de ambiente e a rede.')
      setSubmitting(false)
      return
    }
    const data = signUpRes?.data
    const signUpError = signUpRes?.error
    if (!data) {
      setError('Falha ao criar usuário (resposta inválida do Supabase).')
      setSubmitting(false)
      return
    }

    if (signUpError) {
      const msg = signUpError.message
      const lower = msg.toLowerCase()
      const tooShort = lower.includes('password') && lower.includes('at least')
      const alreadyRegistered = lower.includes('already') || lower.includes('registered') || lower.includes('exists')
      const redirectInvalid = lower.includes('redirect') || lower.includes('url')
      const captcha = lower.includes('captcha')
      if (alreadyRegistered) {
        setInfo('Este email já possui conta. Se ainda não recebeu a confirmação, clique em Reenviar confirmação.')
        setSubmitting(false)
        return
      }
      if (redirectInvalid) {
        setError(`O Supabase rejeitou a URL de redirecionamento. Adicione nas Redirect URLs: ${redirectTo}`)
        setSubmitting(false)
        return
      }
      if (captcha) {
        setError('O Supabase está exigindo CAPTCHA no cadastro e o app ainda não envia o token. Desative CAPTCHA no Auth
ou adicione suporte no app.')
        setSubmitting(false)
        return
      }

      setError(tooShort ? `A senha deve ter no mínimo ${MIN_PASSWORD_LENGTH} caracteres.` : msg)
      setSubmitting(false)
      return
    }

    const userId = data.user?.id
    if (!userId) {
      setError('Falha ao criar usuário')
      setSubmitting(false)
      return
    }

    if (!data.session) {
      const confirmationSentAt = (data.user as unknown as { confirmation_sent_at?: string | null } |
null)?.confirmation_sent_at
      if (!confirmationSentAt) {
        const { error: resendErr } = await supabase.auth.resend({ type: 'signup', email: cleanEmail, options: {
emailRedirectTo: redirectTo } })
        if (resendErr) {
          setError(resendErr.message)
          setSubmitting(false)
          return
        }
      }

      setInfo('Conta criada. Enviamos um email de confirmação. Verifique sua caixa de entrada e spam.')
      setSubmitting(false)
      return
    }

    const { data: usuarioRow, error: usuarioErr } = await supabase.from('usuarios').select('id').eq('id',
userId).maybeSingle()
    if (usuarioErr) {
      setError(usuarioErr.message)
```

```
      setSubmitting(false)
      return
    }
    if (!usuarioRow) {
      await supabase.rpc('ensure_usuario_profile')
      const { data: usuarioRow2 } = await supabase.from('usuarios').select('id').eq('id', userId).maybeSingle()
      if (!usuarioRow2) {
        setError('Cadastro criado, mas o perfil do usuário não foi configurado no Supabase.')
        setSubmitting(false)
        return
      }
    }

    const { error: acceptErr } = await supabase.rpc('accept_legal_terms', {
      p_terms_version: TERMS_VERSION,
      p_privacy_version: PRIVACY_VERSION,
    })
    if (acceptErr) {
      const msg = acceptErr.message
      const lower = msg.toLowerCase()
      const missingFn = lower.includes('accept_legal_terms') && (lower.includes('function') || lower.includes('rpc'))
      const missingCols = lower.includes('termos_aceitos_em') || lower.includes('privacidade_aceita_em')

      if (missingFn || missingCols) {
        setInfo(
          'Conta criada. Para registrar o aceite no perfil do usuário, rode o SQL de Termos/Privacidade em /admin/configuracoes (Super Admin).'
        )
      }
    }

    await refresh()
    navigate('/onboarding', { replace: true })
    setSubmitting(false)
  }

  const resendConfirm = async () => {
    setSubmitting(true)
    setError(null)
    setDiagnostico(null)
    if (!supabaseEnv.ok) {
      setError(`Supabase não configurado. Faltando: ${supabaseEnv.missing.join(', ')}`)
      setSubmitting(false)
      return
    }
    const cleanEmail = email.trim().toLowerCase()
    const redirectTo = `${window.location.origin}/login?type=signup`

    try {
      const res = await fetch(`${supabaseEnv.values.VITE_SUPABASE_URL.replace(/\/$/, '')}/auth/v1/settings`, {
        headers: {
          apikey=[REDACTED]
          Authorization: `Bearer ${supabaseEnv.values.VITE_SUPABASE_ANON_KEY}`,
        },
      })
      const json = (await res.json().catch(() => null)) as
        | { external?: { email?: boolean }; mailer_autoconfirm?: boolean }
        | null
      const emailOn = json?.external?.email === true
      const autoConfirm = json?.mailer_autoconfirm === true
      if (emailOn && !autoConfirm) {
        setDiagnostico('Dica: se o email não chega, configure SMTP (Resend) em Authentication → SMTP Settings e valide em /admin/configuracoes.')
      }
      if (!emailOn && !autoConfirm) {
        setError('O provedor de Email está desativado no Supabase. Ative em Authentication → Providers → Email.')
        setSubmitting(false)
        return
      }
    } catch {
      setError('Falha de rede ao verificar o Supabase. Verifique URL e chave.')
      setSubmitting(false)
```

```
      return
    }

    const { error: resendErr } = await supabase.auth.resend({ type: 'signup', email: cleanEmail, options: {
emailRedirectTo: redirectTo } })
    if (resendErr) {
      setError(resendErr.message)
      setSubmitting(false)
      return
    }
    setInfo('Email de confirmação reenviado. Verifique sua caixa de entrada e spam.')
    setSubmitting(false)
  }

  const canResend = useMemo(() => Boolean(email.trim()) && (info !== null || error?.toLowerCase().includes('confirm')
=== true), [email, error, info])

  return (
    <div className="min-h-screen bg-slate-50 flex items-center justify-center px-4">
      <div className="w-full max-w-md">
        <div className="mb-6 text-center">
          <div className="text-2xl font-semibold text-slate-900">Criar conta</div>
          <div className="text-sm text-slate-600">Setup em menos de 10 minutos</div>
        </div>

        <form onSubmit={onSubmit} className="rounded-2xl border border-slate-200 bg-white p-6 shadow-sm">
          <div className="space-y-4">
            <Input label="Nome completo" value={nomeCompleto} onChange={(e) => setNomeCompleto(e.target.value)} />
            <Input label="Nome do negócio" value={nomeNegocio} onChange={(e) => setNomeNegocio(e.target.value)} />
            <Input label="Telefone (com WhatsApp)" value={telefone} onChange={(e) => setTelefone(e.target.value)} />
            <Input label="Email" value={email} onChange={(e) => setEmail(e.target.value)} type="email"
autoComplete="email" />
            <div className="space-y-1">
              <Input
                label="Senha"
                value={senha}
                onChange={(e) => setSenha(e.target.value)}
                type="password"
                autoComplete="new-password"
                minLength={MIN_PASSWORD_LENGTH}
              />
              <div className="text-xs text-slate-600">Mínimo: {MIN_PASSWORD_LENGTH} caracteres.</div>
            </div>
            <Input label="Slug (editável)" value={slug} onChange={(e) => setSlug(e.target.value)} placeholder=
{derivedSlug} />
            <div className="text-xs text-slate-600">Seu link ficará: /agendar/{derivedSlug}</div>

            <label className="flex items-start gap-2 text-sm text-slate-700">
              <input
                type="checkbox"
                className="mt-1"
                checked={acceptedLegal}
                onChange={(e) => setAcceptedLegal(e.target.checked)}
              />
              <span>
                Eu li e aceito os{' '}
                <a className="font-medium text-slate-900 hover:underline" href="/termos" target="_blank"
rel="noreferrer">
                  Termos de Uso
                </a>{' '}
                e a{' '}
                <a className="font-medium text-slate-900 hover:underline" href="/privacidade" target="_blank"
rel="noreferrer">
                  Política de Privacidade
                </a>
                .
              </span>
            </label>

            {info ? <div className="text-sm text-emerald-700">{info}</div> : null}
            {diagnostico ? <div className="text-xs text-slate-600">{diagnostico}</div> : null}
            {error ? <div className="text-sm text-rose-600">{error}</div> : null}
```

```
            <Button type="submit" fullWidth disabled={!canSubmit || submitting}>
              Criar conta grátis
            </Button>

            {canResend ? (
              <Button type="button" fullWidth variant="secondary" disabled={!email.trim() || submitting} onClick=
{resendConfirm}>
                Reenviar confirmação
              </Button>
            ) : null}
          </div>

          <div className="mt-4 text-center text-sm text-slate-600">
            Já tem conta?{' '}
            <Link to="/login" className="font-medium text-slate-900 hover:underline">
              Entrar
            </Link>
          </div>
        </form>
      </div>
    </div>
  )
}
```

smagenda/src/views/auth/ForgotPasswordPage.tsx

```
import { useMemo, useState } from 'react'
import { Link } from 'react-router-dom'
import { Button } from '../../components/ui/Button'
import { Input } from '../../components/ui/Input'
import { supabase } from '../../lib/supabase'

export function ForgotPasswordPage() {
  const [email, setEmail] = useState('')
  const [submitting, setSubmitting] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [sent, setSent] = useState(false)

  const canSubmit = useMemo(() => email.trim(), [email])

  const onSubmit = async (e: React.FormEvent) => {
    e.preventDefault()
    setSubmitting(true)
    setError(null)
    setSent(false)

    const redirectTo = `${window.location.origin}/resetar-senha`
    const { error: err } = await supabase.auth.resetPasswordForEmail(email.trim(), { redirectTo })
    if (err) {
      setError(err.message)
      setSubmitting(false)
      return
    }

    setSubmitting(false)
    setSent(true)
  }

  return (
    <div className="min-h-screen bg-slate-50 flex items-center justify-center px-4">
      <div className="w-full max-w-md">
        <div className="mb-6 text-center">
          <div className="text-2xl font-semibold text-slate-900">Recuperar senha</div>
          <div className="text-sm text-slate-600">Envie um link para criar uma nova senha</div>
        </div>

        <form onSubmit={onSubmit} className="rounded-2xl border border-slate-200 bg-white p-6 shadow-sm">
          <div className="space-y-4">
            <Input label="Email" value={email} onChange={(e) => setEmail(e.target.value)} type="email"
autoComplete="email" />

            {error ? <div className="text-sm text-rose-600">{error}</div> : null}
            {sent ? (
              <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">
                Enviamos um link para seu email. Verifique a caixa de entrada e spam.
              </div>
            ) : null}

            <Button type="submit" fullWidth disabled={!canSubmit || submitting}>
              Enviar link
            </Button>
          </div>

          <div className="mt-4 text-center text-sm text-slate-600">
            <Link to="/login" className="font-medium text-slate-900 hover:underline">
              Voltar para login
            </Link>
          </div>
        </form>
      </div>
    </div>
  )
}
```

smagenda/src/views/auth/LoginPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { Link, useNavigate } from 'react-router-dom'
import { supabase } from '../../lib/supabase'
import { supabaseEnv } from '../../lib/supabase'
import { Button } from '../../components/ui/Button'
import { Input } from '../../components/ui/Input'
import { useAuth } from '../../state/auth/useAuth'

export function LoginPage() {
  const navigate = useNavigate()
  const { refresh } = useAuth()
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')
  const [submitting, setSubmitting] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [info, setInfo] = useState<string | null>(null)

  const diagnoseProfile = async () => {
    const { data: userData, error: userErr } = await supabase.auth.getUser()
    if (userErr) return `auth.getUser: ${userErr.message}`
    const uid = userData.user?.id ?? null
    if (!uid) return 'Sessão não encontrada após login.'

    const check = async (table: 'usuarios' | 'funcionarios' | 'super_admin') => {
      const { data, error: err } = await supabase.from(table).select('id').eq('id', uid).maybeSingle()
      return { ok: !err, hasRow: Boolean((data as unknown as { id?: string | null } | null)?.id), err: err?.message ??
null }
    }

    const [u, f, s] = await Promise.all([check('usuarios'), check('funcionarios'), check('super_admin')])

    const firstErr = u.err ?? f.err ?? s.err
    if (firstErr) {
      const lower = firstErr.toLowerCase()
      const missingTable = lower.includes('could not find the table') || lower.includes('schema cache')
      if (missingTable) {
        return 'Tabelas SQL não configuradas no Supabase (public.usuarios / public.funcionarios / public.super_admin).'
      }

      const permission = lower.includes('permission denied') || lower.includes('row-level security') ||
lower.includes('rls')
      if (permission) {
        return 'Sem permissão para ler seu perfil na tabela. Execute o bloco “SQL de políticas (Usuário / Funcionário)”
em /admin/configuracoes no SQL Editor do Supabase.'
      }

      return firstErr
    }

    if (!u.hasRow && !f.hasRow && !s.hasRow) {
      return 'Usuário existe no Auth, mas não existe registro nas tabelas (usuarios/funcionarios/super_admin). Execute
“SQL de Trial” e depois faça login novamente.'
    }

    return null
  }

  useEffect(() => {
    const run = async () => {
      const url = new URL(window.location.href)
      const code = url.searchParams.get('code')
      const tokenHash = url.searchParams.get('token_hash')
      const type = url.searchParams.get('type')

      const hashParams = url.hash.startsWith('#') ? new URLSearchParams(url.hash.slice(1)) : null
      const accessToken=[REDACTED]'access_token')
      const refreshToken = hashParams?.get('refresh_token')
      const hashType = hashParams?.get('type')

      const hasCallback = Boolean(code || tokenHash)
```

```
      const hasHashSession = Boolean(accessToken && refreshToken)
      if (!hasCallback && !hasHashSession) return

      setSubmitting(true)
      setError(null)

      const finalType = type ?? hashType

      if (code) {
        const { error: exchangeErr } = await supabase.auth.exchangeCodeForSession(code)
        if (exchangeErr) {
          setError(exchangeErr.message)
          setSubmitting(false)
          return
        }
      } else if (hasHashSession && accessToken && refreshToken) {
        const { error: setErr } = await supabase.auth.setSession({ access_token=[REDACTED]
        if (setErr) {
          setError(setErr.message)
          setSubmitting(false)
          return
        }
      } else if (tokenHash && finalType) {
        const { error: verifyErr } = await supabase.auth.verifyOtp({
          type: finalType as 'signup' | 'magiclink' | 'recovery' | 'email_change' | 'invite',
          token_hash: tokenHash,
        })
        if (verifyErr) {
          setError(verifyErr.message)
          setSubmitting(false)
          return
        }
      }

      const { error: ensureErr } = await supabase.rpc('ensure_usuario_profile')
      if (ensureErr) {
        const msg = ensureErr.message
        const lower = msg.toLowerCase()
        const missingFn = lower.includes('ensure_usuario_profile') && (lower.includes('function') ||
lower.includes('rpc'))
        if (missingFn) {
          setError('Configuração do Supabase incompleta: crie a função ensure_usuario_profile (trial).')
          setSubmitting(false)
          return
        }
        const slugTaken = lower.includes('slug') && (lower.includes('duplicate') || lower.includes('unique'))
        if (slugTaken) {
          setError('Seu link (slug) já está em uso. Tente criar a conta novamente com outro slug.')
          setSubmitting(false)
          return
        }
      }

      const next = await refresh()
      if (!next) {
        const diag = await diagnoseProfile()
        setError(
          diag
            ? `Não foi possível carregar seu perfil. ${diag}`
            : 'Não foi possível carregar seu perfil. Verifique se as políticas (RLS) e tabelas do Supabase estão
configuradas.'
        )
        setSubmitting(false)
        return
      }
      if (next.kind === 'funcionario') {
        navigate('/funcionario/agenda', { replace: true })
      } else if (next.kind === 'super_admin') {
        navigate('/admin/dashboard', { replace: true })
      } else {
        navigate(finalType === 'signup' ? '/onboarding' : '/dashboard', { replace: true })
      }
```

```
      setSubmitting(false)
    }

    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Falha ao confirmar email')
      setSubmitting(false)
    })
  }, [navigate, refresh])

  const canSubmit = useMemo(() => email.trim() && password.trim(), [email, password])

  const onSubmit = async (e: React.FormEvent) => {
    e.preventDefault()
    setSubmitting(true)
    setError(null)
    setInfo(null)
    const { error: signInError } = await supabase.auth.signInWithPassword({ email: email.trim().toLowerCase(), password
})
    if (signInError) {
      const msg = signInError.message
      const notConfirmed = msg.toLowerCase().includes('confirm')
      setError(notConfirmed ? 'Email não confirmado. Confirme no email ou solicite um novo link de confirmação.' : msg)
      setSubmitting(false)
      return
    }

    const { error: ensureErr } = await supabase.rpc('ensure_usuario_profile')
    if (ensureErr) {
      const msg = ensureErr.message
      const lower = msg.toLowerCase()
      const missingFn = lower.includes('ensure_usuario_profile') && (lower.includes('function') ||
lower.includes('rpc'))
      if (missingFn) {
        setError('Configuração do Supabase incompleta: crie a função ensure_usuario_profile (trial).')
        setSubmitting(false)
        return
      }
      const slugTaken = lower.includes('slug') && (lower.includes('duplicate') || lower.includes('unique'))
      if (slugTaken) {
        setError('Seu link (slug) já está em uso. Peça para alterar o slug no painel de admin.')
        setSubmitting(false)
        return
      }
    }

    const next = await refresh()
    if (!next) {
      const diag = await diagnoseProfile()
      setError(
        diag ? `Não foi possível carregar seu perfil. ${diag}` : 'Não foi possível carregar seu perfil. Verifique se as
políticas (RLS) e tabelas do Supabase estão configuradas.'
      )
      setSubmitting(false)
      return
    }
    if (next.kind === 'funcionario') {
      navigate('/funcionario/agenda', { replace: true })
    } else if (next.kind === 'super_admin') {
      navigate('/admin/dashboard', { replace: true })
    } else {
      navigate('/dashboard', { replace: true })
    }
    setSubmitting(false)
  }

  return (
    <div className="min-h-screen bg-slate-50 flex items-center justify-center px-4">
      <div className="w-full max-w-md">
        <div className="mb-6 text-center">
          <div className="text-2xl font-semibold text-slate-900">SMagenda</div>
          <div className="text-sm text-slate-600">Entre para acessar sua agenda</div>
        </div>
```

```
        <form onSubmit={onSubmit} className="rounded-2xl border border-slate-200 bg-white p-6 shadow-sm">
          <div className="space-y-4">
            <Input label="Email" value={email} onChange={(e) => setEmail(e.target.value)} type="email" autoComplete="email" />
            <Input
              label="Senha"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              type="password"
              autoComplete="current-password"
            />
            {info ? <div className="text-sm text-emerald-700">{info}</div> : null}
            {error ? <div className="text-sm text-rose-600">{error}</div> : null}
            <Button type="submit" fullWidth disabled={!canSubmit || submitting}>
              Entrar
            </Button>
            {error?.toLowerCase().includes('confirm') === true ? (
              <Button
                type="button"
                fullWidth
                variant="secondary"
                disabled={!email.trim() || submitting}
                onClick={async () => {
                  setSubmitting(true)
                  setError(null)
                  setInfo(null)
                  if (!supabaseEnv.ok) {
                    setError(`Supabase não configurado. Faltando: ${supabaseEnv.missing.join(', ')}`)
                    setSubmitting(false)
                    return
                  }
                  const cleanEmail = email.trim().toLowerCase()
                  const redirectTo = `${window.location.origin}/login?type=signup`

                  try {
                    const res = await fetch(`${supabaseEnv.values.VITE_SUPABASE_URL.replace(/\/$/, '')}/auth/v1/settings`, {
                      headers: {
                        apikey=[REDACTED]
                        Authorization: `Bearer ${supabaseEnv.values.VITE_SUPABASE_ANON_KEY}`,
                      },
                    })
                    const json = (await res.json().catch(() => null)) as
                      | { external?: { email?: boolean }; mailer_autoconfirm?: boolean }
                      | null
                    const emailOn = json?.external?.email === true
                    const autoConfirm = json?.mailer_autoconfirm === true
                    if (!emailOn && !autoConfirm) {
                      setError('O provedor de Email está desativado no Supabase. Ative em Authentication → Providers → Email.')

                      setSubmitting(false)
                      return
                    }
                  } catch {
                    setError('Falha de rede ao verificar o Supabase. Verifique URL e chave.')
                    setSubmitting(false)
                    return
                  }

                  const { error: resendErr } = await supabase.auth.resend({
                    type: 'signup',
                    email: cleanEmail,
                    options: { emailRedirectTo: redirectTo },
                  })
                  if (resendErr) {
                    setError(resendErr.message)
                    setSubmitting(false)
                    return
                  }
                  setInfo('Email de confirmação reenviado. Verifique sua caixa de entrada e spam.')
                  setSubmitting(false)
```

```
        }}
       >
        Reenviar confirmação
       </Button>
      ) : null}
      <div className="text-center text-sm text-slate-600">
       <Link to="/esqueci-senha" className="font-medium text-slate-900 hover:underline">
        Esqueci minha senha
       </Link>
      </div>
     </div>

     <div className="mt-4 text-center text-sm text-slate-600">
      Não tem conta?{' '}
      <Link to="/cadastro" className="font-medium text-slate-900 hover:underline">
       Criar conta
      </Link>
     </div>
    </form>
   </div>
  </div>
 )
}
```

smagenda/src/views/auth/OnboardingPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { supabase } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type Step = 1 | 2 | 3 | 4

const weekdayOptions = [
  { value: 1, label: 'S' },
  { value: 2, label: 'T' },
  { value: 3, label: 'Q' },
  { value: 4, label: 'Q' },
  { value: 5, label: 'S' },
  { value: 6, label: 'S' },
  { value: 0, label: 'D' },
]

export function OnboardingPage() {
  const { appPrincipal, masterUsuario, masterUsuarioLoading, refresh } = useAuth()
  const isGerente = appPrincipal?.kind === 'funcionario' && appPrincipal.profile.permissao === 'admin'
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : isGerente ? masterUsuario : null
  const usuarioId = usuario?.id ?? null
  const slug = usuario?.slug ?? null
  const initialLogoUrl = usuario?.logo_url ?? null

  const [step, setStep] = useState<Step>(1)
  const [horarioInicio, setHorarioInicio] = useState('08:00')
  const [horarioFim, setHorarioFim] = useState('18:00')
  const [dias, setDias] = useState<number[]>([1, 2, 3, 4, 5, 6])
  const [intervaloInicio, setIntervaloInicio] = useState('')
  const [intervaloFim, setIntervaloFim] = useState('')
  const [logoUrl, setLogoUrl] = useState(() => initialLogoUrl ?? '')
  const [logoFile, setLogoFile] = useState<File | null>(null)
  const [logoFailed, setLogoFailed] = useState(false)

  const [servicoNome, setServicoNome] = useState('')
  const [servicoDuracao, setServicoDuracao] = useState('45')
  const [servicoPreco, setServicoPreco] = useState('')
  const [servicoCor, setServicoCor] = useState('#0f172a')

  const [submitting, setSubmitting] = useState(false)
  const [error, setError] = useState<string | null>(null)

  const linkPublico = useMemo(() => (slug ? `${window.location.origin}/agendar/${slug}` : ''), [slug])

  const logoObjectUrl = useMemo(() => (logoFile ? URL.createObjectURL(logoFile) : null), [logoFile])

  useEffect(() => {
    if (!logoObjectUrl) return
    return () => {
      URL.revokeObjectURL(logoObjectUrl)
    }
  }, [logoObjectUrl])

  if (!usuarioId) {
    return <div className="text-slate-700">{isGerente && masterUsuarioLoading ? 'Carregando…' : 'Conta inválida para
onboarding.'}</div>
  }

  const toggleDia = (day: number) => {
    setDias((prev) => (prev.includes(day) ? prev.filter((d) => d !== day) : [...prev, day]))
  }

  const uploadLogo = async (file: File) => {
    const safeName = file.name
      .trim()
      .replace(/\s+/g, '-')
      .replace(/[^a-zA-Z0-9._-]/g, '')
```

```
      .slice(0, 80)

    const key = `${usuarioId}/logo-${Date.now()}-${safeName || 'logo'}`

    const { error: uploadErr } = await supabase.storage.from('logos').upload(key, file, {
      upsert: true,
      contentType: file.type || undefined,
    })
    if (uploadErr) throw uploadErr

    const { data } = supabase.storage.from('logos').getPublicUrl(key)
    const publicUrl = data?.publicUrl
    if (!publicUrl) throw new Error('Não foi possível obter a URL pública do logo')
    return publicUrl
  }

  const saveStep1 = async () => {
    setSubmitting(true)
    setError(null)

    let nextLogoUrl: string | null = logoUrl.trim() ? logoUrl.trim() : null
    if (logoFile) {
      try {
        const publicUrl = await uploadLogo(logoFile)
        nextLogoUrl = publicUrl
        setLogoUrl(publicUrl)
        setLogoFile(null)
      } catch (e: unknown) {
        const msg = e instanceof Error ? e.message : 'Falha ao enviar logo'
        const lower = msg.toLowerCase()
        const missingBucket = lower.includes('bucket') || lower.includes('not found')
        const rls = lower.includes('row-level security') || lower.includes('row level security')
        setError(
          missingBucket
            ? 'Configuração do Supabase incompleta: crie o bucket "logos" no Storage e habilite leitura pública + upload
do próprio usuário.'
            : rls
              ? 'Sem permissão para enviar ao Storage. Execute o SQL do Storage (logos) em /admin/configuracoes.'
              : msg
        )
        setSubmitting(false)
        return
      }
    }

    const { error: updateError } = await supabase
      .from('usuarios')
      .update({
        horario_inicio: horarioInicio,
        horario_fim: horarioFim,
        dias_trabalho: dias,
        intervalo_inicio: intervaloInicio || null,
        intervalo_fim: intervaloFim || null,
        logo_url: nextLogoUrl,
      })
      .eq('id', usuarioId)
    if (updateError) {
      const msg = updateError.message
      const lower = msg.toLowerCase()
      const rls = lower.includes('row-level security') || lower.includes('row level security')
      setError(rls ? 'Sem permissão para atualizar seu perfil (RLS). Execute o SQL de políticas (Usuário / Funcionário)
em /admin/configuracoes.' : msg)
      setSubmitting(false)
      return
    }
    await refresh()
    setSubmitting(false)
    setStep(2)
  }

  const saveStep2 = async () => {
    setSubmitting(true)
```

```
    setError(null)
    const { error: insertError } = await supabase.from('servicos').insert({
      usuario_id: usuarioId,
      nome: servicoNome,
      duracao_minutos: Number(servicoDuracao),
      preco: servicoPreco ? Number(servicoPreco) : 0,
      cor: servicoCor,
      ativo: true,
      ordem: 0,
    })
    if (insertError) {
      setError(insertError.message)
      setSubmitting(false)
      return
    }
    setSubmitting(false)
    setStep(3)
  }

  const saveStep3 = async () => {
    setSubmitting(true)
    setError(null)
    setSubmitting(false)
    setStep(4)
  }

  const copyLink = async () => {
    await navigator.clipboard.writeText(linkPublico)
  }

  return (
    <div className="space-y-6">
      <div>
        <div className="text-sm font-semibold text-slate-500">Onboarding</div>
        <div className="text-xl font-semibold text-slate-900">Configurar sua conta</div>
      </div>

      {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> :
null}

      {step === 1 ? (
        <Card>
          <div className="p-6 space-y-4">
            <div className="text-sm font-semibold text-slate-900">Etapa 1: Horário de funcionamento</div>
            <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
              <Input label="Início" type="time" value={horarioInicio} onChange={(e) => setHorarioInicio(e.target.value)}
/>
              <Input label="Fim" type="time" value={horarioFim} onChange={(e) => setHorarioFim(e.target.value)} />
            </div>
            <div>
              <div className="text-sm font-medium text-slate-700 mb-2">Dias de trabalho</div>
              <div className="flex flex-wrap gap-2">
                {weekdayOptions.map((d) => (
                  <button
                    type="button"
                    key={d.value}
                    onClick={() => toggleDia(d.value)}
                    className={[
                      'h-9 w-9 rounded-lg border text-sm font-semibold',
                      dias.includes(d.value) ? 'bg-slate-900 text-white border-slate-900' : 'bg-white text-slate-700
border-slate-200',
                    ].join(' ')}
                  >
                    {d.label}
                  </button>
                ))}
              </div>
            </div>
            <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
              <Input label="Intervalo (opcional) início" type="time" value={intervaloInicio} onChange={(e) =>
setIntervaloInicio(e.target.value)} />
              <Input label="Intervalo (opcional) fim" type="time" value={intervaloFim} onChange={(e) =>
```

```
setIntervaloFim(e.target.value)} />
            </div>

            {(logoObjectUrl || logoUrl.trim()) && !logoFailed ? (
              <div className="flex items-center justify-between gap-3 rounded-xl border border-slate-200 bg-white p-3">
                <div className="flex items-center gap-3">
                  <img
                    src={logoObjectUrl ?? logoUrl.trim()}
                    alt="Logo"
                    className="h-12 w-12 rounded-xl object-cover border border-slate-200"
                    onError={() => setLogoFailed(true)}
                  />
                  <div className="text-sm text-slate-700">Logo atual</div>
                </div>
                <Button
                  variant="secondary"
                  type="button"
                  onClick={() => {
                    setLogoFile(null)
                    setLogoUrl('')
                    setLogoFailed(false)
                  }}
                >
                  Remover
                </Button>
              </div>
            ) : null}

            <Input
              label="Logo (imagem opcional)"
              type="file"
              accept="image/*"
              onChange={(e) => {
                const file = e.target.files?.[0] ?? null
                if (!file) {
                  setLogoFile(null)
                  setLogoFailed(false)
                  return
                }
                if (!file.type.startsWith('image/')) {
                  setError('Selecione um arquivo de imagem (PNG/JPG/WebP).')
                  setLogoFile(null)
                  return
                }
                const maxBytes = 2 * 1024 * 1024
                if (file.size > maxBytes) {
                  setError('Imagem muito grande (máx 2MB).')
                  setLogoFile(null)
                  return
                }
                setError(null)
                setLogoFailed(false)
                setLogoFile(file)
              }}
            />
            <div className="flex justify-end">
              <Button onClick={saveStep1} disabled={submitting}>
                Continuar
              </Button>
            </div>
          </div>
        </Card>
      ) : null}

      {step === 2 ? (
        <Card>
          <div className="p-6 space-y-4">
            <div className="text-sm font-semibold text-slate-900">Etapa 2: Primeiro serviço</div>
            <Input label="Nome do serviço" value={servicoNome} onChange={(e) => setServicoNome(e.target.value)} />
            <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
              <Input label="Duração (min)" type="number" value={servicoDuracao} onChange={(e) =>
setServicoDuracao(e.target.value)} />
```

```
              <Input label="Preço" type="number" value={servicoPreco} onChange={(e) => setServicoPreco(e.target.value)}
/>
            </div>
            <Input label="Cor" type="color" value={servicoCor} onChange={(e) => setServicoCor(e.target.value)} />
            <div className="flex justify-end">
              <Button onClick={saveStep2} disabled={submitting || !servicoNome.trim()}>
                Continuar
              </Button>
            </div>
          </div>
        </Card>
      ) : null}

      {step === 3 ? (
        <Card>
          <div className="p-6 space-y-4">
            <div className="text-sm font-semibold text-slate-900">Etapa 3: WhatsApp</div>
            <div className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-sm text-slate-700">
              A configuração da Evolution API é feita no painel do Super Admin.
            </div>

            <div className="flex justify-end">
              <Button onClick={saveStep3} disabled={submitting}>
                Continuar
              </Button>
            </div>
          </div>
        </Card>
      ) : null}

      {step === 4 ? (
        <Card>
          <div className="p-6 space-y-4">
            <div className="text-sm font-semibold text-slate-900">Etapa 4: Pronto</div>
            <div className="rounded-lg border border-slate-200 bg-slate-50 p-3 text-sm text-slate-700 break-all">
{linkPublico}</div>
            <div className="flex gap-3">
              <Button variant="secondary" onClick={copyLink}>
                Copiar link
              </Button>
              <a
                className="inline-flex items-center justify-center rounded-lg px-4 py-2 text-sm font-medium bg-slate-900
text-white hover:bg-slate-800"
                href={`https://wa.me/?text=${encodeURIComponent(linkPublico)}`}
                target="_blank"
              >
                Compartilhar no WhatsApp
              </a>
            </div>
            <div className="flex justify-end">
              <Button onClick={() => (window.location.href = '/dashboard')}>Ir para minha agenda</Button>
            </div>
          </div>
        </Card>
      ) : null}
    </div>
  )
}
```

smagenda/src/views/auth/ResetPasswordPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { Link, useNavigate } from 'react-router-dom'
import { Button } from '../../components/ui/Button'
import { Input } from '../../components/ui/Input'
import { supabase } from '../../lib/supabase'

export function ResetPasswordPage() {
  const navigate = useNavigate()

  const [password, setPassword] = useState('')
  const [confirm, setConfirm] = useState('')
  const [submitting, setSubmitting] = useState(false)
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const [success, setSuccess] = useState(false)

  const canSubmit = useMemo(() => password.trim().length >= 8 && password === confirm, [password, confirm])

  useEffect(() => {
    const run = async () => {
      const url = new URL(window.location.href)
      const code = url.searchParams.get('code')
      if (code) {
        const { error: exchangeErr } = await supabase.auth.exchangeCodeForSession(code)
        if (exchangeErr) {
          setError(exchangeErr.message)
          setLoading(false)
          return
        }
      }

      const { data } = await supabase.auth.getSession()
      if (!data.session) {
        setError('Link inválido ou expirado. Solicite um novo link de recuperação.')
        setLoading(false)
        return
      }

      setLoading(false)
    }

    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao validar link')
      setLoading(false)
    })
  }, [])

  const onSubmit = async (e: React.FormEvent) => {
    e.preventDefault()
    if (!canSubmit) return
    setSubmitting(true)
    setError(null)

    const { error: updateErr } = await supabase.auth.updateUser({ password })
    if (updateErr) {
      setError(updateErr.message)
      setSubmitting(false)
      return
    }

    setSubmitting(false)
    setSuccess(true)

    setTimeout(() => {
      navigate('/login', { replace: true })
    }, 800)
  }

  return (
    <div className="min-h-screen bg-slate-50 flex items-center justify-center px-4">
```

```
      <div className="w-full max-w-md">
        <div className="mb-6 text-center">
          <div className="text-2xl font-semibold text-slate-900">Criar nova senha</div>
          <div className="text-sm text-slate-600">Defina uma nova senha para sua conta</div>
        </div>

        <form onSubmit={onSubmit} className="rounded-2xl border border-slate-200 bg-white p-6 shadow-sm">
          <div className="space-y-4">
            {loading ? <div className="text-sm text-slate-600">Validando link…</div> : null}
            {!loading ? (
              <>
                <Input
                  label="Nova senha"
                  value={password}
                  onChange={(e) => setPassword(e.target.value)}
                  type="password"
                  autoComplete="new-password"
                />
                <Input
                  label="Confirmar senha"
                  value={confirm}
                  onChange={(e) => setConfirm(e.target.value)}
                  type="password"
                  autoComplete="new-password"
                />
              </>
            ) : null}

            {error ? <div className="text-sm text-rose-600">{error}</div> : null}
            {success ? (
              <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">
                Senha atualizada. Redirecionando…
              </div>
            ) : null}

            <Button type="submit" fullWidth disabled={loading || !canSubmit || submitting}>
              Salvar nova senha
            </Button>
          </div>

          <div className="mt-4 text-center text-sm text-slate-600">
            <Link to="/esqueci-senha" className="font-medium text-slate-900 hover:underline">
              Solicitar novo link
            </Link>
          </div>
        </form>
      </div>
    </div>
  )
}
```

smagenda/src/views/public/AjudaPage.tsx

```
import { Link } from 'react-router-dom'
import { AppShell } from '../../components/layout/AppShell'
import { getOptionalEnv } from '../../lib/env'
import { useAuth } from '../../state/auth/useAuth'

function normalizeWhatsApp(value: string) {
  const digits = value.replace(/\D+/g, '')
  if (!digits) return null
  if (digits.startsWith('55')) return digits
  if (digits.length === 10 || digits.length === 11) return `55${digits}`
  return digits
}

export function AjudaPage() {
  const { appPrincipal } = useAuth()

  const inApp = appPrincipal?.kind === 'usuario' || appPrincipal?.kind === 'funcionario'
  const backTo = inApp ? (appPrincipal?.kind === 'funcionario' ? '/funcionario/agenda' : '/dashboard') : '/login'

  const supportEmail = (getOptionalEnv('VITE_SUPORTE_EMAIL') ?? getOptionalEnv('VITE_SUPPORT_EMAIL') ??
'suporte@smagenda.com').trim()
  const supportWhatsAppRaw = (
    getOptionalEnv('VITE_SUPORTE_WHATSAPP') ??
    getOptionalEnv('VITE_SUPPORT_WHATSAPP_NUMBER') ??
    getOptionalEnv('VITE_SUPPORT_WHATSAPP') ??
    '(31) 9 7518-4428'
  ).trim()
  const supportWhatsAppDigits = supportWhatsAppRaw ? normalizeWhatsApp(supportWhatsAppRaw) : null

  const waLink = supportWhatsAppDigits
    ? `https://wa.me/${supportWhatsAppDigits}?text=${encodeURIComponent('Olá! Preciso de ajuda com o SMagenda.')}`
    : null

  const content = (
    <div className={inApp ? '' : 'min-h-screen bg-slate-50 px-4 py-10'}>
      <div className={inApp ? 'mx-auto w-full max-w-3xl space-y-6' : 'mx-auto w-full max-w-3xl space-y-6'}>
        <div className="rounded-2xl border border-slate-200 bg-white p-6 shadow-sm">
          <div className="space-y-1">
            <div className="text-2xl font-semibold text-slate-900">Ajuda e Suporte</div>
            <div className="text-sm text-slate-600">Contato, termos e dúvidas frequentes.</div>
          </div>

          <div className="mt-6 space-y-4">
            <div className="rounded-xl border border-slate-200 bg-white p-4">
              <div className="text-sm font-semibold text-slate-900">Fale com a gente</div>
              <div className="mt-2 space-y-2 text-sm text-slate-700">
                {waLink ? (
                  <a className="inline-flex items-center gap-2 font-medium text-slate-900 hover:underline" href={waLink}
target="_blank" rel="noreferrer">
                    <span>WhatsApp:</span>
                    <span className="font-semibold text-slate-900">{supportWhatsAppRaw}</span>
                  </a>
                ) : (
                  <div className="text-slate-600">WhatsApp do suporte não configurado.</div>
                )}

                <a className="block font-medium text-slate-900 hover:underline" href={`mailto:${supportEmail}`}>
                  Email: {supportEmail}
                </a>
              </div>
            </div>

            <div className="rounded-xl border border-slate-200 bg-white p-4">
              <div className="text-sm font-semibold text-slate-900">Documentos</div>
              <div className="mt-2 flex flex-wrap gap-3 text-sm">
                <Link to="/termos" className="font-medium text-slate-900 hover:underline">
                  Termos de Uso
                </Link>
                <Link to="/privacidade" className="font-medium text-slate-900 hover:underline">
                  Política de Privacidade
```

```
                </Link>
              </div>
            </div>

            <div className="rounded-xl border border-slate-200 bg-white p-4">
              <div className="text-sm font-semibold text-slate-900">Perguntas frequentes</div>
              <div className="mt-3 space-y-3">
                <details className="rounded-xl border border-slate-200 bg-slate-50 p-3">
                  <summary className="cursor-pointer text-sm font-semibold text-slate-900">Como o cliente agenda pelo
link público?</summary>
                  <div className="mt-2 text-sm text-slate-700">
                    Você configura os serviços e horários; depois compartilha o link /agendar/SEU-SLUG. O cliente
escolhe serviço, dia e horário e confirma.
                  </div>
                </details>

                <details className="rounded-xl border border-slate-200 bg-slate-50 p-3">
                  <summary className="cursor-pointer text-sm font-semibold text-slate-900">Não estou recebendo email de
confirmação do Supabase</summary>
                  <div className="mt-2 text-sm text-slate-700">
                    Verifique spam e configure SMTP em Authentication → SMTP Settings. No painel admin do SMagenda
existe uma área de validação do Resend.
                  </div>
                </details>

                <details className="rounded-xl border border-slate-200 bg-slate-50 p-3">
                  <summary className="cursor-pointer text-sm font-semibold text-slate-900">Como funcionam serviços de
dia inteiro?</summary>
                  <div className="mt-2 text-sm text-slate-700">
                    Marque o serviço como “dia inteiro” e defina a capacidade diária (1 ou 2). No link público, o
cliente escolhe a data em um calendário com dias disponíveis.
                  </div>
                </details>

                <details className="rounded-xl border border-slate-200 bg-slate-50 p-3">
                  <summary className="cursor-pointer text-sm font-semibold text-slate-900">Onde vejo e altero os
horários de trabalho?</summary>
                  <div className="mt-2 text-sm text-slate-700">
                    No painel do usuário você configura horários base. Para funcionários, cada profissional pode ajustar
seus horários na própria agenda.
                  </div>
                </details>
              </div>
            </div>
          </div>
        </div>

        <div className="text-center text-sm text-slate-600">
          <Link to={backTo} className="hover:underline">
            Voltar
          </Link>
        </div>
      </div>
    </div>
  )

  return inApp ? <AppShell>{content}</AppShell> : content
}
```

smagenda/src/views/public/LandingPage.tsx

```
import { useCallback, useMemo, useState } from 'react'
import { Link, Navigate } from 'react-router-dom'
import { getOptionalEnv, getPublicBrandConfig } from '../../lib/env'
import { useAuth } from '../../state/auth/useAuth'

function normalizeWhatsApp(value: string) {
  const digits = value.replace(/\D+/g, '')
  if (!digits) return null
  if (digits.startsWith('55')) return digits
  if (digits.length === 10 || digits.length === 11) return `55${digits}`
  return digits
}

export function LandingPage() {
  const { appPrincipal, loading } = useAuth()
  const brand = useMemo(() => getPublicBrandConfig(), [])
  const [copyState, setCopyState] = useState<'idle' | 'ok' | 'error'>('idle')

  const supportWhatsAppRaw = (
    getOptionalEnv('VITE_SUPORTE_WHATSAPP') ??
    getOptionalEnv('VITE_SUPPORT_WHATSAPP_NUMBER') ??
    getOptionalEnv('VITE_SUPPORT_WHATSAPP') ??
    ''
  ).trim()
  const supportWhatsAppDigits = supportWhatsAppRaw ? normalizeWhatsApp(supportWhatsAppRaw) : null
  const waLink = supportWhatsAppDigits
    ? `https://wa.me/${supportWhatsAppDigits}?text=${encodeURIComponent('Olá! Quero conhecer os planos do SMagenda.')}`
    : null

  const preSaleUntilLabel = '08/02/2026'
  const isPreSale = useMemo(() => {
    const now = new Date()
    const end = new Date(2026, 1, 8, 23, 59, 59, 999)
    return now.getTime() <= end.getTime()
  }, [])

  const plans = useMemo(
    () =>
      [
        {
          key: 'basic',
          title: 'BASIC',
          priceLabel: 'R$ 34,99/mês',
          subtitle: 'Ideal para começar com 1 profissional',
          bullets: ['Agendamentos 60 por mês', '1 profissional incluído', 'Lembretes automáticos via WhatsApp', 'Até 3
serviços', 'Página pública personalizável'],
          highlight: false,
        },
        {
          key: 'pro',
          title: 'PRO',
          priceLabel: 'R$ 59,99/mês',
          subtitle: 'Até 6 profissionais (4 inclusos + até 2 adicionais)',
          bullets: ['4 profissionais incluídos', 'Serviços ilimitados', 'Logo e fotos de serviços', 'Relatórios',
'Bloqueios recorrentes'],
          highlight: true,
        },
        {
          key: 'enterprise',
          title: 'EMPRESA',
          priceLabel: 'R$ 98,99/mês',
          subtitle: 'Até 10 profissionais',
          bullets: ['Até 10 profissionais', 'Multi-unidades', 'Agendamentos ilimitados', 'Serviços ilimitados', 'Para
mais profissionais, fale com o suporte'],
          highlight: false,
        },
      ] as const,
    []
  )
```

```
  const exampleBookingUrl = useMemo(() => {
    const origin = typeof window !== 'undefined' ? window.location.origin : ''
    return origin ? `${origin}/agendar/seu-negocio` : '/agendar/seu-negocio'
  }, [])

  const copyExampleUrl = useCallback(async () => {
    try {
      if (!navigator?.clipboard?.writeText) {
        setCopyState('error')
        return
      }
      await navigator.clipboard.writeText(exampleBookingUrl)
      setCopyState('ok')
      window.setTimeout(() => setCopyState('idle'), 1400)
    } catch {
      setCopyState('error')
      window.setTimeout(() => setCopyState('idle'), 1800)
    }
  }, [exampleBookingUrl])

  if (!loading && appPrincipal) {
    if (appPrincipal.kind === 'funcionario') return <Navigate to="/funcionario/agenda" replace />
    if (appPrincipal.kind === 'super_admin') return <Navigate to="/admin/dashboard" replace />
    return <Navigate to="/dashboard" replace />
  }

  return (
    <div className="relative min-h-screen overflow-hidden bg-slate-950 text-white">
      <div className="pointer-events-none absolute inset-0">
        <div className="absolute inset-0 bg-gradient-to-b from-indigo-500/20 via-slate-950 to-slate-950" />
        <div className="absolute -top-40 left-1/2 h-[560px] w-[860px] -translate-x-1/2 rounded-full bg-gradient-to-r
from-indigo-500/25 via-cyan-400/15 to-fuchsia-500/20 blur-3xl" />
        <svg
          className="absolute inset-0 h-full w-full opacity-[0.14]"
          viewBox="0 0 1200 800"
          fill="none"
          xmlns="http://www.w3.org/2000/svg"
          preserveAspectRatio="none"
        >
          <defs>
            <pattern id="grid" width="42" height="42" patternUnits="userSpaceOnUse">
              <path d="M 42 0 L 0 0 0 42" stroke="rgba(148,163,184,0.35)" strokeWidth="1" />
            </pattern>
          </defs>
          <rect width="1200" height="800" fill="url(#grid)" />
        </svg>
      </div>

      <div className="relative">
        <div className="mx-auto w-full max-w-6xl px-4">
          <div className="flex items-center justify-between py-6">
            <div className="flex items-center gap-2">
              {brand.companyLogoUrl ? (
                <div className="flex items-center rounded-xl bg-white/5 px-2 py-1 ring-1 ring-white/10">
                  <img src={brand.companyLogoUrl} alt={brand.companyName} className="h-6 w-auto" />
                </div>
              ) : (
                <>
                  <div className="flex h-9 w-9 items-center justify-center rounded-xl bg-white/10 ring-1 ring-white/10">
                    <svg viewBox="0 0 24 24" fill="none" className="h-5 w-5">
                      <path
                        d="M7 7h10M7 12h10M7 17h7"
                        stroke="currentColor"
                        strokeWidth="2"
                        strokeLinecap="round"
                        strokeLinejoin="round"
                      />
                    </svg>
                  </div>
                </>
              )}
              <div>
```

```
                <div className="text-sm font-semibold leading-tight">{brand.productName}</div>
                <div className="text-xs text-white/60 leading-tight">por {brand.companyName}</div>
              </div>
            </div>

            <div className="flex items-center gap-3">
              <Link to="/login" className="hidden text-sm font-semibold text-white/80 hover:text-white sm:inline">
                Entrar
              </Link>
              <Link
                to="/cadastro"
                className="inline-flex h-10 items-center rounded-xl bg-white px-4 text-sm font-semibold text-slate-950"
              >
                Criar conta
              </Link>
            </div>
          </div>

          <div className="grid grid-cols-1 gap-10 pb-14 pt-10 md:grid-cols-2 md:items-center">
            <div className="space-y-6">
              <div className="inline-flex items-center gap-2 rounded-full bg-white/10 px-3 py-1 text-xs font-semibold text-white/80 ring-1 ring-white/10">
                <span className="h-1.5 w-1.5 rounded-full bg-emerald-400" />
                Agendamentos, WhatsApp e equipe em um só lugar
              </div>

              <div className="text-4xl font-semibold tracking-tight sm:text-5xl">
                Agendamentos online com painel completo
              </div>
              <div className="max-w-xl text-base leading-relaxed text-white/80">
                Link público para clientes agendarem em poucos cliques, agenda com profissionais, serviços e mensagens automáticas no WhatsApp.
              </div>

              <div className="flex flex-col gap-3 sm:flex-row sm:items-center">
                <Link
                  to="/cadastro"
                  className="inline-flex h-11 items-center justify-center rounded-xl bg-white px-5 text-sm font-semibold text-slate-950"
                >
                  Criar minha conta
                </Link>
                <a
                  href="#planos"
                  className="inline-flex h-11 items-center justify-center rounded-xl bg-white/10 px-5 text-sm font-semibold text-white ring-1 ring-white/10"
                >
                  Ver planos
                </a>
                {waLink ? (
                  <a
                    href={waLink}
                    target="_blank"
                    rel="noreferrer"
                    className="inline-flex h-11 items-center justify-center rounded-xl bg-emerald-400/15 px-5 text-sm font-semibold text-emerald-100 ring-1 ring-emerald-300/20"
                  >
                    Falar no WhatsApp
                  </a>
                ) : null}
              </div>

              <div className="text-xs text-white/60">
                Ao criar conta você concorda com{' '}
                <Link to="/termos" className="font-semibold text-white/80 hover:text-white">
                  Termos
                </Link>{' '}
                e{' '}
                <Link to="/privacidade" className="font-semibold text-white/80 hover:text-white">
                  Privacidade
                </Link>
                .
```

```
              </div>
            </div>

            <div className="relative">
              <div className="absolute -inset-6 rounded-[32px] bg-gradient-to-r from-indigo-500/25 via-cyan-400/15 to-fuchsia-500/20 blur-2xl" />
              <div className="relative rounded-[28px] border border-white/10 bg-slate-900/40 p-4 shadow-2xl">
                <div className="flex items-center justify-between rounded-2xl border border-white/10 bg-slate-950/40 px-4 py-3">
                  <div className="flex items-center gap-2">
                    <div className="h-2.5 w-2.5 rounded-full bg-rose-400/80" />
                    <div className="h-2.5 w-2.5 rounded-full bg-amber-300/80" />
                    <div className="h-2.5 w-2.5 rounded-full bg-emerald-400/80" />
                  </div>
                  <div className="text-xs font-semibold text-white/70">Painel {brand.productName}</div>
                  <div className="h-5 w-16 rounded-lg bg-white/5" />
                </div>

                <div className="mt-4 grid grid-cols-1 gap-3">
                  <div className="grid grid-cols-3 gap-3">
                    {[
                      { title: 'Agendamentos', value: 'Organizados', color: 'bg-cyan-400/20 text-cyan-200 ring-cyan-400/30' },
                      { title: 'Clientes', value: 'Centralizados', color: 'bg-indigo-500/20 text-indigo-200 ring-indigo-400/30' },
                      { title: 'Equipe', value: 'Com permissões', color: 'bg-fuchsia-500/20 text-fuchsia-200 ring-fuchsia-400/30' },
                    ].map((k) => (
                      <div key={k.title} className="rounded-2xl border border-white/10 bg-white/5 p-3">
                        <div className="text-xs font-semibold text-white/70">{k.title}</div>
                        <div className="mt-2">
                          <div className={['inline-flex items-center rounded-full px-2 py-1 text-xs font-semibold ring-1', k.color].join(' ')}>
                            {k.value}
                          </div>
                        </div>
                      </div>
                    ))}
                  </div>

                  <div className="rounded-2xl border border-white/10 bg-white/5 p-4">
                    <div className="flex items-center justify-between">
                      <div>
                        <div className="text-sm font-semibold">Link público de agendamento</div>
                        <div className="mt-1 text-xs text-white/60">Compartilhe com clientes para reservar horários</div>
                      </div>
                      <button
                        type="button"
                        onClick={copyExampleUrl}
                        className="inline-flex h-9 items-center rounded-xl bg-white px-3 text-xs font-semibold text-slate-950"
                      >
                        {copyState === 'ok' ? 'Copiado!' : copyState === 'error' ? 'Falhou' : 'Copiar'}
                      </button>
                    </div>
                    <div className="mt-3 rounded-xl border border-white/10 bg-slate-950/30 px-3 py-2 text-xs text-white/70">
                      Exemplo: {exampleBookingUrl}
                    </div>
                  </div>

                  <div className="grid grid-cols-1 gap-3 sm:grid-cols-2">
                    {[
                      { title: 'Confirmação', desc: 'Mensagem automática após agendar' },
                      { title: 'Lembrete', desc: 'Aviso antes do horário marcado' },
                    ].map((b) => (
                      <div key={b.title} className="rounded-2xl border border-white/10 bg-white/5 p-4">
                        <div className="flex items-center gap-2">
                          <div className="flex h-8 w-8 items-center justify-center rounded-xl bg-white/10 ring-1 ring-white/10">
                            <svg viewBox="0 0 24 24" fill="none" className="h-4 w-4 text-white">
```

```
                        <path
                          d="M8 12l2.5 2.5L16 9"
                          stroke="currentColor"
                          strokeWidth="2"
                          strokeLinecap="round"
                          strokeLinejoin="round"
                        />
                      </svg>
                    </div>
                    <div className="text-sm font-semibold">{b.title}</div>
                  </div>
                  <div className="mt-2 text-xs text-white/65">{b.desc}</div>
                </div>
              ))}
            </div>
          </div>
        </div>
      </div>
    </div>

    <div className="pb-14">
      <div className="grid grid-cols-1 gap-4 md:grid-cols-3">
        {[
          {
            title: 'Link público com cara do seu negócio',
            desc: 'Cores, logo e fundo (conforme plano). Cliente agenda em poucos cliques.',
            icon: (
              <svg viewBox="0 0 24 24" fill="none" className="h-5 w-5">
                <path
                  d="M10.5 13.5L13.5 10.5"
                  stroke="currentColor"
                  strokeWidth="2"
                  strokeLinecap="round"
                />
                <path
                  d="M8 16a4 4 0 010-5.657l1.343-1.343A4 4 0 0115 9"
                  stroke="currentColor"
                  strokeWidth="2"
                  strokeLinecap="round"
                />
                <path
                  d="M16 8a4 4 0 010 5.657l-1.343 1.343A4 4 0 019 15"
                  stroke="currentColor"
                  strokeWidth="2"
                  strokeLinecap="round"
                />
              </svg>
            ),
          },
          {
            title: 'Agenda, serviços e profissionais',
            desc: 'Organize horários, regras por serviço e visibilidade por permissões.',
            icon: (
              <svg viewBox="0 0 24 24" fill="none" className="h-5 w-5">
                <path d="M8 7V5m8 2V5" stroke="currentColor" strokeWidth="2" strokeLinecap="round" />
                <path
                  d="M6 9h12M7 19h10a2 2 0 002-2V8a2 2 0 00-2-2H7a2 2 0 00-2 2v9a2 2 0 002 2z"
                  stroke="currentColor"
                  strokeWidth="2"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                />
              </svg>
            ),
          },
          {
            title: 'Automação no WhatsApp',
            desc: 'Templates de confirmação e lembrete, com variáveis do agendamento.',
            icon: (
              <svg viewBox="0 0 24 24" fill="none" className="h-5 w-5">
                <path
                  d="M21 12a8 8 0 01-8 8H7l-4 2 1.2-3.6A8 8 0 1121 12z"
```

```
                      stroke="currentColor"
                      strokeWidth="2"
                      strokeLinecap="round"
                      strokeLinejoin="round"
                    />
                    <path d="M8 12h8" stroke="currentColor" strokeWidth="2" strokeLinecap="round" />
                    <path d="M8 9h5" stroke="currentColor" strokeWidth="2" strokeLinecap="round" />
                  </svg>
                ),
              },
            ].map((f) => (
              <div key={f.title} className="rounded-3xl border border-white/10 bg-white/5 p-5">
                <div className="flex items-start gap-3">
                  <div className="flex h-10 w-10 items-center justify-center rounded-2xl bg-white/10 ring-1 ring-
white/10">
                    {f.icon}
                  </div>
                  <div>
                    <div className="text-sm font-semibold">{f.title}</div>
                    <div className="mt-1 text-sm text-white/70">{f.desc}</div>
                  </div>
                </div>
              </div>
            ))}
          </div>
        </div>

        <div className="pb-14">
          <div className="rounded-[28px] border border-white/10 bg-white/5 p-6">
            <div className="flex flex-col gap-2 sm:flex-row sm:items-end sm:justify-between">
              <div>
                <div className="text-xs font-semibold tracking-wide text-white/60">Para quem é</div>
                <div className="mt-2 text-2xl font-semibold">Feito para negócios com agenda e equipe</div>
                <div className="mt-2 text-sm text-white/70">Funciona bem para atendimentos com horários, profissionais
e regras por serviço.</div>
              </div>
              <Link
                to="/cadastro"
                className="inline-flex h-10 items-center justify-center rounded-xl bg-white/10 px-4 text-sm font-
semibold text-white ring-1 ring-white/10"
              >
                Começar agora
              </Link>
            </div>

            <div className="mt-6 grid grid-cols-1 gap-4 sm:grid-cols-2 lg:grid-cols-3">
              {[
                { title: 'Barbearias e salões', desc: 'Serviços com duração fixa, buffers e profissionais.' },
                { title: 'Clínicas e consultórios', desc: 'Histórico de clientes e lembretes automáticos.' },
                { title: 'Estúdios e aulas', desc: 'Horários recorrentes e organização por equipe.' },
                { title: 'Lava-jato', desc: 'Serviços por tempo e automações no WhatsApp.' },
                { title: 'Serviços em domicílio', desc: 'Link público e regras de antecedência.' },
                { title: 'Negócios multi-unidade', desc: 'Estrutura para planos com unidades (quando habilitado).' },
              ].map((s) => (
                <div key={s.title} className="rounded-3xl border border-white/10 bg-slate-950/20 p-5">
                  <div className="text-sm font-semibold">{s.title}</div>
                  <div className="mt-2 text-sm text-white/70">{s.desc}</div>
                </div>
              ))}
            </div>
          </div>
        </div>

        <div className="pb-14">
          <div className="rounded-[28px] border border-white/10 bg-white/5 p-6">
            <div className="grid grid-cols-1 gap-6 md:grid-cols-3 md:items-start">
              <div>
                <div className="text-xs font-semibold tracking-wide text-white/60">Diferenciais</div>
                <div className="mt-2 text-2xl font-semibold">Mais controle, menos trabalho manual</div>
                <div className="mt-2 text-sm text-white/70">
                  Feito para reduzir no-show, organizar a equipe e padronizar a comunicação com o cliente.
                </div>
```

```
                </div>
                <div className="md:col-span-2">
                  <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                    {[
                      {
                        title: 'Regras por serviço',
                        desc: 'Antecedência, janela máxima, buffers e suporte a serviços de dia inteiro.',
                      },
                      {
                        title: 'Permissões por função',
                        desc: 'Gerente, atendente e profissional com acessos alinhados ao seu fluxo.',
                      },
                      {
                        title: 'WhatsApp com variáveis',
                        desc: 'Confirmação e lembrete com dados reais do agendamento, sem copiar e colar.',
                      },
                      {
                        title: 'Identidade no link público',
                        desc: 'Cores e aparência para passar confiança na hora do cliente agendar (conforme plano).',
                      },
                    ].map((d) => (
                      <div key={d.title} className="rounded-3xl border border-white/10 bg-slate-950/20 p-5">
                        <div className="text-sm font-semibold">{d.title}</div>
                        <div className="mt-2 text-sm text-white/70">{d.desc}</div>
                      </div>
                    ))}
                  </div>
                </div>
              </div>
            </div>
          </div>

          <div className="rounded-[28px] border border-white/10 bg-white/5 p-6">
            <div className="grid grid-cols-1 gap-6 md:grid-cols-3">
              {[
                {
                  step: '1',
                  title: 'Cadastre serviços e horários',
                  desc: 'Defina duração, regras e disponibilidade por profissional.',
                },
                {
                  step: '2',
                  title: 'Compartilhe seu link público',
                  desc: 'O cliente escolhe serviço, dia e horário e confirma.',
                },
                {
                  step: '3',
                  title: 'Envie confirmações e lembretes',
                  desc: 'Templates prontos e personalizáveis no WhatsApp.',
                },
              ].map((s) => (
                <div key={s.step} className="rounded-3xl border border-white/10 bg-slate-950/20 p-5">
                  <div className="flex items-center gap-3">
                    <div className="flex h-9 w-9 items-center justify-center rounded-2xl bg-white text-sm font-semibold
text-slate-950">
                      {s.step}
                    </div>
                    <div>
                      <div className="text-sm font-semibold">{s.title}</div>
                      <div className="mt-1 text-sm text-white/70">{s.desc}</div>
                    </div>
                  </div>
                </div>
              ))}
            </div>
          </div>

          <div id="planos" className="mt-10 rounded-[28px] border border-white/10 bg-white/5 p-6">
            <div className="flex flex-col gap-2 sm:flex-row sm:items-end sm:justify-between">
              <div>
                <div className="text-xs font-semibold tracking-wide text-white/60">Planos</div>
                <div className="mt-2 text-2xl font-semibold">Escolha um plano e comece hoje</div>
```

```
                <div className="mt-2 text-sm text-white/70">Agendamento online + automações no WhatsApp + gestão de
equipe.</div>
                {isPreSale ? <div className="mt-3 text-[11px] text-white/60">Valores de pré-venda • válido até
{preSaleUntilLabel}.</div> : null}
              </div>

              <div className="flex flex-col gap-3 sm:flex-row sm:items-center">
                <Link
                  to="/cadastro"
                  className="inline-flex h-10 items-center justify-center rounded-xl bg-white px-4 text-sm font-semibold
text-slate-950"
                >
                  Criar conta e escolher plano
                </Link>
                {waLink ? (
                  <a
                    href={waLink}
                    target="_blank"
                    rel="noreferrer"
                    className="inline-flex h-10 items-center justify-center rounded-xl bg-white/10 px-4 text-sm font-
semibold text-white ring-1 ring-white/10"
                  >
                    Tirar dúvidas no WhatsApp
                  </a>
                ) : null}
              </div>
            </div>

            <div className="mt-6 grid grid-cols-1 gap-4 lg:grid-cols-3">
              {plans.map((p) => (
                <div
                  key={p.key}
                  className={
                    p.highlight
                      ? 'relative rounded-3xl border border-emerald-300/20 bg-emerald-400/10 p-5 ring-1 ring-emerald-
300/20'
                      : 'rounded-3xl border border-white/10 bg-slate-950/20 p-5'
                  }
                >
                  {p.highlight ? (
                    <div className="absolute -top-3 left-5 inline-flex items-center rounded-full bg-emerald-300/20 px-3
py-1 text-[11px] font-semibold text-emerald-100 ring-1 ring-emerald-300/30">
                      Mais escolhido
                    </div>
                  ) : null}
                  <div className="flex items-start justify-between gap-4">
                    <div>
                      <div className="text-sm font-semibold">{p.title}</div>
                      <div className="mt-1 text-xs text-white/60">{p.subtitle}</div>
                    </div>
                    <div className="text-right">
                      <div className="text-lg font-semibold">{p.priceLabel}</div>
                      <div className="mt-1 text-[11px] text-white/60">por negócio</div>
                    </div>
                  </div>

                  <div className="mt-4 space-y-2 text-sm text-white/75">
                    {p.bullets.map((b) => (
                      <div key={b} className="flex gap-2">
                        <span className="mt-1 h-1.5 w-1.5 shrink-0 rounded-full bg-white/50" />
                        <span>{b}</span>
                      </div>
                    ))}
                  </div>

                  <div className="mt-5">
                    <Link
                      to="/cadastro"
                      className={
                        p.highlight
                          ? 'inline-flex h-10 w-full items-center justify-center rounded-xl bg-white px-4 text-sm font-
semibold text-slate-950'
```

```
                      : 'inline-flex h-10 w-full items-center justify-center rounded-xl bg-white/10 px-4 text-sm
font-semibold text-white ring-1 ring-white/10'
                  }
                >
                  Começar com {p.title}
                </Link>
              </div>
            </div>
          ))}
        </div>

        <div className="mt-6 grid grid-cols-1 gap-4 md:grid-cols-3">
          {[
            { title: 'Checkout em produção', desc: 'Pagamento via PIX (30 dias) ou cartão (assinatura) no Stripe.'
},
            { title: 'Cancelamento simples', desc: 'Você controla sua assinatura dentro do painel.' },
            { title: 'Suporte humano', desc: waLink && supportWhatsAppRaw ? `Atendimento via WhatsApp:
${supportWhatsAppRaw}` : 'Atendimento via WhatsApp e email.' },
          ].map((s) => (
            <div key={s.title} className="rounded-3xl border border-white/10 bg-slate-950/20 p-5">
              <div className="text-sm font-semibold">{s.title}</div>
              <div className="mt-2 text-sm text-white/70">{s.desc}</div>
            </div>
          ))}
        </div>
      </div>

      <div className="mt-10 rounded-[28px] border border-white/10 bg-white/5 p-6">
        <div className="flex flex-col gap-2 sm:flex-row sm:items-end sm:justify-between">
          <div>
            <div className="text-xs font-semibold tracking-wide text-white/60">Dúvidas frequentes</div>
            <div className="mt-2 text-2xl font-semibold">Perguntas comuns antes de começar</div>
          </div>
          <Link to="/ajuda" className="text-sm font-semibold text-white/80 hover:text-white">
            Ver ajuda
          </Link>
        </div>

        <div className="mt-6 grid grid-cols-1 gap-4 md:grid-cols-2">
          {[
            {
              q: 'O cliente precisa instalar aplicativo?',
              a: 'Não. Ele agenda pelo link público no navegador (celular ou PC).',
            },
            {
              q: 'Como funciona a cobrança?',
              a: 'No painel, você escolhe PIX (30 dias) ou cartão (assinatura). O checkout abre em produção via
Stripe.',
            },
            {
              q: 'Consigo reduzir faltas (no-show)?',
              a: 'Sim. Use confirmações e lembretes automáticos no WhatsApp para padronizar o atendimento.',
            },
            {
              q: 'Dá para ter equipe com acessos diferentes?',
              a: 'Sim. Permissões por função para organizar rotina e evitar alterações indevidas.',
            },
            {
              q: 'Meu link pode ter minha identidade?',
              a: 'Sim. Aparência da página pública com cores e, conforme plano, logo e fundo.',
            },
            {
              q: 'O que muda entre BASIC, PRO e EMPRESA?',
              a: 'Principalmente limites (agendamentos e profissionais) e recursos como relatórios, multi-unidades e
personalização completa.',
            },
          ].map((f) => (
            <div key={f.q} className="rounded-3xl border border-white/10 bg-slate-950/20 p-5">
              <div className="text-sm font-semibold">{f.q}</div>
              <div className="mt-2 text-sm text-white/70">{f.a}</div>
            </div>
          ))}
```

```
            </div>
          </div>

          <div className="mt-10 rounded-[28px] border border-white/10 bg-white/5 p-6">
            <div className="flex flex-col gap-4 sm:flex-row sm:items-center sm:justify-between">
              <div>
                <div className="text-lg font-semibold">Quer ver o {brand.productName} rodando?</div>
                <div className="mt-1 text-sm text-white/70">Crie sua conta e configure seu link público em minutos.</div>
              </div>
              <div className="flex gap-3">
                <Link to="/cadastro" className="inline-flex h-11 items-center rounded-xl bg-white px-5 text-sm font-semibold text-slate-950">
                  Criar conta
                </Link>
                <Link
                  to="/login"
                  className="inline-flex h-11 items-center rounded-xl bg-white/10 px-5 text-sm font-semibold text-white ring-1 ring-white/10"
                >
                  Entrar
                </Link>
              </div>
            </div>
          </div>

          <div className="py-10">
            <div className="flex flex-wrap items-center justify-center gap-5 text-sm text-white/70">
              <Link to="/ajuda" className="hover:text-white">
                Ajuda
              </Link>
              <Link to="/termos" className="hover:text-white">
                Termos
              </Link>
              <Link to="/privacidade" className="hover:text-white">
                Privacidade
              </Link>
            </div>
            <div className="mt-4 text-center text-xs text-white/40">
              © {new Date().getFullYear()} {brand.companyName} — {brand.productName}
            </div>
          </div>
        </div>
      </div>
    </div>
  )
}
```

smagenda/src/views/public/PrivacidadePage.tsx

```
import { Link } from 'react-router-dom'

const PRIVACY_VERSION = '2026-01-11'

export function PrivacidadePage() {
  return (
    <div className="min-h-screen bg-slate-50 px-4 py-10">
      <div className="mx-auto w-full max-w-3xl">
        <div className="rounded-2xl border border-slate-200 bg-white p-6 shadow-sm">
          <div className="space-y-1">
            <div className="text-2xl font-semibold text-slate-900">Política de Privacidade — SMagenda</div>
            <div className="text-sm text-slate-600">Versão {PRIVACY_VERSION}</div>
          </div>

          <div className="mt-6 space-y-5 text-sm leading-relaxed text-slate-700">
            <div>
              <div className="font-semibold text-slate-900">1. Visão geral</div>
              <div>
                Esta Política explica como o SMagenda trata dados pessoais ao oferecer a Plataforma. Em muitos casos, o
seu negócio é o
                “Controlador” dos dados dos seus clientes, e o SMagenda atua como “Operador”, tratando dados conforme
suas instruções.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">2. Quais dados tratamos</div>
              <div className="space-y-2">
                <div>
                  <span className="font-medium text-slate-900">Dados de conta:</span> nome, e-mail, telefone, nome do
negócio, slug,
                  configurações.
                </div>
                <div>
                  <span className="font-medium text-slate-900">Dados de agendamentos:</span> informações inseridas por
você, como nome e
                  telefone do cliente, datas/horários, serviço, observações e campos extras (ex.: endereço).
                </div>
                <div>
                  <span className="font-medium text-slate-900">Dados técnicos:</span> registros de autenticação e
segurança, e quando
                  disponível, IP e user-agent no registro de aceite.
                </div>
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">3. Finalidades e bases legais</div>
              <div className="space-y-2">
                <div>
                  <span className="font-medium text-slate-900">Prestação do serviço:</span> operar a agenda, criar e
listar agendamentos,
                  autenticação e suporte.
                </div>
                <div>
                  <span className="font-medium text-slate-900">Cumprimento legal e segurança:</span> prevenção a
fraudes, auditoria e
                  manutenção de logs.
                </div>
                <div>
                  <span className="font-medium text-slate-900">Comunicações:</span> envio de confirmações e lembretes
conforme sua
                  configuração e instruções.
                </div>
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">4. Compartilhamento</div>
              <div>
```

```
                Podemos compartilhar dados com provedores necessários para operar o serviço (por exemplo:
infraestrutura, envio de
                e-mails e integrações de mensagens), sempre com finalidade compatível e medidas de segurança. Não
vendemos dados pessoais.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">5. Armazenamento, retenção e segurança</div>
              <div>
                Aplicamos medidas técnicas e organizacionais para proteger dados. Mantemos dados enquanto sua conta
estiver ativa e pelo
                tempo necessário para cumprir obrigações legais, resolver disputas e fazer cumprir acordos.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">6. Direitos do titular (LGPD)</div>
              <div>
                Titulares podem ter direitos como confirmação, acesso, correção e eliminação. Quando o dado pertence aos
seus clientes,
                normalmente o pedido deve ser direcionado ao seu negócio (Controlador). Podemos auxiliar tecnicamente
mediante sua
                solicitação.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">7. Cookies e tecnologias similares</div>
              <div>
                Podemos usar armazenamento local e recursos essenciais para autenticação e experiência do usuário. Caso
usemos cookies
                adicionais, poderemos apresentar aviso/gestão conforme aplicável.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">8. Transferência internacional</div>
              <div>
                Alguns provedores podem processar dados fora do Brasil. Adotamos salvaguardas contratuais e medidas de
segurança quando
                aplicável.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">9. Alterações desta Política</div>
              <div>
                Podemos atualizar esta Política e, quando necessário, solicitar novo aceite.
              </div>
            </div>
          </div>

          <div className="mt-8 flex flex-wrap items-center justify-between gap-3 border-t border-slate-100 pt-4 text-
sm">
            <Link to="/termos" className="font-medium text-slate-900 hover:underline">
              Ler Termos de Uso
            </Link>
            <Link to="/cadastro" className="text-slate-600 hover:underline">
              Voltar ao cadastro
            </Link>
          </div>
        </div>
      </div>
    </div>
  )
}
```

smagenda/src/views/public/PublicBookingPage.tsx

```
import { useCallback, useEffect, useMemo, useRef, useState } from 'react'
import { useParams } from 'react-router-dom'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { formatBRMoney } from '../../lib/dates'
import { supabase, supabaseEnv } from '../../lib/supabase'

type UsuarioPublico = {
  id: string
  nome_negocio: string
  logo_url: string | null
  tipo_negocio?: string | null
  endereco: string | null
  telefone: string | null
  instagram_url?: string | null
  horario_inicio: string | null
  horario_fim: string | null
  dias_trabalho: number[] | null
  intervalo_inicio: string | null
  intervalo_fim: string | null
  ativo: boolean
  tipo_conta: 'master' | 'individual'
  plano: 'free' | 'basic' | 'pro' | 'team' | 'enterprise'
  public_primary_color: string | null
  public_background_color: string | null
  public_use_background_image: boolean | null
  public_background_image_url: string | null
  unidade_id?: string | null
  unidade_nome?: string | null
  unidade_slug?: string | null
}

function coerceHexColor(value: string | null | undefined, fallback: string) {
  const v = (value ?? '').trim()
  if (!v) return fallback
  if (!/^#(?:[0-9a-fA-F]{3}|[0-9a-fA-F]{6})$/.test(v)) return fallback
  return v
}

type Servico = {
  id: string
  nome: string
  descricao?: string | null
  duracao_minutos: number
  buffer_antes_min?: number
  buffer_depois_min?: number
  antecedencia_minutos?: number
  janela_max_dias?: number
  dia_inteiro?: boolean
  preco: number
  taxa_agendamento: number
  cor: string | null
  foto_url: string | null
}

function startOfMonth(d: Date) {
  return new Date(d.getFullYear(), d.getMonth(), 1)
}

function endOfMonth(d: Date) {
  return new Date(d.getFullYear(), d.getMonth() + 1, 0)
}

function addMonths(d: Date, months: number) {
  return new Date(d.getFullYear(), d.getMonth() + months, 1)
}

function addYears(d: Date, years: number) {
```

```
  return new Date(d.getFullYear() + years, d.getMonth(), d.getDate())
}

function monthRangeIsos(monthIso: string, minDate: Date, maxDate: Date) {
  const base = monthIso ? new Date(`${monthIso}T00:00:00`) : null
  if (!base || !Number.isFinite(base.getTime())) return [] as string[]

  const mStart = startOfMonth(base)
  const mEnd = endOfMonth(base)
  const start = mStart < minDate ? minDate : mStart
  const end = mEnd > maxDate ? maxDate : mEnd

  const out: string[] = []
  const cur = new Date(start)
  cur.setHours(0, 0, 0, 0)
  const last = new Date(end)
  last.setHours(0, 0, 0, 0)

  while (cur <= last) {
    out.push(toIsoDateLocal(cur))
    cur.setDate(cur.getDate() + 1)
  }
  return out
}

function monthLabelPTBR(d: Date) {
  const months = ['Janeiro', 'Fevereiro', 'Março', 'Abril', 'Maio', 'Junho', 'Julho', 'Agosto', 'Setembro', 'Outubro',
'Novembro', 'Dezembro']
  return `${months[d.getMonth()] ?? ''} ${d.getFullYear()}`.trim()
}

function DiaInteiroCalendar(args: {
  primaryColor: string
  selectedIso: string
  setSelectedIso: (iso: string) => void
  minIso: string
  maxIso: string
  monthIso: string
  setMonthIso: (iso: string) => void
  isDayDisabled: (d: Date) => boolean
  diaInteiroDisponibilidade: Record<string, string>
  calendarioLoading: boolean
}) {
  const minIso = args.minIso ?? ''
  const maxIso = args.maxIso ?? ''

  const minDate = useMemo(() => {
    if (!minIso) return null
    const d = new Date(`${minIso}T00:00:00`)
    return Number.isFinite(d.getTime()) ? d : null
  }, [minIso])

  const maxDate = useMemo(() => {
    if (!maxIso) return null
    const d = new Date(`${maxIso}T00:00:00`)
    return Number.isFinite(d.getTime()) ? d : null
  }, [maxIso])

  const fallbackMonthIso = useMemo(() => {
    const fromSelected = args.selectedIso ? new Date(`${args.selectedIso}T00:00:00`) : null
    const base = fromSelected && Number.isFinite(fromSelected.getTime()) ? fromSelected : minDate
    if (base) return toIsoDateLocal(startOfMonth(base))
    const today = new Date()
    today.setHours(0, 0, 0, 0)
    return toIsoDateLocal(startOfMonth(today))
  }, [args.selectedIso, minDate])

  const monthIso = args.monthIso || fallbackMonthIso

  const calendarMonth = useMemo(() => {
    const base = monthIso ? new Date(`${monthIso}T00:00:00`) : null
    if (!base || !Number.isFinite(base.getTime())) return startOfMonth(new Date())
```

```
    return startOfMonth(base)
  }, [monthIso])

  const calendarGridDays = useMemo(() => {
    const monthStart = startOfMonth(calendarMonth)
    const monthEnd = endOfMonth(calendarMonth)
    const startWeekday = monthStart.getDay()
    const gridStart = addDays(monthStart, -startWeekday)
    const gridDays: Date[] = []
    for (let i = 0; i < 42; i += 1) gridDays.push(addDays(gridStart, i))
    return { monthStart, monthEnd, gridDays }
  }, [calendarMonth])

  const goPrevMonth = () => {
    const prev = addMonths(calendarMonth, -1)
    if (minDate && Number.isFinite(minDate.getTime()) && prev < startOfMonth(minDate)) return
    args.setMonthIso(toIsoDateLocal(prev))
  }

  const goNextMonth = () => {
    const next = addMonths(calendarMonth, 1)
    if (maxDate && Number.isFinite(maxDate.getTime()) && startOfMonth(next) > startOfMonth(maxDate)) return
    args.setMonthIso(toIsoDateLocal(next))
  }

  return (
    <div className="mt-3 space-y-3">
      <div className="flex items-center justify-between gap-2">
        <button
          type="button"
          onClick={goPrevMonth}
          className="rounded-xl border border-slate-200 bg-white px-3 py-2 text-sm font-semibold text-slate-700
hover:bg-slate-50"
        >
          ◀
        </button>
        <div className="text-sm font-semibold text-slate-900">{monthLabelPTBR(calendarMonth)}</div>
        <button
          type="button"
          onClick={goNextMonth}
          className="rounded-xl border border-slate-200 bg-white px-3 py-2 text-sm font-semibold text-slate-700
hover:bg-slate-50"
        >
          ▶
        </button>
      </div>

      <div className="grid grid-cols-7 gap-1">
        {['Dom', 'Seg', 'Ter', 'Qua', 'Qui', 'Sex', 'Sáb'].map((w) => (
          <div key={w} className="px-1 py-1 text-center text-[11px] font-semibold text-slate-500">
            {w}
          </div>
        ))}

        {calendarGridDays.gridDays.map((dObj) => {
          const inMonth = dObj.getMonth() === calendarGridDays.monthStart.getMonth()
          if (!inMonth) {
            return <div key={toIsoDateLocal(dObj)} className="h-10 sm:h-12 md:h-14" />
          }
          const iso = toIsoDateLocal(dObj)
          const disabledByRule = args.isDayDisabled(dObj)
          const slot = args.diaInteiroDisponibilidade[iso] ?? ''
          const disabled = disabledByRule || !slot
          const selected = args.selectedIso === iso && !disabled
          return (
            <button
              key={iso}
              type="button"
              disabled={disabled}
              onClick={() => {
                args.setSelectedIso(iso)
              }}
```

```
              className={[
                'h-10 sm:h-12 md:h-14 rounded-xl border text-sm font-semibold',
                disabled ? 'opacity-40 cursor-not-allowed bg-white border-slate-200 text-slate-500' : 'bg-white border-
slate-200 text-slate-900 hover:bg-slate-50',
              ].join(' ')}
              style={selected ? { backgroundColor: args.primaryColor, borderColor: args.primaryColor, color: '#fff' } :
undefined}
            >
              {dObj.getDate()}
            </button>
          )
        })}
      </div>

      <div className="text-xs text-slate-600">{args.calendarioLoading ? 'Carregando disponibilidade…' : 'Selecione um
dia disponível.'}</div>
    </div>
  )
}

function DatePopupCalendar(args: {
  open: boolean
  primaryColor: string
  selectedIso: string
  monthIso: string
  setMonthIso: (iso: string) => void
  onClose: () => void
  onSelect: (iso: string) => void
  minIso: string
  maxIso: string
  isDayDisabled: (d: Date) => boolean
  dayHasSlots: Record<string, boolean>
  availabilityLoading: boolean
}) {
  const minIso = args.minIso ?? ''
  const maxIso = args.maxIso ?? ''

  const minDate = useMemo(() => {
    if (!minIso) return null
    const d = new Date(`${minIso}T00:00:00`)
    return Number.isFinite(d.getTime()) ? d : null
  }, [minIso])

  const maxDate = useMemo(() => {
    if (!maxIso) return null
    const d = new Date(`${maxIso}T00:00:00`)
    return Number.isFinite(d.getTime()) ? d : null
  }, [maxIso])

  const fallbackMonthIso = useMemo(() => {
    const fromSelected = args.selectedIso ? new Date(`${args.selectedIso}T00:00:00`) : null
    const base = fromSelected && Number.isFinite(fromSelected.getTime()) ? fromSelected : minDate
    if (base) return toIsoDateLocal(startOfMonth(base))
    const today = new Date()
    today.setHours(0, 0, 0, 0)
    return toIsoDateLocal(startOfMonth(today))
  }, [args.selectedIso, minDate])

  const monthIso = args.monthIso || fallbackMonthIso

  const calendarMonth = useMemo(() => {
    const base = monthIso ? new Date(`${monthIso}T00:00:00`) : null
    if (!base || !Number.isFinite(base.getTime())) return startOfMonth(new Date())
    return startOfMonth(base)
  }, [monthIso])

  const calendarGridDays = useMemo(() => {
    const monthStart = startOfMonth(calendarMonth)
    const monthEnd = endOfMonth(calendarMonth)
    const startWeekday = monthStart.getDay()
    const gridStart = addDays(monthStart, -startWeekday)
    const gridDays: Date[] = []
```

```
    for (let i = 0; i < 42; i += 1) gridDays.push(addDays(gridStart, i))
    return { monthStart, monthEnd, gridDays }
  }, [calendarMonth])

  const canPrev = useMemo(() => {
    if (!minDate) return true
    const prevMonth = addMonths(calendarMonth, -1)
    return prevMonth >= startOfMonth(minDate)
  }, [calendarMonth, minDate])

  const canNext = useMemo(() => {
    if (!maxDate) return true
    const nextMonth = addMonths(calendarMonth, 1)
    return nextMonth <= startOfMonth(maxDate)
  }, [calendarMonth, maxDate])

  useEffect(() => {
    if (!args.open) return
    const onKeyDown = (e: KeyboardEvent) => {
      if (e.key === 'Escape') args.onClose()
    }
    window.addEventListener('keydown', onKeyDown)
    const prevOverflow = document.body.style.overflow
    document.body.style.overflow = 'hidden'
    return () => {
      window.removeEventListener('keydown', onKeyDown)
      document.body.style.overflow = prevOverflow
    }
  }, [args])

  if (!args.open) return null

  return (
    <div className="fixed inset-0 z-50">
      <button type="button" className="absolute inset-0 bg-slate-900/40" onClick={args.onClose} />
      <div className="relative h-full w-full flex items-end sm:items-center justify-center p-4">
        <div className="w-full max-w-sm rounded-2xl border border-slate-200 bg-white shadow-xl">
          <div className="p-4 space-y-3">
            <div className="flex items-center justify-between gap-3">
              <button
                type="button"
                onClick={() => {
                  if (!canPrev) return
                  args.setMonthIso(toIsoDateLocal(addMonths(calendarMonth, -1)))
                }}
                disabled={!canPrev}
                className="rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm font-semibold text-slate-900 disabled:opacity-40"
              >
                ‹
              </button>
              <div className="text-sm font-semibold text-slate-900">{monthLabelPTBR(calendarMonth)}</div>
              <button
                type="button"
                onClick={() => {
                  if (!canNext) return
                  args.setMonthIso(toIsoDateLocal(addMonths(calendarMonth, 1)))
                }}
                disabled={!canNext}
                className="rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm font-semibold text-slate-900 disabled:opacity-40"
              >
                ›
              </button>
            </div>

            <div className="grid grid-cols-7 gap-1">
              {['Dom', 'Seg', 'Ter', 'Qua', 'Qui', 'Sex', 'Sáb'].map((w) => (
                <div key={w} className="text-center text-[11px] font-semibold text-slate-500 py-1">
                  {w}
                </div>
              ))}
```

```
            </div>

            <div className="grid grid-cols-7 gap-1">
              {calendarGridDays.gridDays.map((dObj) => {
                const iso = toIsoDateLocal(dObj)
                const inMonth = dObj >= calendarGridDays.monthStart && dObj <= calendarGridDays.monthEnd
                const disabled = !inMonth || args.isDayDisabled(dObj) || args.dayHasSlots[iso] === false
                const selected = args.selectedIso === iso && !disabled
                return (
                  <button
                    key={iso}
                    type="button"
                    disabled={disabled}
                    onClick={() => {
                      if (disabled) return
                      args.onSelect(iso)
                      args.onClose()
                    }}
                    className={[
                      'h-10 rounded-xl border text-sm font-semibold',
                      disabled
                        ? 'opacity-40 cursor-not-allowed bg-white border-slate-200 text-slate-400'
                        : 'bg-white border-slate-200 text-slate-900 hover:bg-slate-50',
                      !inMonth && !selected ? 'opacity-60' : '',
                    ].join(' ')}
                    style={selected ? { backgroundColor: args.primaryColor, borderColor: args.primaryColor, color:
'#fff' } : undefined}
                  >
                    {dObj.getDate()}
                  </button>
                )
              })}
            </div>

            <div className="flex justify-end">
              <Button variant="secondary" onClick={args.onClose}>
                Fechar
              </Button>
            </div>

            <div className="text-xs text-slate-600">{args.availabilityLoading ? 'Carregando disponibilidade…' :
'Selecione um dia disponível.'}</div>
          </div>
        </div>
      </div>
    </div>
  )
}

async function callPublicPaymentsFn(body: Record<string, unknown>) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } as unknown }
  }

  const supabaseUrl = String(supabaseEnv.values.VITE_SUPABASE_URL ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
    .replace(/\/+$/g, '')
  const supabaseAnonKey = String(supabaseEnv.values.VITE_SUPABASE_ANON_KEY ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')

  const fnUrl = `${supabaseUrl}/functions/v1/payments`

  let res: Response
  try {
    res = await fetch(fnUrl, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        apikey=[REDACTED]
      },
```

```
      body: JSON.stringify(body),
    })
  } catch (e: unknown) {
    const msg = e instanceof Error ? e.message : 'Falha de rede'
    return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } as unknown }
  }

  const text = await res.text().catch(() => '')
  let parsed: unknown = null
  try {
    parsed = text ? (JSON.parse(text) as unknown) : null
  } catch {
    parsed = text
  }

  return { ok: res.ok as boolean, status: res.status, body: parsed as unknown }
}

type Funcionario = {
  id: string
  nome_completo: string
  horario_inicio: string | null
  horario_fim: string | null
  dias_trabalho: number[] | null
  intervalo_inicio: string | null
  intervalo_fim: string | null
}

function normalizePhone(value: string) {
  return value.replace(/[^0-9+]/g, '').trim()
}

function normalizeEmail(value: string) {
  const v = String(value ?? '').trim().toLowerCase()
  if (!v) return ''
  const ok = /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(v)
  return ok ? v : ''
}

function resolvePublicLabels(tipoNegocio: string | null | undefined) {
  const key = (tipoNegocio ?? '').trim().toLowerCase()
  if (key === 'lava_jatos') return { servico: 'Lavagem', servicos: 'Lavagens', profissional: 'Lavador', profissionais: 'Lavadores' }
  if (key === 'barbearia') return { servico: 'Serviço', servicos: 'Serviços', profissional: 'Barbeiro', profissionais: 'Barbeiros' }
  if (key === 'salao') return { servico: 'Serviço', servicos: 'Serviços', profissional: 'Profissional', profissionais: 'Profissionais' }
  if (key === 'estetica') return { servico: 'Procedimento', servicos: 'Procedimentos', profissional: 'Especialista', profissionais: 'Especialistas' }
  if (key === 'odontologia') return { servico: 'Procedimento', servicos: 'Procedimentos', profissional: 'Dentista', profissionais: 'Dentistas' }
  if (key === 'manicure') return { servico: 'Serviço', servicos: 'Serviços', profissional: 'Manicure', profissionais: 'Manicures' }
  if (key === 'pilates') return { servico: 'Aula', servicos: 'Aulas', profissional: 'Instrutor', profissionais: 'Instrutores' }
  if (key === 'faxina') return { servico: 'Diária', servicos: 'Diárias', profissional: 'Profissional', profissionais: 'Profissionais' }
  return { servico: 'Serviço', servicos: 'Serviços', profissional: 'Profissional', profissionais: 'Profissionais' }
}

function addDays(d: Date, days: number) {
  const x = new Date(d)
  x.setDate(x.getDate() + days)
  return x
}

function toWhatsappNumber(value: string) {
  const digits = String(value ?? '').replace(/[^0-9]/g, '')
  if (!digits) return null
  if (digits.startsWith('55')) return digits
  if (digits.length === 10 || digits.length === 11) return `55${digits}`
  return digits
```

```
}

function toGoogleMapsLink(address: string) {
  return `https://www.google.com/maps/search/?api=1&query=${encodeURIComponent(address)}`
}

function parseHHMMToMinutes(value: string | null | undefined) {
  const raw = String(value ?? '').trim()
  if (!raw) return null
  const parts = raw.split(':')
  const hh = Number(parts[0])
  const mm = Number(parts[1] ?? 0)
  if (!Number.isFinite(hh) || !Number.isFinite(mm)) return null
  if (hh < 0 || hh > 23 || mm < 0 || mm > 59) return null
  return hh * 60 + mm
}

function normalizeInstagramUrl(value: string) {
  const raw = String(value ?? '').trim()
  if (!raw) return null
  if (raw.startsWith('http://') || raw.startsWith('https://')) return raw
  if (raw.startsWith('@')) return `https://instagram.com/${raw.slice(1)}`
  if (raw.includes('instagram.com/')) return `https://${raw.replace(/^https?:\/\//, '')}`
  return `https://instagram.com/${raw}`
}

function rpcErrText(err: unknown) {
  const e = err && typeof err === 'object' ? (err as Record<string, unknown>) : null
  const msg = typeof e?.message === 'string' ? e.message.trim() : 'Erro'
  const code = typeof e?.code === 'string' ? e.code.trim() : ''
  const details = typeof e?.details === 'string' ? e.details.trim() : ''
  const hint = typeof e?.hint === 'string' ? e.hint.trim() : ''
  const parts = [msg]
  if (code && !msg.includes(code)) parts.push(code)
  if (details && details !== msg) parts.push(details)
  if (hint) parts.push(hint)
  return parts.filter(Boolean).join(' — ')
}

function shouldRetryWithoutUnidade(err: unknown) {
  const lower = rpcErrText(err).toLowerCase()
  const mentionsUnidadeArg = lower.includes('p_unidade_id')
  const looksLikeSignatureMismatch = isSignatureMismatchText(lower)
  return mentionsUnidadeArg && looksLikeSignatureMismatch
}

function shouldRetryWithoutFuncionario(err: unknown) {
  const lower = rpcErrText(err).toLowerCase()
  const mentionsFuncionarioArg = lower.includes('p_funcionario_id')
  const looksLikeSignatureMismatch = isSignatureMismatchText(lower)
  return mentionsFuncionarioArg && looksLikeSignatureMismatch
}

function shouldRetryWithoutExtras(err: unknown) {
  const lower = rpcErrText(err).toLowerCase()
  const mentionsExtrasArg = lower.includes('p_extras')
  const looksLikeSignatureMismatch = isSignatureMismatchText(lower)
  return mentionsExtrasArg && looksLikeSignatureMismatch
}

function isMissingRpcFnText(lower: string, fnName: string) {
  const needle = fnName.toLowerCase()
  if (!lower.includes(needle)) return false
  return lower.includes('could not find the function') || lower.includes('does not exist') || (lower.includes('schema
cache') && lower.includes('could not find'))
}

function isSignatureMismatchText(lower: string) {
  return (
    lower.includes('pgrst202') ||
    lower.includes('pgrst203') ||
    lower.includes('could not find the function') ||
```

```
    lower.includes('does not exist') ||
    lower.includes('could not choose the best candidate function between') ||
    (lower.includes('schema cache') && lower.includes('could not find'))
  )
}

async function callPublicRpc<T>(fnName: string, args: Record<string, unknown>) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, errorText: `Supabase não configurado. Faltando: ${supabaseEnv.missing.join(', ')}` }
  }

  const base = String(supabaseEnv.values.VITE_SUPABASE_URL ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
    .replace(/\/+$/g, '')
  const key = String(supabaseEnv.values.VITE_SUPABASE_ANON_KEY ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
  const url = `${base}/rest/v1/rpc/${fnName}`

  let res: Response
  try {
    res = await fetch(url, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        apikey=[REDACTED]
        Authorization: `Bearer ${key}`,
      },
      body: JSON.stringify(args),
    })
  } catch (e: unknown) {
    return { ok: false as const, errorText: e instanceof Error ? e.message : 'Falha de rede' }
  }

  const raw = await res.text().catch(() => '')
  const parsed: unknown = raw
    ? (() => {
        try {
          return JSON.parse(raw) as unknown
        } catch {
          return raw
        }
      })()
    : null

  if (!res.ok) {
    const obj = parsed && typeof parsed === 'object' ? (parsed as Record<string, unknown>) : null
    const msg =
      (typeof obj?.message === 'string' && obj.message.trim()) ||
      (typeof obj?.error === 'string' && obj.error.trim()) ||
      (typeof obj?.hint === 'string' && obj.hint.trim()) ||
      `HTTP ${res.status}`
    const details = typeof obj?.details === 'string' ? obj.details.trim() : ''
    const code = typeof obj?.code === 'string' ? obj.code.trim() : ''
    const parts = [msg]
    if (code && !msg.includes(code)) parts.push(code)
    if (details && details !== msg) parts.push(details)
    return { ok: false as const, errorText: parts.filter(Boolean).join(' — ') }
  }

  return { ok: true as const, data: parsed as T }
}

async function callPublicRpcWithSignatureFallback<T>(
  fnName: string,
  args: Record<string, unknown>,
  unidadeId: string | null
): Promise<{ ok: true; data: T } | { ok: false; errorText: string }> {
  const first = await callPublicRpc<T>(fnName, args)
  if (first.ok) return first
```

```
  const firstLower = first.errorText.toLowerCase()
  if (!isSignatureMismatchText(firstLower)) return first

  const hasUnidadeKey = Object.prototype.hasOwnProperty.call(args, 'p_unidade_id')
  const hasFuncionarioKey = Object.prototype.hasOwnProperty.call(args, 'p_funcionario_id')
  const hasExtrasKey = Object.prototype.hasOwnProperty.call(args, 'p_extras')

  const stableKey = (o: Record<string, unknown>) => {
    const keys = Object.keys(o).sort()
    const ordered: Record<string, unknown> = {}
    for (const k of keys) ordered[k] = o[k]
    return JSON.stringify(ordered)
  }

  const withoutKey = (o: Record<string, unknown>, key: string) => {
    const next: Record<string, unknown> = { ...o }
    delete (next as Record<string, unknown>)[key]
    return next
  }

  const seen = new Set<string>([stableKey(args)])
  const variants: Record<string, unknown>[] = []

  if (hasUnidadeKey) {
    if (shouldRetryWithoutUnidade({ message: first.errorText })) variants.push(withoutKey(args, 'p_unidade_id'))
  } else {
    variants.push({ ...args, p_unidade_id: unidadeId ?? null })
  }

  if (hasFuncionarioKey) {
    if (shouldRetryWithoutFuncionario({ message: first.errorText })) variants.push(withoutKey(args, 'p_funcionario_id'))
  } else {
    variants.push({ ...args, p_funcionario_id: null })
  }

  if (hasExtrasKey) {
    if (shouldRetryWithoutExtras({ message: first.errorText })) variants.push(withoutKey(args, 'p_extras'))
  }

  if (hasUnidadeKey && hasFuncionarioKey) {
    if (shouldRetryWithoutUnidade({ message: first.errorText }) || shouldRetryWithoutFuncionario({ message:
first.errorText })) {
      variants.push(withoutKey(withoutKey(args, 'p_unidade_id'), 'p_funcionario_id'))
    }
  }

  for (const v of variants) {
    const key = stableKey(v)
    if (seen.has(key)) continue
    seen.add(key)
    const res = await callPublicRpc<T>(fnName, v)
    if (res.ok) return res
  }

  return first
}

function toIsoDateLocal(d: Date) {
  const x = new Date(d)
  x.setHours(0, 0, 0, 0)
  const yyyy = x.getFullYear()
  const mm = String(x.getMonth() + 1).padStart(2, '0')
  const dd = String(x.getDate()).padStart(2, '0')
  return `${yyyy}-${mm}-${dd}`
}

function formatIsoDateBR(iso: string) {
  const [yyyy, mm, dd] = iso.split('-')
  if (!yyyy || !mm || !dd) return iso
  return `${dd}/${mm}/${yyyy}`
}
```

```
export function PublicBookingPage() {
  const { slug, unidadeSlug } = useParams()

  const enableTaxaAgendamento = (import.meta.env.VITE_ENABLE_TAXA_AGENDAMENTO as string | undefined) === '1'

  const debugEnabled = useMemo(() => {
    try {
      return new URLSearchParams(window.location.search).get('debug') === '1'
    } catch {
      return false
    }
  }, [])

  const [usuario, setUsuario] = useState<UsuarioPublico | null>(null)
  const [servicos, setServicos] = useState<Servico[]>([])
  const [funcionarios, setFuncionarios] = useState<Funcionario[]>([])
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)

  const [servicoId, setServicoId] = useState('')
  const [funcionarioId, setFuncionarioId] = useState('')
  const [data, setData] = useState('')
  const [hora, setHora] = useState('')
  const [clienteNome, setClienteNome] = useState('')
  const [clienteTelefone, setClienteTelefone] = useState('')
  const [clienteEmail, setClienteEmail] = useState('')
  const [clientePlaca, setClientePlaca] = useState('')
  const [clienteEndereco, setClienteEndereco] = useState('')
  const [submitting, setSubmitting] = useState(false)
  const [successId, setSuccessId] = useState<string | null>(null)
  const [logoFailed, setLogoFailed] = useState(false)

  const [syncingPayment, setSyncingPayment] = useState(() => {
    try {
      const qp = new URLSearchParams(window.location.search)
      const paid = qp.get('paid') === '1'
      const sessionId = (qp.get('session_id') ?? '').trim()
      return Boolean(enableTaxaAgendamento && paid && sessionId)
    } catch {
      return false
    }
  })

  const [modalOpen, setModalOpen] = useState(false)

  const [datePopupOpen, setDatePopupOpen] = useState(false)
  const [datePopupMonthIso, setDatePopupMonthIso] = useState('')
  const [diaInteiroMonthIso, setDiaInteiroMonthIso] = useState('')

  const [step, setStep] = useState<'servico' | 'profissional' | 'quando'>('servico')

  const servicoRef = useRef<HTMLDivElement | null>(null)
  const profissionalRef = useRef<HTMLDivElement | null>(null)
  const whenRef = useRef<HTMLDivElement | null>(null)

  const canChooseProfessional = useMemo(() => {
    const p = String(usuario?.plano ?? '').trim().toLowerCase()
    return p === 'pro' || p === 'team' || p === 'enterprise'
  }, [usuario?.plano])

  const labels = useMemo(() => resolvePublicLabels(usuario?.tipo_negocio ?? null), [usuario?.tipo_negocio])

  const needsPlaca = useMemo(() => String(usuario?.tipo_negocio ?? '').trim().toLowerCase() === 'lava_jatos',
[usuario?.tipo_negocio])

  const needsEndereco = useMemo(() => String(usuario?.tipo_negocio ?? '').trim().toLowerCase() === 'faxina',
[usuario?.tipo_negocio])

  const hasStaff = useMemo(() => canChooseProfessional && funcionarios.length > 0, [canChooseProfessional,
funcionarios.length])

  const primaryColor = useMemo(() => coerceHexColor(usuario?.public_primary_color ?? null, '#0f172a'),
```

```
[usuario?.public_primary_color])
  const backgroundColor = useMemo(
    () => coerceHexColor(usuario?.public_background_color ?? null, '#f8fafc'),
    [usuario?.public_background_color]
  )
  const backgroundImageUrl = (usuario?.public_background_image_url ?? '').trim()
  const shouldUseBgImage = Boolean(usuario?.public_use_background_image) && Boolean(backgroundImageUrl)
  const hasBgImage = Boolean(shouldUseBgImage)

  useEffect(() => {
    const run = async () => {
      if (!slug) {
        setError('Link inválido')
        setLoading(false)
        return
      }
      setLoading(true)
      setError(null)
      setSuccessId(null)

      const wantsUnidade = Boolean((unidadeSlug ?? '').trim())
      const userArgs: Record<string, unknown> = { p_slug: slug }
      if (wantsUnidade) userArgs.p_unidade_slug = unidadeSlug
      const { data: userData, error: userErr } = await supabase.rpc('public_get_usuario_publico',
userArgs).maybeSingle()
      if (userErr) {
        const msg = userErr.message
        const lower = msg.toLowerCase()
        const missingFn = isMissingRpcFnText(lower, 'public_get_usuario_publico')
        setError(
          missingFn
            ? wantsUnidade
              ? 'Configuração do Supabase incompleta: crie o SQL do link público e o SQL de Multi-unidades (EMPRESA).'
              : 'Configuração do Supabase incompleta: crie o SQL do link público (listar + agendar).'
            : msg
        )
        setLoading(false)
        return
      }
      if (!userData) {
        setError('Página não encontrada')
        setLoading(false)
        return
      }
      const usuarioPublico = userData as unknown as UsuarioPublico
      if ((unidadeSlug ?? '').trim() && !usuarioPublico.unidade_id) {
        setError('Unidade não encontrada')
        setLoading(false)
        return
      }
      setUsuario(usuarioPublico)

      const unidadeId = usuarioPublico.unidade_id ?? null

      const { data: servicesData, error: servicesErr } = await supabase.rpc('public_get_servicos_publicos', {
p_usuario_id: usuarioPublico.id })
      if (servicesErr) {
        const msg = servicesErr.message
        const lower = msg.toLowerCase()
        const missingFn = isMissingRpcFnText(lower, 'public_get_servicos_publicos')
        setError(missingFn ? 'Configuração do Supabase incompleta: crie o SQL do link público (listar + agendar).' :
msg)
        setLoading(false)
        return
      }
      const serviceList = (servicesData ?? []) as unknown as Servico[]
      setServicos(serviceList)

      const staffArgs: Record<string, unknown> = { p_usuario_master_id: usuarioPublico.id }
      if (unidadeId) staffArgs.p_unidade_id = unidadeId
      const { data: staffData, error: staffErr } = await supabase.rpc('public_get_funcionarios_publicos', staffArgs)
      if (staffErr) {
```

```
        const msg = staffErr.message
        const lower = msg.toLowerCase()
        const missingFn = isMissingRpcFnText(lower, 'public_get_funcionarios_publicos')
        setError(missingFn ? 'Configuração do Supabase incompleta: crie o SQL do link público (listar + agendar).' :
msg)
        setLoading(false)
        return
      }
      const staffList = (staffData ?? []) as unknown as Funcionario[]
      setFuncionarios(staffList)
      setLoading(false)
    }

    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar')
      setLoading(false)
    })
  }, [slug, unidadeSlug])

  const selectedServico = useMemo(() => servicos.find((s) => s.id === servicoId) ?? null, [servicos, servicoId])
  const selectedFuncionario = useMemo(() => funcionarios.find((f) => f.id === funcionarioId) ?? null, [funcionarios,
funcionarioId])

  const isDiaInteiro = selectedServico?.dia_inteiro === true

  const usuarioId = usuario?.id ?? null

  const minLeadMinutes = useMemo(() => {
    const v = selectedServico?.antecedencia_minutos
    return typeof v === 'number' && Number.isFinite(v) && v >= 0 ? Math.floor(v) : 120
  }, [selectedServico?.antecedencia_minutos])

  const [availableSlots, setAvailableSlots] = useState<string[]>([])
  const [slotsLoading, setSlotsLoading] = useState(false)

  const [debugSlotsText, setDebugSlotsText] = useState<string | null>(null)

  const today = useMemo(() => {
    const d = new Date()
    d.setHours(0, 0, 0, 0)
    return d
  }, [])

  const maxDate = useMemo(() => {
    const d = addYears(today, 1)
    d.setHours(0, 0, 0, 0)
    return d
  }, [today])

  const maxDays = useMemo(() => {
    const diff = maxDate.getTime() - today.getTime()
    if (!Number.isFinite(diff) || diff <= 0) return 0
    return Math.max(0, Math.round(diff / (24 * 60 * 60 * 1000)))
  }, [maxDate, today])

  const allowedWeekdays = useMemo(() => {
    const base = usuario?.dias_trabalho ?? null
    if (!base || base.length === 0) return null
    return new Set(base)
  }, [usuario?.dias_trabalho])

  const isDayDisabledByRule = useCallback(
    (d: Date) => {
      const x = new Date(d)
      x.setHours(0, 0, 0, 0)
      if (x < today) return true
      if (x > maxDate) return true
      if (allowedWeekdays && !allowedWeekdays.has(x.getDay())) return true
      return false
    },
    [allowedWeekdays, maxDate, today]
  )
```

```
  const [diaInteiroDisponibilidade, setDiaInteiroDisponibilidade] = useState<Record<string, string>>({})
  const [calendarDayHasSlots, setCalendarDayHasSlots] = useState<Record<string, boolean>>({})
  const [calendarioLoading, setCalendarioLoading] = useState(false)

  const minIso = useMemo(() => toIsoDateLocal(today), [today])
  const maxIso = useMemo(() => toIsoDateLocal(maxDate), [maxDate])

  const activeCalendarMonthIso = useMemo(() => {
    if (!usuarioId || !selectedServico) return ''
    if (hasStaff && !funcionarioId) return ''
    if (step !== 'quando') return ''
    if (isDiaInteiro) {
      if (diaInteiroMonthIso) return diaInteiroMonthIso
      const base = data ? new Date(`${data}T00:00:00`) : null
      const safe = base && Number.isFinite(base.getTime()) ? base : today
      return toIsoDateLocal(startOfMonth(safe))
    }
    if (datePopupOpen) return datePopupMonthIso || toIsoDateLocal(startOfMonth(today))
    return ''
  }, [data, datePopupMonthIso, datePopupOpen, diaInteiroMonthIso, funcionarioId, hasStaff, isDiaInteiro,
selectedServico, step, today, usuarioId])

  const calendarFetchIsos = useMemo(() => {
    if (!activeCalendarMonthIso) return [] as string[]
    return monthRangeIsos(activeCalendarMonthIso, today, maxDate)
  }, [activeCalendarMonthIso, maxDate, today])

  useEffect(() => {
    let alive = true
    const run = async () => {
      if (!usuarioId || !selectedServico) {
        setCalendarioLoading(false)
        return
      }
      if (hasStaff && !funcionarioId) {
        setCalendarioLoading(false)
        return
      }
      if (calendarFetchIsos.length === 0) {
        setCalendarioLoading(false)
        return
      }
      setDiaInteiroDisponibilidade({})
      setCalendarDayHasSlots({})
      setCalendarioLoading(true)
      const entries = await Promise.all(
        calendarFetchIsos.map(async (iso) => {
          const dObj = new Date(`${iso}T00:00:00`)
          if (!Number.isFinite(dObj.getTime()) || isDayDisabledByRule(dObj)) return null

          const slotsArgs: Record<string, unknown> = {
            p_usuario_id: usuarioId,
            p_data: iso,
            p_servico_id: selectedServico.id,
            p_funcionario_id: hasStaff ? funcionarioId : null,
          }
          if (usuario?.unidade_id) slotsArgs.p_unidade_id = usuario.unidade_id

          const slotsRes = await callPublicRpcWithSignatureFallback<unknown>('public_get_slots_publicos', slotsArgs,
usuario?.unidade_id ?? null)
          if (!slotsRes.ok) return null

          const raw = (slotsRes.data ?? []) as unknown
          const rows = Array.isArray(raw) ? raw : []
          const list = rows
            .map((r) => {
              if (typeof r === 'string') return r.trim()
              if (!r || typeof r !== 'object') return ''
              const obj = r as Record<string, unknown>
              if (typeof obj.hora_inicio === 'string') return obj.hora_inicio.trim()
              if (typeof obj.hora === 'string') return obj.hora.trim()
```

```
            return ''
          })
          .filter(Boolean)
        const slot = list[0] ?? ''
        return { iso, has: Boolean(slot), slot }
      })
    )

    if (!alive) return
    const nextHasSlots: Record<string, boolean> = {}
    for (const iso of calendarFetchIsos) nextHasSlots[iso] = false
    const next: Record<string, string> = {}
    for (const e of entries) {
      if (!e) continue
      nextHasSlots[e.iso] = e.has
      if (e.has && e.slot) next[e.iso] = e.slot
    }
    setCalendarDayHasSlots(nextHasSlots)
    setDiaInteiroDisponibilidade(isDiaInteiro ? next : {})
    setCalendarioLoading(false)
  }
  run().catch(() => {
    if (!alive) return
    setCalendarioLoading(false)
  })
  return () => {
    alive = false
  }
}, [calendarFetchIsos, funcionarioId, hasStaff, isDayDisabledByRule, isDiaInteiro, selectedServico,
usuario?.unidade_id, usuarioId])

useEffect(() => {
  if (!usuario) return
  if (!data) return
  const current = new Date(`${data}T00:00:00`)
  if (!Number.isFinite(current.getTime())) return
  if (!isDayDisabledByRule(current)) return
  setTimeout(() => {
    setHora('')
    setData('')
  }, 0)
}, [data, isDayDisabledByRule, today, usuario])

useEffect(() => {
  let alive = true
  const run = async () => {
    if (!usuarioId || !selectedServico || !data) {
      setAvailableSlots([])
      setSlotsLoading(false)
      if (debugEnabled) setDebugSlotsText(JSON.stringify({ ok: false, reason: 'missing_payload', usuarioId, servicoId:
selectedServico?.id ?? null, data }, null, 2))
      return
    }
    if (hasStaff && !funcionarioId) {
      setAvailableSlots([])
      setSlotsLoading(false)
      if (debugEnabled) setDebugSlotsText(JSON.stringify({ ok: false, reason: 'missing_funcionario', usuarioId, data,
servicoId: selectedServico.id }, null, 2))
      return
    }

    setSlotsLoading(true)
    setError(null)

    const slotsArgs: Record<string, unknown> = {
      p_usuario_id: usuarioId,
      p_data: data,
      p_servico_id: selectedServico.id,
      p_funcionario_id: hasStaff ? funcionarioId : null,
    }
    if (usuario?.unidade_id) slotsArgs.p_unidade_id = usuario.unidade_id
```

```
      const slotsRes = await callPublicRpcWithSignatureFallback<unknown>('public_get_slots_publicos', slotsArgs,
usuario?.unidade_id ?? null)
      if (!alive) return
      if (!slotsRes.ok) {
        const msg = slotsRes.errorText
        const lower = msg.toLowerCase()
        const missingSlotsFn = lower.includes('public_get_slots_publicos') && (lower.includes('function') ||
lower.includes('rpc'))
        const missingOcupacoesFn = lower.includes('public_get_ocupacoes') && isSignatureMismatchText(lower)
        if (missingSlotsFn) {
          setError(
            (unidadeSlug ?? '').trim()
              ? 'Configuração do Supabase incompleta: atualize o SQL do link público e o SQL de Multi-unidades
(EMPRESA).'
              : 'Configuração do Supabase incompleta: atualize o SQL do link público (listar + agendar).'
          )
        } else if (missingOcupacoesFn) {
          setError('Configuração do Supabase incompleta: crie o SQL de horários públicos (ocupações + bloqueios).')
        } else if (lower.includes('data_passada')) {
          setError('Selecione uma data futura.')
        } else if (lower.includes('data_muito_futura')) {
          setError('Selecione uma data dentro de 1 ano.')
        } else if (lower.includes('fora_do_dia_de_trabalho')) {
          setError('Esse dia não está disponível para agendamento.')
        } else if (lower.includes('horarios_nao_configurados')) {
          setError('Horários de trabalho não configurados.')
        } else {
          setError(msg)
        }
        setSlotsLoading(false)
        if (debugEnabled) {
          const current = new Date(`${data}T00:00:00`)
          const schedStartMin = parseHHMMToMinutes(usuario?.horario_inicio ?? null)
          const schedEndMin = parseHHMMToMinutes(usuario?.horario_fim ?? null)
          const ivStartMin = parseHHMMToMinutes(usuario?.intervalo_inicio ?? null)
          const ivEndMin = parseHHMMToMinutes(usuario?.intervalo_fim ?? null)
          const schedSpanMin = schedStartMin !== null && schedEndMin !== null ? schedEndMin - schedStartMin : null
          const duracaoMin = selectedServico?.duracao_minutos ?? null
          const snapshot = {
            ok: false,
            error: msg,
            env: supabaseEnv.ok ? { url: supabaseEnv.values.VITE_SUPABASE_URL, anonKeyPrefix:
`${supabaseEnv.values.VITE_SUPABASE_ANON_KEY.slice(0, 12)}…` } : { missing: supabaseEnv.missing },
            route: { slug: slug ?? null, unidadeSlug: unidadeSlug ?? null },
            payload: slotsArgs,
            selected: {
              servico: selectedServico ? { id: selectedServico.id, duracao_minutos: selectedServico.duracao_minutos,
nome: selectedServico.nome } : null,
              funcionario: hasStaff ? (selectedFuncionario ? { id: selectedFuncionario.id, nome:
selectedFuncionario.nome_completo } : { id: funcionarioId || null }) : null,
            },
            rules: {
              maxDays,
              minLeadMinutes,
              allowedWeekdays: Array.isArray(usuario?.dias_trabalho) ? usuario?.dias_trabalho : null,
              isDayDisabled: Number.isFinite(current.getTime()) ? isDayDisabledByRule(current) : null,
              weekday: Number.isFinite(current.getTime()) ? current.getDay() : null,
            },
            base: {
              usuario_id: usuario?.id ?? null,
              unidade_id: usuario?.unidade_id ?? null,
              horario_inicio: usuario?.horario_inicio ?? null,
              horario_fim: usuario?.horario_fim ?? null,
              intervalo_inicio: usuario?.intervalo_inicio ?? null,
              intervalo_fim: usuario?.intervalo_fim ?? null,
              timezone: (usuario as unknown as { timezone?: string | null } | null)?.timezone ?? null,
            },
            derived: {
              schedStartMin,
              schedEndMin,
              ivStartMin,
              ivEndMin,
```

```
              schedSpanMin,
              duracaoMin,
              duracaoMaiorQueExpediente: schedSpanMin !== null && duracaoMin !== null ? duracaoMin > schedSpanMin :
null,
            },
          }
          setDebugSlotsText(JSON.stringify(snapshot, null, 2))
        }
        return
      }

      const raw = (slotsRes.data ?? []) as unknown
      const rows = Array.isArray(raw) ? raw : []

      const list = rows
        .map((r) => {
          if (typeof r === 'string') return r.trim()
          if (!r || typeof r !== 'object') return ''
          const obj = r as Record<string, unknown>
          if (typeof obj.hora_inicio === 'string') return obj.hora_inicio.trim()
          if (typeof obj.hora === 'string') return obj.hora.trim()
          return ''
        })
        .filter(Boolean)

      const uniqueSorted = Array.from(new Set(list)).sort((a, b) => a.localeCompare(b))
      setAvailableSlots(uniqueSorted)
      setSlotsLoading(false)

      if (debugEnabled) {
        const current = new Date(`${data}T00:00:00`)
        const schedStartMin = parseHHMMToMinutes(usuario?.horario_inicio ?? null)
        const schedEndMin = parseHHMMToMinutes(usuario?.horario_fim ?? null)
        const ivStartMin = parseHHMMToMinutes(usuario?.intervalo_inicio ?? null)
        const ivEndMin = parseHHMMToMinutes(usuario?.intervalo_fim ?? null)
        const schedSpanMin = schedStartMin !== null && schedEndMin !== null ? schedEndMin - schedStartMin : null
        const duracaoMin = selectedServico?.duracao_minutos ?? null
        const snapshot = {
          ok: true,
          env: supabaseEnv.ok ? { url: supabaseEnv.values.VITE_SUPABASE_URL, anonKeyPrefix:
`${supabaseEnv.values.VITE_SUPABASE_ANON_KEY.slice(0, 12)}…` } : { missing: supabaseEnv.missing },
          route: { slug: slug ?? null, unidadeSlug: unidadeSlug ?? null },
          payload: slotsArgs,
          selected: {
            servico: selectedServico ? { id: selectedServico.id, duracao_minutos: selectedServico.duracao_minutos, nome:
selectedServico.nome } : null,
            funcionario: hasStaff ? (selectedFuncionario ? { id: selectedFuncionario.id, nome:
selectedFuncionario.nome_completo } : { id: funcionarioId || null }) : null,
          },
          rules: {
            maxDays,
            minLeadMinutes,
            allowedWeekdays: Array.isArray(usuario?.dias_trabalho) ? usuario?.dias_trabalho : null,
            isDayDisabled: Number.isFinite(current.getTime()) ? isDayDisabledByRule(current) : null,
            weekday: Number.isFinite(current.getTime()) ? current.getDay() : null,
          },
          base: {
            usuario_id: usuario?.id ?? null,
            unidade_id: usuario?.unidade_id ?? null,
            horario_inicio: usuario?.horario_inicio ?? null,
            horario_fim: usuario?.horario_fim ?? null,
            intervalo_inicio: usuario?.intervalo_inicio ?? null,
            intervalo_fim: usuario?.intervalo_fim ?? null,
            timezone: (usuario as unknown as { timezone?: string | null } | null)?.timezone ?? null,
          },
          derived: {
            schedStartMin,
            schedEndMin,
            ivStartMin,
            ivEndMin,
            schedSpanMin,
            duracaoMin,
```

```
            duracaoMaiorQueExpediente: schedSpanMin !== null && duracaoMin !== null ? duracaoMin > schedSpanMin : null,
          },
          result: {
            slots: uniqueSorted.length,
            slots_preview: uniqueSorted.slice(0, 40),
          },
        }
        setDebugSlotsText(JSON.stringify(snapshot, null, 2))
      }
    }
    run().catch((e: unknown) => {
      if (!alive) return
      setError(e instanceof Error ? e.message : 'Erro ao calcular horários')
      setSlotsLoading(false)
      if (debugEnabled) setDebugSlotsText(JSON.stringify({ ok: false, error: e instanceof Error ? e.message : 'Erro ao
calcular horários' }, null, 2))
    })

    return () => {
      alive = false
    }
  }, [data, debugEnabled, funcionarioId, hasStaff, isDayDisabledByRule, maxDays, minLeadMinutes, selectedFuncionario,
selectedServico, slug, unidadeSlug, usuario, usuarioId])

  const effectiveHora = useMemo(() => {
    if (!isDiaInteiro) return hora
    if (!data) return ''
    const slot = diaInteiroDisponibilidade[data] ?? ''
    return slot || ''
  }, [data, diaInteiroDisponibilidade, hora, isDiaInteiro])

  const submit = async () => {
    if (!usuarioId || !selectedServico || !data || !effectiveHora || !clienteNome.trim() || !clienteTelefone.trim())
return
    if (needsPlaca && !clientePlaca.trim()) return
    if (needsEndereco && !clienteEndereco.trim()) return
    setSubmitting(true)
    setError(null)
    setSuccessId(null)

    const telefone = normalizePhone(clienteTelefone)
    if (telefone.length < 8) {
      setError('Telefone inválido.')
      setSubmitting(false)
      return
    }

    const email = normalizeEmail(clienteEmail)
    if (clienteEmail.trim() && !email) {
      setError('Email inválido.')
      setSubmitting(false)
      return
    }

    const placa = clientePlaca.trim().replace(/[^a-zA-Z0-9-]/g, '').toUpperCase()
    const clienteNomeFinal = needsPlaca && placa ? `${clienteNome.trim()} (Placa: ${placa})` : clienteNome.trim()

    const extras: Record<string, unknown> = {}
    if (needsEndereco) extras.endereco = clienteEndereco.trim()
    if (needsPlaca && placa) extras.placa = placa
    if (email) extras.email = email
    const extrasFinal = Object.keys(extras).length > 0 ? extras : null

    const taxa = typeof selectedServico.taxa_agendamento === 'number' &&
Number.isFinite(selectedServico.taxa_agendamento) ? selectedServico.taxa_agendamento : 0
    const normalizedTaxa = Math.max(0, taxa)
    if (enableTaxaAgendamento && normalizedTaxa > 0) {
      const payload: Record<string, unknown> = {
        action: 'create_booking_fee_checkout',
        usuario_id: usuarioId,
        servico_id: selectedServico.id,
        data,
```

```
        hora_inicio: effectiveHora,
        cliente_nome: clienteNomeFinal,
        cliente_telefone: telefone,
        cliente_endereco: needsEndereco ? clienteEndereco.trim() : null,
        slug: slug ?? null,
        unidade_slug: unidadeSlug ?? null,
      }
      if (extrasFinal) payload.extras = extrasFinal
      if (hasStaff) payload.funcionario_id = funcionarioId
      if (usuario?.unidade_id) payload.unidade_id = usuario.unidade_id

      const res = await callPublicPaymentsFn(payload)
      if (!res.ok) {
        const obj = res.body && typeof res.body === 'object' ? (res.body as Record<string, unknown>) : null
        const msg =
          (typeof obj?.message === 'string' && obj.message.trim()) ||
          (typeof obj?.error === 'string' && obj.error.trim()) ||
          (typeof res.body === 'string' && res.body.trim()) ||
          `Erro ao iniciar pagamento (HTTP ${res.status}).`
        setError(msg)
        setSubmitting(false)
        return
      }

      const body = res.body && typeof res.body === 'object' ? (res.body as Record<string, unknown>) : null
      const checkoutUrl = typeof body?.checkout_url === 'string' ? body.checkout_url.trim() : ''
      const kind = typeof body?.kind === 'string' ? body.kind.trim().toLowerCase() : ''
      const agendamentoId = typeof body?.agendamento_id === 'string' ? body.agendamento_id.trim() : ''

      if (kind === 'credit' && agendamentoId) {
        setSuccessId(agendamentoId)
        setSubmitting(false)
        setModalOpen(false)
        return
      }

      if (!checkoutUrl) {
        setError('Falha ao iniciar pagamento: checkout_url ausente.')
        setSubmitting(false)
        return
      }

      window.location.href = checkoutUrl
      return
    }

    const createArgs: Record<string, unknown> = {
      p_usuario_id: usuarioId,
      p_data: data,
      p_hora_inicio: effectiveHora,
      p_servico_id: selectedServico.id,
      p_cliente_nome: clienteNomeFinal,
      p_cliente_telefone: telefone,
      p_funcionario_id: hasStaff ? funcionarioId : null,
    }
    if (extrasFinal) createArgs.p_extras = extrasFinal
    if (usuario?.unidade_id) createArgs.p_unidade_id = usuario.unidade_id

    const createRes = await callPublicRpcWithSignatureFallback<unknown>('public_create_agendamento_publico', createArgs,
usuario?.unidade_id ?? null)
    if (!createRes.ok) {
      const msg = createRes.errorText
      const lower = msg.toLowerCase()
      const missingFn = isMissingRpcFnText(lower, 'public_create_agendamento_publico')
      if (missingFn) {
        setError(
          (unidadeSlug ?? '').trim()
            ? 'Configuração do Supabase incompleta: crie o SQL do link público e o SQL de Multi-unidades (EMPRESA).'
            : 'Configuração do Supabase incompleta: crie o SQL do link público (listar + agendar).'
        )
      } else if (lower.includes('data_passada')) {
        setError('Selecione uma data futura.')
```

```
      } else if (lower.includes('data_muito_futura')) {
        setError('Selecione uma data dentro de 1 ano.')
      } else if (lower.includes('antecedencia_minima')) {
        setError(`Selecione um horário com no mínimo ${Math.round(minLeadMinutes / 60)}h de antecedência.`)
      } else if (lower.includes('ocupado')) {
        setError('Esse horário acabou de ser ocupado. Selecione outro horário.')
      } else if (lower.includes('limite_mensal_atingido')) {
        setError('Este estabelecimento atingiu o limite de agendamentos do mês. Tente novamente no próximo mês.')
      } else if (needsEndereco && lower.includes('p_extras')) {
        setError('Atualização necessária no Supabase: aplique o SQL do link público para salvar o endereço.')
      } else {
        setError(msg)
      }
      setSubmitting(false)
      return
    }
    setSuccessId(createRes.data ? String(createRes.data) : 'ok')
    setSubmitting(false)
    setModalOpen(false)
  }

  useEffect(() => {
    const qp = (() => {
      try {
        return new URLSearchParams(window.location.search)
      } catch {
        return null
      }
    })()

    const paid = qp?.get('paid') === '1'
    const sessionId = (qp?.get('session_id') ?? '').trim()
    if (!enableTaxaAgendamento || !paid || !sessionId) return
    if (!syncingPayment || submitting) return

    callPublicPaymentsFn({ action: 'sync_booking_fee_session', session_id: sessionId })
      .then((res) => {
        if (!res.ok) {
          const obj = res.body && typeof res.body === 'object' ? (res.body as Record<string, unknown>) : null
          const msg =
            (typeof obj?.message === 'string' && obj.message.trim()) ||
            (typeof obj?.error === 'string' && obj.error.trim()) ||
            (typeof res.body === 'string' && res.body.trim()) ||
            `Erro ao confirmar pagamento (HTTP ${res.status}).`
          setError(msg)
          setSyncingPayment(false)
          return
        }
        const body = res.body && typeof res.body === 'object' ? (res.body as Record<string, unknown>) : null
        const agendamentoId = typeof body?.agendamento_id === 'string' ? body.agendamento_id.trim() : ''
        if (agendamentoId) setSuccessId(agendamentoId)
        setSyncingPayment(false)
      })
      .catch((e: unknown) => {
        setError(e instanceof Error ? e.message : 'Erro ao confirmar pagamento')
        setSyncingPayment(false)
      })
  }, [enableTaxaAgendamento, slug, submitting, syncingPayment, unidadeSlug])

  const canSubmit = Boolean(
    usuarioId &&
      selectedServico &&
      data &&
      effectiveHora &&
      clienteNome.trim() &&
      clienteTelefone.trim() &&
      (!hasStaff || funcionarioId) &&
      (!needsPlaca || clientePlaca.trim()) &&
      (!needsEndereco || clienteEndereco.trim())
  )

  const closeModal = () => {
```

```
    setModalOpen(false)
    setError(null)
  }


  const summary = useMemo(() => {
    const servicoNome = selectedServico?.nome ?? null
    const servicoPreco = typeof selectedServico?.preco === 'number' ? selectedServico.preco : null
    const servicoTaxa = typeof selectedServico?.taxa_agendamento === 'number' ? selectedServico.taxa_agendamento : null
    const profissionalNome = hasStaff ? (selectedFuncionario?.nome_completo ?? null) : null
    const canContinue = Boolean(usuarioId && selectedServico && data && effectiveHora && (!hasStaff || funcionarioId))
    const actionLabel = canContinue ? 'Continuar' : 'Escolher horário'
    return { servicoNome, servicoPreco, servicoTaxa, profissionalNome, canContinue, actionLabel }
  }, [data, effectiveHora, funcionarioId, hasStaff, selectedFuncionario?.nome_completo, selectedServico, usuarioId])

  const openFromFooter = () => {
    if (!usuario) return
    if (!summary.canContinue) {
      if (!selectedServico) {
        setStep('servico')
        setTimeout(() => {
          servicoRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
        }, 0)
        return
      }
      if (hasStaff && !funcionarioId) {
        setStep('profissional')
        setTimeout(() => {
          profissionalRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
        }, 0)
        return
      }

      setStep('quando')
      setTimeout(() => {
        whenRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
      }, 0)
      return
    }
    setModalOpen(true)
  }

  const whatsappHref = useMemo(() => {
    const digits = toWhatsappNumber(usuario?.telefone ?? '')
    return digits ? `https://wa.me/${digits}` : null
  }, [usuario?.telefone])

  const instagramHref = useMemo(() => {
    return normalizeInstagramUrl(String(usuario?.instagram_url ?? ''))
  }, [usuario?.instagram_url])

  const visibleSteps = useMemo(() => {
    return hasStaff ? (['servico', 'profissional', 'quando'] as const) : (['servico', 'quando'] as const)
  }, [hasStaff])

  const canGoStep = (target: 'servico' | 'profissional' | 'quando') => {
    if (target === 'servico') return true
    if (target === 'profissional') return Boolean(selectedServico && hasStaff)
    if (target === 'quando') return Boolean(selectedServico && (!hasStaff || funcionarioId))
    return false
  }

  const goStep = (target: 'servico' | 'profissional' | 'quando') => {
    if (!canGoStep(target)) return
    setStep(target)
    setTimeout(() => {
      if (target === 'servico') servicoRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
      if (target === 'profissional') profissionalRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
      if (target === 'quando') whenRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
    }, 0)
  }
```

```
  const slotGroups = useMemo(() => {
    const manha: string[] = []
    const tarde: string[] = []
    const noite: string[] = []
    for (const t of availableSlots) {
      const hh = Number(String(t).split(':')[0])
      if (!Number.isFinite(hh)) continue
      if (hh < 12) manha.push(t)
      else if (hh < 18) tarde.push(t)
      else noite.push(t)
    }
    return { manha, tarde, noite }
  }, [availableSlots])

  return (
    <div
      className="min-h-screen"
      style={{
        backgroundColor: shouldUseBgImage ? 'transparent' : backgroundColor,
        backgroundImage: shouldUseBgImage ? `url(${backgroundImageUrl})` : undefined,
        backgroundSize: shouldUseBgImage ? 'cover' : undefined,
        backgroundPosition: shouldUseBgImage ? 'center' : undefined,
      }}
    >
      <div className="min-h-screen" style={hasBgImage ? { backgroundColor: 'transparent' } : undefined}>
        <div className="mx-auto max-w-xl px-4 py-8 space-y-6">
          <div>
            <div className="text-xs font-semibold tracking-wide text-slate-500">SMagenda</div>
            <div className="text-2xl font-semibold text-slate-900">Agendar</div>
          </div>

          {loading ? <div className="text-sm text-slate-600">Carregando…</div> : null}
          {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div>
: null}
          {successId ? (
            <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">Agendamento
criado.</div>
          ) : null}

          {usuario ? (
            <Card>
              <div className="overflow-hidden rounded-xl">
                <div
                  className="h-24"
                  style={
                    usuario.public_use_background_image && usuario.public_background_image_url
                      ? { backgroundImage: `url(${usuario.public_background_image_url})`, backgroundSize: 'cover',
backgroundPosition: 'center' }
                      : {
                          backgroundImage: `linear-gradient(135deg, ${primaryColor} 0%, #0b1220 100%)`,
                        }
                  }
                />
                <div className="px-5 pb-5">
                  <div className="-mt-10 flex items-end gap-3">
                    {usuario.logo_url && !logoFailed ? (
                      <img
                        src={usuario.logo_url}
                        alt={usuario.nome_negocio}
                        className="h-16 w-16 rounded-full border-2 border-white object-cover shadow-sm"
                        onError={() => setLogoFailed(true)}
                      />
                    ) : (
                      <div className="h-16 w-16 rounded-full border-2 border-white bg-white/70 shadow-sm" />
                    )}
                    <div className="min-w-0">
                      <div className="text-lg font-semibold text-slate-900 truncate">{usuario.nome_negocio}</div>
                      {usuario.unidade_nome ? <div className="text-xs text-slate-600 truncate">{usuario.unidade_nome}
</div> : null}
                    </div>
                  </div>
```

```
            <div className="mt-4 flex flex-wrap gap-2">
              <Badge tone="green">Agendamento online</Badge>
              {usuario.endereco ? <Badge>Endereço</Badge> : null}
              {usuario.telefone && whatsappHref ? (
                <a href={whatsappHref} target="_blank" rel="noreferrer noopener" className="inline-flex">
                  <Badge>WhatsApp</Badge>
                </a>
              ) : usuario.telefone ? (
                <Badge>WhatsApp</Badge>
              ) : null}
              {instagramHref ? (
                <a href={instagramHref} target="_blank" rel="noreferrer noopener" className="inline-flex">
                  <Badge>Instagram</Badge>
                </a>
              ) : null}
            </div>

            <div className="mt-4 space-y-2">
              {usuario.endereco ? (
                <a
                  className="block text-sm text-slate-700 underline underline-offset-2"
                  href={toGoogleMapsLink(usuario.endereco)}
                  target="_blank"
                  rel="noreferrer noopener"
                >
                  {usuario.endereco}
                </a>
              ) : null}
            </div>
          </div>
        </div>
      </Card>
    ) : null}

    {usuario ? (
      <Card>
        <div className="p-4 space-y-4">
          <div className="flex gap-2">
            {visibleSteps.map((s) => {
              const active = step === s
              const enabled = canGoStep(s)
              const label =
                s === 'servico'
                  ? `1. ${labels.servico}`
                  : s === 'profissional'
                    ? `2. ${labels.profissional}`
                    : `${hasStaff ? '3' : '2'}. Quando`
              return (
                <button
                  key={s}
                  type="button"
                  disabled={!enabled}
                  onClick={() => goStep(s)}
                  className={[
                    'flex-1 rounded-lg border px-3 py-2 text-sm font-medium',
                    enabled ? 'cursor-pointer' : 'opacity-40 cursor-not-allowed',
                    active ? 'text-white' : 'bg-white border-slate-200 text-slate-700',
                  ].join(' ')}
                  style={active ? { backgroundColor: primaryColor, borderColor: primaryColor } : undefined}
                >
                  {label}
                </button>
              )
            })}
          </div>
        </div>
      </Card>
    ) : null}

    {usuario ? (
      <div ref={servicoRef} className="space-y-3">
        <div className="text-sm font-semibold text-slate-900">1) {labels.servico}</div>
```

```
              {servicos.length === 0 ? (
                <div className="text-sm text-slate-600">Sem {labels.servicos.toLowerCase()} disponíveis.</div>
              ) : (
                <div className="grid grid-cols-1 gap-3">
                  {servicos.map((s) => {
                    const selected = servicoId === s.id
                    const hasPreco = typeof s.preco === 'number'
                    return (
                      <button
                        key={s.id}
                        type="button"
                        onClick={() => {
                          setError(null)
                          setSuccessId(null)
                          setHora('')
                          setServicoId(s.id)
                          const next = hasStaff ? 'profissional' : 'quando'
                          setStep(next)
                          setTimeout(() => {
                            if (next === 'profissional') profissionalRef.current?.scrollIntoView({ behavior: 'smooth',
block: 'start' })
                            if (next === 'quando') whenRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start'
})
                          }, 0)
                        }}
                        className={[
                          'rounded-2xl border p-4 text-left transition',
                          selected ? 'text-white' : 'bg-white border-slate-200 text-slate-900 hover:bg-slate-50',
                        ].join(' ')}
                        style={selected ? { backgroundColor: primaryColor, borderColor: primaryColor } : undefined}
                      >
                        <div className="flex items-start justify-between gap-3">
                          <div className="flex items-start gap-3 min-w-0">
                            {s.foto_url ? (
                              <img
                                src={s.foto_url}
                                alt={s.nome}
                                className="h-14 w-14 rounded-xl object-cover border border-slate-200"
                                loading="lazy"
                              />
                            ) : (
                              <div className={['h-14 w-14 rounded-xl border border-slate-200', selected ? 'bg-white/20'
: 'bg-slate-50'].join(' ')} />
                            )}
                            <div className="min-w-0">
                              <div className="text-sm font-semibold truncate">{s.nome}</div>
                              {s.descricao ? <div className={selected ? 'mt-1 text-xs text-white/90' : 'mt-1 text-xs
text-slate-600'}>{s.descricao}</div> : null}
                              <div className="mt-2 flex flex-wrap gap-2">
                                <span
                                  className={[
                                    'inline-flex items-center rounded-full px-2 py-0.5 text-xs font-medium',
                                    selected ? 'bg-white/20 text-white' : 'bg-slate-100 text-slate-700',
                                  ].join(' ')}
                                >
                                  {s.duracao_minutos} min
                                </span>
                                {hasPreco ? (
                                  <span
                                    className={[
                                      'inline-flex items-center rounded-full px-2 py-0.5 text-xs font-medium',
                                      selected ? 'bg-white/20 text-white' : 'bg-slate-100 text-slate-700',
                                    ].join(' ')}
                                  >
                                    {formatBRMoney(s.preco)}
                                  </span>
                                ) : null}
                              </div>
                            </div>
                          </div>
                          <div className={['shrink-0 rounded-full px-3 py-1 text-xs font-semibold', selected ? 'bg-
white/20 text-white' : 'bg-slate-900 text-white'].join(' ')}>
```

```
                          {selected ? 'Selecionado' : 'Selecionar'}
                        </div>
                      </div>
                    </button>
                  )
                })}
              </div>
            )}
          </div>
        ) : null}

        {usuario && hasStaff && step !== 'servico' ? (
          <div ref={profissionalRef} className="space-y-3">
            <div className="text-sm font-semibold text-slate-900">2) {labels.profissional}</div>
            {funcionarios.length === 0 ? (
              <div className="text-sm text-slate-600">Sem {labels.profissionais.toLowerCase()} disponíveis.</div>
            ) : (
              <div className="grid grid-cols-1 gap-3">
                {funcionarios.map((f) => {
                  const selected = funcionarioId === f.id
                  return (
                    <button
                      key={f.id}
                      type="button"
                      onClick={() => {
                        setError(null)
                        setSuccessId(null)
                        setHora('')
                        setFuncionarioId(f.id)
                        setStep('quando')
                        setTimeout(() => {
                          whenRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
                        }, 0)
                      }}
                      className={[
                        'rounded-2xl border p-4 text-left transition',
                        selected ? 'text-white' : 'bg-white border-slate-200 text-slate-900 hover:bg-slate-50',
                      ].join(' ')}
                      style={selected ? { backgroundColor: primaryColor, borderColor: primaryColor } : undefined}
                    >
                      <div className="flex items-center justify-between gap-3">
                        <div className="text-sm font-semibold truncate">{f.nome_completo}</div>
                        <div className={['shrink-0 rounded-full px-3 py-1 text-xs font-semibold', selected ? 'bg-
white/20 text-white' : 'bg-slate-900 text-white'].join(' ')}>
                          {selected ? 'Selecionado' : 'Selecionar'}
                        </div>
                      </div>
                    </button>
                  )
                })}
              </div>
            )}
          </div>
        ) : null}

        {usuario && step === 'quando' ? (
          <div ref={whenRef} className="space-y-3">
            <div className="text-sm font-semibold text-slate-900">{hasStaff ? '3' : '2'}) Quando</div>
            <Card>
              <div className="p-4 space-y-4">
                <div>
                  <div className="text-xs font-semibold tracking-wide text-slate-500">Data</div>
                  {isDiaInteiro ? (
                    <DiaInteiroCalendar
                      key={`${servicoId}:${funcionarioId ?? ''}:${usuarioId ?? ''}`}
                      primaryColor={primaryColor}
                      selectedIso={data}
                      setSelectedIso={(iso) => {
                        setError(null)
                        setSuccessId(null)
                        setHora('')
                        setData(iso)
```

```
                          setStep('quando')
                        }}
                        minIso={minIso}
                        maxIso={maxIso}
                        monthIso={diaInteiroMonthIso}
                        setMonthIso={setDiaInteiroMonthIso}
                        isDayDisabled={isDayDisabledByRule}
                        diaInteiroDisponibilidade={diaInteiroDisponibilidade}
                        calendarioLoading={calendarioLoading}
                      />
                    ) : (
                      <button
                        type="button"
                        onClick={() => {
                          const baseIso = data || minIso
                          const base = baseIso ? new Date(`${baseIso}T00:00:00`) : null
                          const safeBase = base && Number.isFinite(base.getTime()) ? base : new Date()
                          setDatePopupMonthIso(toIsoDateLocal(startOfMonth(safeBase)))
                          setDatePopupOpen(true)
                        }}
                        aria-haspopup="dialog"
                        aria-expanded={datePopupOpen}
                        className="mt-2 w-full rounded-xl border border-slate-200 bg-white px-3 py-3 text-left hover:bg-
slate-50"
                      >
                        <div className="text-xs font-semibold text-slate-500">Data selecionada</div>
                        <div className="mt-0.5 text-sm font-semibold text-slate-900">{formatIsoDateBR(data)}</div>
                      </button>
                    )}

                    {!isDiaInteiro ? (
                      <DatePopupCalendar
                        open={datePopupOpen}
                        primaryColor={primaryColor}
                        selectedIso={data}
                        monthIso={datePopupMonthIso}
                        setMonthIso={setDatePopupMonthIso}
                        onClose={() => setDatePopupOpen(false)}
                        onSelect={(iso) => {
                          setError(null)
                          setSuccessId(null)
                          setHora('')
                          setData(iso)
                          setDatePopupMonthIso(toIsoDateLocal(startOfMonth(new Date(`${iso}T00:00:00`))))
                          setDatePopupOpen(false)
                          setStep('quando')
                        }}
                        minIso={minIso}
                        maxIso={maxIso}
                        isDayDisabled={isDayDisabledByRule}
                        dayHasSlots={calendarDayHasSlots}
                        availabilityLoading={calendarioLoading}
                      />
                    ) : null}

                    <div className="mt-2 text-xs text-slate-600">
                      {calendarioLoading ? 'Carregando disponibilidade do calendário… ' : ''}Escolha qualquer data até 1
ano a partir de hoje (mude o mês no calendário). Antecedência mínima: {Math.round(minLeadMinutes / 60)}h.
                    </div>
                  </div>

                  {!isDiaInteiro ? (
                    <div>
                      <div className="text-xs font-semibold tracking-wide text-slate-500">Horário</div>
                      {!selectedServico ? (
                        <div className="mt-2 text-sm text-slate-600">Selecione um {labels.servico.toLowerCase()}.</div>
                      ) : hasStaff && !selectedFuncionario ? (
                        <div className="mt-2 text-sm text-slate-600">Selecione um {labels.profissional.toLowerCase()}.
</div>
                      ) : slotsLoading ? (
                        <div className="mt-2 text-sm text-slate-600">Buscando horários…</div>
                      ) : availableSlots.length === 0 ? (
```

```
                        <div className="mt-2 text-sm text-slate-600">Sem horários disponíveis.</div>
                      ) : (
                        <div className="mt-3 space-y-4">
                          {slotGroups.manha.length > 0 ? (
                            <div className="space-y-2">
                              <div className="text-xs font-semibold text-slate-700">Manhã</div>
                              <div className="grid grid-cols-3 gap-2">
                                {slotGroups.manha.map((t) => {
                                  const selected = hora === t
                                  return (
                                    <button
                                      key={t}
                                      type="button"
                                      onClick={() => setHora(t)}
                                      className={['rounded-xl px-3 py-2 text-sm font-semibold border', selected ? '' : 'bg-white text-slate-700 border-slate-200'].join(' ')}
                                      style={selected ? { backgroundColor: primaryColor, borderColor: primaryColor, color: '#fff' } : { backgroundColor: '#fff' }}
                                    >
                                      {t}
                                    </button>
                                  )
                                })}
                              </div>
                            </div>
                          ) : null}

                          {slotGroups.tarde.length > 0 ? (
                            <div className="space-y-2">
                              <div className="text-xs font-semibold text-slate-700">Tarde</div>
                              <div className="grid grid-cols-3 gap-2">
                                {slotGroups.tarde.map((t) => {
                                  const selected = hora === t
                                  return (
                                    <button
                                      key={t}
                                      type="button"
                                      onClick={() => setHora(t)}
                                      className={['rounded-xl px-3 py-2 text-sm font-semibold border', selected ? '' : 'bg-white text-slate-700 border-slate-200'].join(' ')}
                                      style={selected ? { backgroundColor: primaryColor, borderColor: primaryColor, color: '#fff' } : { backgroundColor: '#fff' }}
                                    >
                                      {t}
                                    </button>
                                  )
                                })}
                              </div>
                            </div>
                          ) : null}

                          {slotGroups.noite.length > 0 ? (
                            <div className="space-y-2">
                              <div className="text-xs font-semibold text-slate-700">Noite</div>
                              <div className="grid grid-cols-3 gap-2">
                                {slotGroups.noite.map((t) => {
                                  const selected = hora === t
                                  return (
                                    <button
                                      key={t}
                                      type="button"
                                      onClick={() => setHora(t)}
                                      className={['rounded-xl px-3 py-2 text-sm font-semibold border', selected ? '' : 'bg-white text-slate-700 border-slate-200'].join(' ')}
                                      style={selected ? { backgroundColor: primaryColor, borderColor: primaryColor, color: '#fff' } : { backgroundColor: '#fff' }}
                                    >
                                      {t}
                                    </button>
                                  )
                                })}
                              </div>
```

```
                            </div>
                          ) : null}
                        </div>
                      )}

                      {debugEnabled && debugSlotsText ? (
                        <div className="mt-4">
                          <Card>
                            <div className="p-4 space-y-2">
                              <div className="text-xs font-semibold tracking-wide text-slate-500">Debug (slots)</div>
                              <pre className="text-[11px] leading-4 text-slate-700 whitespace-pre-wrap break-words font-mono">{debugSlotsText}</pre>
                            </div>
                          </Card>
                        </div>
                      ) : null}
                    </div>
                  ) : (
                    <div>
                      <div className="text-xs font-semibold tracking-wide text-slate-500">Diária</div>
                      {!selectedServico ? (
                        <div className="mt-2 text-sm text-slate-600">Selecione um {labels.servico.toLowerCase()}.</div>
                      ) : hasStaff && !selectedFuncionario ? (
                        <div className="mt-2 text-sm text-slate-600">Selecione um {labels.profissional.toLowerCase()}.</div>
                      ) : slotsLoading ? (
                        <div className="mt-2 text-sm text-slate-600">Buscando disponibilidade…</div>
                      ) : hora ? (
                        <div className="mt-2 text-sm text-slate-700">Dia selecionado com disponibilidade.</div>
                      ) : (
                        <div className="mt-2 text-sm text-slate-600">Selecione um dia disponível no calendário.</div>
                      )}
                    </div>
                  )}
                </div>
              </Card>
            </div>
          ) : null}
        </div>
      </div>

      {usuario && !modalOpen && !successId ? (
        <div className="fixed bottom-0 left-0 right-0 z-40">
          <div className="mx-auto max-w-xl px-4 pb-4">
            <div className="rounded-2xl border border-slate-200 bg-white/95 backdrop-blur p-4 shadow-sm">
              <div className="flex items-center justify-between gap-3">
                <div className="min-w-0">
                  <div className="text-xs font-semibold tracking-wide text-slate-500">Resumo do agendamento</div>
                  <div className="mt-1 text-sm font-semibold text-slate-900 truncate">
                    {formatIsoDateBR(data)} {hora ? `• ${hora}` : ''}
                  </div>
                  <div className="mt-1 text-xs text-slate-700 truncate">
                    {labels.servico}: {summary.servicoNome ?? '—'}
                    {summary.servicoPreco !== null ? ` • ${formatBRMoney(summary.servicoPreco)}` : ''}
                  </div>
                  {hasStaff ? (
                    <div className="mt-1 text-xs text-slate-700 truncate">
                      {labels.profissional}: {summary.profissionalNome ?? '—'}
                    </div>
                  ) : null}
                </div>
                <Button
                  onClick={openFromFooter}
                  style={{ backgroundColor: primaryColor, borderColor: primaryColor }}
                  className="px-6 py-3 text-base shadow-md"
                >
                  {summary.actionLabel}
                </Button>
              </div>
            </div>
          </div>
        </div>
```

```
      ) : null}

      {usuario && !modalOpen && !successId ? <div className="h-28" /> : null}

      {modalOpen && usuario ? (
        <div className="fixed inset-0 z-50 flex items-center justify-center p-4">
          <button type="button" className="absolute inset-0 bg-slate-900/40" onClick={closeModal} aria-label="Fechar" />
          <div className="relative w-full max-w-lg">
            <Card>
              <div className="p-6 space-y-4">
                <div className="flex items-start justify-between gap-3">
                  <div>
                    <div className="text-sm font-semibold text-slate-900">Agendar em {formatIsoDateBR(data)}</div>
                    <div className="text-xs text-slate-600">{usuario.nome_negocio}</div>
                  </div>
                  <Button variant="secondary" onClick={closeModal}>
                    Fechar
                  </Button>
                </div>

                {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}

                <div className="space-y-4">
                  <div className="text-sm font-semibold text-slate-900">Confirmar agendamento</div>

                  <div className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-sm text-slate-700">
                    <div className="flex items-center justify-between gap-2">
                      <div>Dia</div>
                      <div className="font-semibold">{formatIsoDateBR(data)}</div>
                    </div>
                    <div className="flex items-center justify-between gap-2">
                      <div>Hora</div>
                      <div className="font-semibold">{hora || '—'}</div>
                    </div>
                    {needsPlaca && clientePlaca.trim() ? (
                      <div className="flex items-center justify-between gap-2">
                        <div>Placa</div>
                        <div className="font-semibold">{clientePlaca.trim().replace(/[^a-zA-Z0-9-]/g, '').toUpperCase()}</div>
                      </div>
                    ) : null}
                    {needsEndereco && clienteEndereco.trim() ? (
                      <div className="flex items-center justify-between gap-2">
                        <div>Endereço</div>
                        <div className="font-semibold text-right">{clienteEndereco.trim()}</div>
                      </div>
                    ) : null}
                    <div className="flex items-center justify-between gap-2">
                      <div>{labels.servico}</div>
                      <div className="font-semibold">{selectedServico?.nome ?? '—'}</div>
                    </div>
                    {hasStaff ? (
                      <div className="flex items-center justify-between gap-2">
                        <div>{labels.profissional}</div>
                        <div className="font-semibold">{selectedFuncionario?.nome_completo ?? '—'}</div>
                      </div>
                    ) : null}
                  </div>

                  <Input label="Nome" value={clienteNome} onChange={(e) => setClienteNome(e.target.value)} />
                  <Input label="WhatsApp" value={clienteTelefone} onChange={(e) => setClienteTelefone(e.target.value)} />
                  <Input label="Email (opcional)" value={clienteEmail} onChange={(e) => setClienteEmail(e.target.value)} />
                  {needsPlaca ? <Input label="Placa do carro" value={clientePlaca} onChange={(e) => setClientePlaca(e.target.value)} /> : null}
                  {needsEndereco ? <Input label="Endereço" value={clienteEndereco} onChange={(e) => setClienteEndereco(e.target.value)} /> : null}

                  <div className="flex justify-between">
                    <Button variant="secondary" onClick={closeModal}>
```

```
                      Voltar
                    </Button>
                    <Button
                      onClick={submit}
                      disabled={!canSubmit || submitting}
                      style={{ backgroundColor: primaryColor, borderColor: primaryColor }}
                      className="px-7 py-3 text-base shadow-lg"
                    >
                      {submitting ? 'Confirmando…' : 'Confirmar'}
                    </Button>
                  </div>
                </div>
              </div>
            </Card>
          </div>
        </div>
      ) : null}

      {successId ? (
        <div className="fixed inset-0 z-50 flex items-center justify-center p-4">
          <div className="absolute inset-0 bg-slate-900/40" />
          <div className="relative w-full max-w-lg">
            <Card>
              <div className="p-6 space-y-4 text-center">
                <div className="text-lg font-semibold text-slate-900">Agendamento enviado!</div>
                <div className="text-sm text-slate-700">Seu agendamento foi enviado. Você receberá a confirmação em breve.</div>
                {data || hora ? (
                  <div className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-sm text-slate-700">
                    <div className="flex items-center justify-between gap-2">
                      <div>Dia</div>
                      <div className="font-semibold">{data ? formatIsoDateBR(data) : '—'}</div>
                    </div>
                    <div className="flex items-center justify-between gap-2">
                      <div>Hora</div>
                      <div className="font-semibold">{hora || '—'}</div>
                    </div>
                  </div>
                ) : null}
                <Button
                  onClick={() => {
                    setSuccessId(null)
                    setModalOpen(false)
                    setError(null)
                  }}
                  style={{ backgroundColor: primaryColor, borderColor: primaryColor }}
                  className="py-3 text-base shadow-md"
                  fullWidth
                >
                  Fechar
                </Button>
              </div>
            </Card>
          </div>
        </div>
      ) : null}
    </div>
  )
}
```

smagenda/src/views/public/TermosPage.tsx

```
import { Link } from 'react-router-dom'

const TERMS_VERSION = '2026-01-11'

export function TermosPage() {
  return (
    <div className="min-h-screen bg-slate-50 px-4 py-10">
      <div className="mx-auto w-full max-w-3xl">
        <div className="rounded-2xl border border-slate-200 bg-white p-6 shadow-sm">
          <div className="space-y-1">
            <div className="text-2xl font-semibold text-slate-900">Termos de Uso — SMagenda</div>
            <div className="text-sm text-slate-600">Versão {TERMS_VERSION}</div>
          </div>

          <div className="mt-6 space-y-5 text-sm leading-relaxed text-slate-700">
            <div>
              <div className="font-semibold text-slate-900">1. Aceite</div>
              <div>
                Ao criar uma conta, acessar ou usar o SMagenda (“Plataforma”), você declara que leu e concorda com estes
Termos de
                Uso e com a Política de Privacidade. Se você não concordar, não utilize a Plataforma.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">2. Quem pode usar</div>
              <div>
                Você deve ter capacidade legal para contratar e fornecer informações verdadeiras. Se você cria a conta
em nome de uma
                empresa, você declara ter poderes para representá-la.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">3. O que a Plataforma faz</div>
              <div>
                O SMagenda é um sistema de gestão de agenda, serviços e clientes, incluindo uma página pública para
agendamentos
                (“Link Público”). A Plataforma não presta os serviços finais ao cliente do seu negócio — você é o
responsável pelo
                atendimento, execução do serviço, preços e políticas do seu estabelecimento.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">4. Conta, acesso e segurança</div>
              <div>
                Você é responsável por manter a confidencialidade das credenciais de acesso e por toda atividade
ocorrida na sua conta.
                Em caso de suspeita de uso indevido, altere sua senha e entre em contato com o suporte.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">5. Conteúdo e dados inseridos</div>
              <div>
                Você mantém a titularidade dos dados inseridos (por exemplo: cadastros, serviços e agendamentos). Você
garante que tem
                base legal para tratar os dados dos seus clientes e profissionais e que cumprirá a legislação aplicável,
inclusive a
                LGPD.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">6. Mensagens e comunicações</div>
              <div>
                A Plataforma pode integrar-se a canais como WhatsApp e e-mail para lembretes e confirmações. Você é o
responsável pelo
                conteúdo das mensagens e por obter consentimentos quando necessário. A disponibilidade de provedores
```

```
terceiros pode
                variar.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">7. Planos, pagamento e limitações</div>
              <div>
                Recursos podem variar por plano e podem existir limites de uso (por exemplo: número de funcionários).
Condições de
                cobrança, reajustes e períodos de teste, quando aplicáveis, podem ser informados no momento da
contratação.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">8. Proibições</div>
              <div>
                É proibido: usar a Plataforma para fins ilícitos; tentar explorar vulnerabilidades; interferir no
funcionamento do
                serviço; automatizar acessos sem autorização; ou violar direitos de terceiros.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">9. Disponibilidade e suporte</div>
              <div>
                Buscamos alta disponibilidade, mas podem ocorrer indisponibilidades por manutenção, atualizações, falhas
de rede ou
                serviços de terceiros. Poderemos realizar melhorias e mudanças na Plataforma.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">10. Propriedade intelectual</div>
              <div>
                A Plataforma, marcas e softwares associados pertencem ao Single Motion e/ou seus licenciadores. Você não
recebe qualquer
                direito de propriedade sobre a Plataforma.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">11. Rescisão e encerramento</div>
              <div>
                Você pode encerrar sua conta a qualquer momento. Podemos suspender ou encerrar o acesso em caso de
violação destes
                Termos, exigências legais, segurança ou uso abusivo. Podemos manter registros quando exigidos por lei.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">12. Limitação de responsabilidade</div>
              <div>
                Na máxima extensão permitida pela lei, não nos responsabilizamos por perdas indiretas, lucros cessantes
ou danos
                decorrentes do uso do serviço, incluindo falhas de terceiros (por exemplo: internet, provedores de
mensagens, e-mail).
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">13. Alterações</div>
              <div>
                Podemos atualizar estes Termos. Quando houver alteração relevante, poderemos solicitar novo aceite. A
versão vigente
                estará sempre disponível nesta página.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">14. Contato</div>
```

```
              <div>
                Para dúvidas, solicite suporte pelos canais informados no painel da Plataforma.
              </div>
            </div>
          </div>

          <div className="mt-8 flex flex-wrap items-center justify-between gap-3 border-t border-slate-100 pt-4 text-
sm">
            <Link to="/privacidade" className="font-medium text-slate-900 hover:underline">
              Ler Política de Privacidade
            </Link>
            <Link to="/cadastro" className="text-slate-600 hover:underline">
              Voltar ao cadastro
            </Link>
          </div>
        </div>
      </div>
    </div>
  )
}
```

smagenda/supabase/functions/admin-create-funcionario/config.toml

```
verify_jwt = false
```

smagenda/supabase/functions/admin-create-funcionario/index.ts

```
import { createClient } from 'https://esm.sh/@supabase/supabase-js@2.89.0'

type Payload = {
  usuario_master_id: string
  nome_completo: string
  email: string
  senha: string
  telefone?: string | null
  permissao: 'admin' | 'funcionario' | 'atendente'
  ativo?: boolean
}

function jsonResponse(status: number, body: unknown) {
  return new Response(JSON.stringify(body), {
    status,
    headers: {
      'Content-Type': 'application/json',
      'Access-Control-Allow-Origin': '*',
      'Access-Control-Allow-Headers': 'authorization, x-client-info, apikey, content-type',
      'Access-Control-Allow-Methods': 'POST, OPTIONS',
    },
  })
}

function normalizeUuid(value: unknown) {
  const s = typeof value === 'string' ? value.trim() : ''
  if (!s) return null
  if (!/^[0-9a-fA-F-]{36}$/.test(s)) return null
  return s
}

Deno.serve(async (req) => {
  if (req.method === 'OPTIONS') return jsonResponse(200, { ok: true })
  if (req.method !== 'POST') return jsonResponse(405, { error: 'method_not_allowed' })

  const supabaseUrl = Deno.env.get('SUPABASE_URL') ?? ''
  const anonKey = Deno.env.get('SUPABASE_ANON_KEY') ?? ''
  const serviceRoleKey=[REDACTED]'SERVICE_ROLE_KEY') ?? Deno.env.get('SUPABASE_SERVICE_ROLE_KEY') ?? ''

  if (!supabaseUrl || !anonKey || !serviceRoleKey) {
    return jsonResponse(500, { error: 'missing_env' })
  }

  const userClient = createClient(supabaseUrl, anonKey, {
    global: {
      headers: {
        Authorization: req.headers.get('Authorization') ?? '',
      },
    },
  })

  const adminClient = createClient(supabaseUrl, serviceRoleKey)

  const { data: userData, error: userErr } = await userClient.auth.getUser()
  if (userErr || !userData?.user) {
    return jsonResponse(401, { error: 'unauthorized', message: userErr?.message ?? 'invalid_session' })
  }

  const callerId = userData.user.id
  const { data: saRow, error: saErr } = await userClient.from('super_admin').select('id').eq('id',
callerId).maybeSingle()
  if (saErr || !saRow) {
    return jsonResponse(403, { error: 'forbidden', message: 'Sem permissão.' })
  }

  let payload: Payload
  try {
    payload = (await req.json()) as Payload
  } catch {
    return jsonResponse(400, { error: 'invalid_json' })
```

```
  }

  const masterId = normalizeUuid(payload.usuario_master_id)
  if (!masterId) return jsonResponse(400, { error: 'invalid_usuario_master_id' })

  const email = (payload.email ?? '').trim().toLowerCase()
  const senha = payload.senha ?? ''
  const nomeCompleto = (payload.nome_completo ?? '').trim()
  if (!email || !senha || !nomeCompleto) {
    return jsonResponse(400, { error: 'missing_fields' })
  }
  if (senha.trim().length < 8) {
    return jsonResponse(400, { error: 'weak_password' })
  }
  if (payload.permissao !== 'admin' && payload.permissao !== 'funcionario' && payload.permissao !== 'atendente') {
    return jsonResponse(400, { error: 'invalid_permissao' })
  }

  const { data: masterRow, error: masterErr } = await adminClient
    .from('usuarios')
    .select('id,plano,limite_funcionarios,ativo')
    .eq('id', masterId)
    .maybeSingle()

  if (masterErr || !masterRow) {
    return jsonResponse(404, { error: 'master_not_found' })
  }
  if (masterRow.ativo === false) {
    return jsonResponse(403, { error: 'master_inactive' })
  }

  const { count: activeCount, error: countErr } = await adminClient
    .from('funcionarios')
    .select('id', { count: 'exact', head: true })
    .eq('usuario_master_id', masterId)
    .eq('ativo', true)

  if (countErr) {
    return jsonResponse(400, { error: 'cannot_check_limit', message: countErr.message })
  }

  const plano = String((masterRow as { plano?: unknown }).plano ?? '')
    .trim()
    .toLowerCase()

  let limiteEfetivo: number | null | undefined = masterRow.limite_funcionarios
  if (plano === 'enterprise') {
    limiteEfetivo = 10
  } else if (plano === 'pro' || plano === 'team') {
    const raw = masterRow.limite_funcionarios
    const n = typeof raw === 'number' && Number.isFinite(raw) && raw > 0 ? Math.floor(raw) : null
    const base = 4
    const max = 6
    limiteEfetivo = !n || n < base ? base : Math.min(max, n)
  } else if (plano === 'basic' || plano === 'free') {
    limiteEfetivo = 1
  }

  if (typeof limiteEfetivo === 'number' && limiteEfetivo > 0 && (activeCount ?? 0) >= limiteEfetivo) {
    return jsonResponse(400, { error: 'limit_reached' })
  }

  const { data: created, error: createErr } = await adminClient.auth.admin.createUser({
    email,
    password: senha,
    email_confirm: true,
    user_metadata: {
      nome_completo: nomeCompleto,
      usuario_master_id: masterId,
      permissao: payload.permissao,
    },
  })
```

```
  if (createErr || !created.user) {
    return jsonResponse(400, { error: 'create_user_failed', message: createErr?.message ?? 'unknown' })
  }

  const funcionarioId = created.user.id

  const insertPayload = {
    id: funcionarioId,
    usuario_master_id: masterId,
    nome_completo: nomeCompleto,
    email,
    telefone: payload.telefone ? String(payload.telefone).trim() : null,
    permissao: payload.permissao,
    ativo: payload.ativo ?? true,
  }

  const { error: insertErr } = await adminClient.from('funcionarios').insert(insertPayload)
  if (insertErr) {
    await adminClient.auth.admin.deleteUser(funcionarioId)
    return jsonResponse(400, { error: 'insert_funcionario_failed', message: insertErr.message })
  }

  return jsonResponse(200, { id: funcionarioId, email })
})
```

smagenda/supabase/functions/create-funcionario/config.toml

```
verify_jwt = false
```

smagenda/supabase/functions/create-funcionario/index.ts

```
import { createClient } from 'https://esm.sh/@supabase/supabase-js@2.89.0'

type Payload = {
  nome_completo: string
  email: string
  senha: string
  telefone?: string | null
  permissao: 'admin' | 'funcionario' | 'atendente'
  horario_inicio?: string | null
  horario_fim?: string | null
  dias_trabalho?: number[] | null
  intervalo_inicio?: string | null
  intervalo_fim?: string | null
  pode_ver_agenda?: boolean
  pode_criar_agendamentos?: boolean
  pode_cancelar_agendamentos?: boolean
  pode_bloquear_horarios?: boolean
  pode_ver_financeiro?: boolean
  pode_gerenciar_servicos?: boolean
  pode_ver_clientes_de_outros?: boolean
  ativo?: boolean
}

function jsonResponse(status: number, body: unknown) {
  return new Response(JSON.stringify(body), {
    status,
    headers: {
      'Content-Type': 'application/json',
      'Access-Control-Allow-Origin': '*',
      'Access-Control-Allow-Headers': 'authorization, x-client-info, apikey, content-type',
      'Access-Control-Allow-Methods': 'POST, OPTIONS',
    },
  })
}

Deno.serve(async (req) => {
  if (req.method === 'OPTIONS') return jsonResponse(200, { ok: true })
  if (req.method !== 'POST') return jsonResponse(405, { error: 'method_not_allowed' })

  const supabaseUrl = Deno.env.get('SUPABASE_URL') ?? ''
  const anonKey = Deno.env.get('SUPABASE_ANON_KEY') ?? ''
  const serviceRoleKey=[REDACTED]'SERVICE_ROLE_KEY') ?? Deno.env.get('SUPABASE_SERVICE_ROLE_KEY') ?? ''

  if (!supabaseUrl || !anonKey || !serviceRoleKey) {
    return jsonResponse(500, { error: 'missing_env' })
  }

  const userClient = createClient(supabaseUrl, anonKey, {
    global: {
      headers: {
        Authorization: req.headers.get('Authorization') ?? '',
      },
    },
  })

  const adminClient = createClient(supabaseUrl, serviceRoleKey)

  const { data: userData, error: userErr } = await userClient.auth.getUser()
  if (userErr || !userData?.user) {
    return jsonResponse(401, { error: 'unauthorized', message: userErr?.message ?? 'invalid_session' })
  }

  let payload: Payload
  try {
    payload = (await req.json()) as Payload
  } catch {
    return jsonResponse(400, { error: 'invalid_json' })
  }

  const email = (payload.email ?? '').trim().toLowerCase()
```

```
const senha = payload.senha ?? ''
const nomeCompleto = (payload.nome_completo ?? '').trim()
if (!email || !senha || !nomeCompleto) {
  return jsonResponse(400, { error: 'missing_fields' })
}
if (senha.trim().length < 8) {
  return jsonResponse(400, { error: 'weak_password' })
}
if (payload.permissao !== 'admin' && payload.permissao !== 'funcionario' && payload.permissao !== 'atendente') {
  return jsonResponse(400, { error: 'invalid_permissao' })
}

const masterId = userData.user.id
const { data: masterRow, error: masterErr } = await userClient
  .from('usuarios')
  .select('id,plano,limite_funcionarios,ativo,status_pagamento,data_vencimento')
  .eq('id', masterId)
  .maybeSingle()

if (masterErr || !masterRow) {
  return jsonResponse(403, { error: 'not_master' })
}
if (masterRow.ativo === false) {
  return jsonResponse(403, { error: 'master_inactive' })
}

const statusPagamento = String((masterRow as { status_pagamento?: unknown }).status_pagamento ?? '')
  .trim()
  .toLowerCase()
const venc = String((masterRow as { data_vencimento?: unknown }).data_vencimento ?? '').trim()
if (statusPagamento === 'trial' && venc) {
  const today = new Date()
  today.setHours(0, 0, 0, 0)
  const exp = new Date(`${venc}T00:00:00`)
  if (Number.isFinite(exp.getTime()) && exp < today) {
    return jsonResponse(403, { error: 'trial_expired' })
  }
}

if (statusPagamento && statusPagamento !== 'ativo' && statusPagamento !== 'trial') {
  return jsonResponse(403, { error: 'payment_inactive', status_pagamento: statusPagamento })
}

const { count: activeCount, error: countErr } = await userClient
  .from('funcionarios')
  .select('id', { count: 'exact', head: true })
  .eq('usuario_master_id', masterId)
  .eq('ativo', true)

if (countErr) {
  return jsonResponse(400, { error: 'cannot_check_limit' })
}

const plano = String((masterRow as { plano?: unknown }).plano ?? '')
  .trim()
  .toLowerCase()

let limiteEfetivo: number | null | undefined = masterRow.limite_funcionarios
if (plano === 'enterprise') {
  limiteEfetivo = 10
} else if (plano === 'pro' || plano === 'team') {
  const raw = masterRow.limite_funcionarios
  const n = typeof raw === 'number' && Number.isFinite(raw) && raw > 0 ? Math.floor(raw) : null
  const base = 4
  const max = 6
  limiteEfetivo = !n || n < base ? base : Math.min(max, n)
} else if (plano === 'basic' || plano === 'free') {
  limiteEfetivo = 1
}

if (typeof limiteEfetivo === 'number' && limiteEfetivo > 0 && (activeCount ?? 0) >= limiteEfetivo) {
  return jsonResponse(400, { error: 'limit_reached' })
```

```
  }

  const { data: created, error: createErr } = await adminClient.auth.admin.createUser({
    email,
    password: senha,
    email_confirm: true,
    user_metadata: {
      nome_completo: nomeCompleto,
      usuario_master_id: masterId,
      permissao: payload.permissao,
    },
  })

  if (createErr || !created.user) {
    return jsonResponse(400, { error: 'create_user_failed', message: createErr?.message ?? 'unknown' })
  }

  const funcionarioId = created.user.id

  const insertPayload = {
    id: funcionarioId,
    usuario_master_id: masterId,
    nome_completo: nomeCompleto,
    email,
    telefone: payload.telefone ? String(payload.telefone).trim() : null,
    permissao: payload.permissao,
    horario_inicio: payload.horario_inicio ?? null,
    horario_fim: payload.horario_fim ?? null,
    dias_trabalho: payload.dias_trabalho ?? null,
    intervalo_inicio: payload.intervalo_inicio ?? null,
    intervalo_fim: payload.intervalo_fim ?? null,
    pode_ver_agenda: payload.pode_ver_agenda ?? true,
    pode_criar_agendamentos: payload.pode_criar_agendamentos ?? true,
    pode_cancelar_agendamentos: payload.pode_cancelar_agendamentos ?? true,
    pode_bloquear_horarios: payload.pode_bloquear_horarios ?? true,
    pode_ver_financeiro: payload.pode_ver_financeiro ?? false,
    pode_gerenciar_servicos: payload.pode_gerenciar_servicos ?? false,
    pode_ver_clientes_de_outros: payload.pode_ver_clientes_de_outros ?? false,
    ativo: payload.ativo ?? true,
  }

  const { error: insertErr } = await adminClient.from('funcionarios').insert(insertPayload)
  if (insertErr) {
    await adminClient.auth.admin.deleteUser(funcionarioId)
    return jsonResponse(400, { error: 'insert_funcionario_failed', message: insertErr.message })
  }

  return jsonResponse(200, { id: funcionarioId, email })
})
```

smagenda/supabase/functions/payments/config.toml

```
verify_jwt = false
```

smagenda/supabase/functions/payments/index.ts

```
import { createClient } from 'https://esm.sh/@supabase/supabase-js@2.89.0'

type Payload =
  | { action: 'create_checkout'; usuario_id: string; plano: string; metodo?: 'card' | 'pix' | null; funcionarios_total?: number | null }
  | { action: 'sync_checkout_session'; session_id: string; usuario_id?: string | null }
  | { action: 'create_billing_portal'; usuario_id: string }
  | {
      action: 'create_booking_fee_checkout'
      usuario_id: string
      servico_id: string
      data: string
      hora_inicio: string
      cliente_nome: string
      cliente_telefone: string
      cliente_endereco?: string | null
      extras?: Record<string, unknown> | null
      funcionario_id?: string | null
      unidade_id?: string | null
      slug?: string | null
      unidade_slug?: string | null
    }
  | { action: 'sync_booking_fee_session'; session_id: string }

type StripeEvent = {
  id?: string
  type?: string
  data?: { object?: unknown }
}

function jsonResponse(status: number, body: unknown) {
  return new Response(JSON.stringify(body), {
    status,
    headers: {
      'Content-Type': 'application/json',
      'Access-Control-Allow-Origin': '*',
      'Access-Control-Allow-Headers': 'authorization, x-client-info, apikey, content-type',
      'Access-Control-Allow-Methods': 'POST, OPTIONS',
    },
  })
}

function normalizeUuid(value: unknown) {
  const s = typeof value === 'string' ? value.trim() : ''
  if (!s) return null
  if (!/^[0-9a-fA-F-]{36}$/.test(s)) return null
  return s
}

function cleanUrl(input: string) {
  const raw = String(input ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
  if (!raw) return ''
  const withProto = /^https?:\/\//i.test(raw) ? raw : `https://${raw}`
  return withProto.replace(/\/+$/g, '')
}

function cleanEnvValue(input: string) {
  return String(input ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
}

function toHex(bytes: ArrayBuffer) {
  const arr = new Uint8Array(bytes)
  let out = ''
  for (let i = 0; i < arr.length; i += 1) out += arr[i].toString(16).padStart(2, '0')
  return out
}
```

```
function hexToBytes(hex: string) {
  const normalized = hex.trim().toLowerCase()
  if (!/^[0-9a-f]+$/.test(normalized) || normalized.length % 2 !== 0) return null
  const out = new Uint8Array(normalized.length / 2)
  for (let i = 0; i < out.length; i += 1) {
    out[i] = Number.parseInt(normalized.slice(i * 2, i * 2 + 2), 16)
  }
  return out
}

function secureEqualBytes(a: Uint8Array, b: Uint8Array) {
  if (a.length !== b.length) return false
  let diff = 0
  for (let i = 0; i < a.length; i += 1) diff |= a[i] ^ b[i]
  return diff === 0
}

function parseStripeSignatureHeader(header: string) {
  const parts = header
    .split(',')
    .map((p) => p.trim())
    .filter(Boolean)
  const kv = new Map<string, string[]>()
  for (const p of parts) {
    const idx = p.indexOf('=')
    if (idx <= 0) continue
    const k = p.slice(0, idx).trim()
    const v = p.slice(idx + 1).trim()
    if (!k || !v) continue
    const prev = kv.get(k) ?? []
    prev.push(v)
    kv.set(k, prev)
  }

  const tRaw = (kv.get('t') ?? [])[0] ?? ''
  const t = Number.parseInt(tRaw, 10)
  const v1 = kv.get('v1') ?? []
  if (!Number.isFinite(t) || t <= 0 || v1.length === 0) return null
  return { t, v1 }
}

async function hmacSha256Hex(secret=[REDACTED]
  const enc = new TextEncoder()
  const key = await crypto.subtle.importKey(
    'raw',
    enc.encode(secret),
    { name: 'HMAC', hash: 'SHA-256' },
    false,
    ['sign']
  )
  const sig = await crypto.subtle.sign('HMAC', key, enc.encode(message))
  return toHex(sig)
}

async function verifyStripeSignature({ payload, header, secret }: { payload: string; header: string; secret=[REDACTED]
  const parsed = parseStripeSignatureHeader(header)
  if (!parsed) return { ok: false as const, error: 'invalid_signature_header' as const }
  const signedPayload = `${parsed.t}.${payload}`
  const expectedHex = await hmacSha256Hex(secret, signedPayload)
  const expectedBytes = hexToBytes(expectedHex)
  if (!expectedBytes) return { ok: false as const, error: 'invalid_expected_sig' as const }

  for (const v of parsed.v1) {
    const gotBytes = hexToBytes(v)
    if (!gotBytes) continue
    if (secureEqualBytes(expectedBytes, gotBytes)) return { ok: true as const }
  }

  return { ok: false as const, error: 'signature_mismatch' as const }
}
```

```
async function stripeApiRequest(path: string, opts: { method: 'GET' | 'POST'; key: string; params?: URLSearchParams }) {
  const url = `https://api.stripe.com/v1${path}`
  const res = await fetch(url, {
    method: opts.method,
    headers: {
      Authorization: `Bearer ${opts.key}`,
      ...(opts.method === 'POST' ? { 'Content-Type': 'application/x-www-form-urlencoded' } : {}),
    },
    body: opts.method === 'POST' ? opts.params?.toString() ?? '' : undefined,
  })

  const text = await res.text()
  let parsed: unknown = null
  try {
    parsed = text ? (JSON.parse(text) as unknown) : null
  } catch {
    parsed = text
  }
  return { ok: res.ok, status: res.status, body: parsed }
}

async function resolveActivePriceIdForProduct(input: {
  productId: string
  key: string
  expectedMode: 'subscription' | 'payment'
  currency?: string
}) {
  const productId = (input.productId ?? '').trim()
  if (!productId) return null
  const currency = (input.currency ?? 'brl').trim().toLowerCase()
  const expectedType = input.expectedMode === 'subscription' ? 'recurring' : 'one_time'

  const res = await stripeApiRequest(`/prices?active=true&product=${encodeURIComponent(productId)}&limit=100`, {
    method: 'GET',
    key: input.key,
  })
  if (!res.ok || !res.body || typeof res.body !== 'object') return null
  const obj = res.body as Record<string, unknown>
  const data = obj.data
  if (!Array.isArray(data) || data.length === 0) return null

  const candidates: Array<{ id: string; unitAmount: number; intervalRank: number }> = []

  for (const p of data) {
    if (!p || typeof p !== 'object') continue
    const price = p as Record<string, unknown>
    const id = typeof price.id === 'string' ? price.id : ''
    if (!id) continue
    if (price.active !== true) continue
    if (typeof price.type !== 'string' || price.type !== expectedType) continue
    const cur = typeof price.currency === 'string' ? price.currency.toLowerCase() : ''
    if (currency && cur && cur !== currency) continue

    let intervalRank = 0
    if (expectedType === 'recurring') {
      const recurring = price.recurring && typeof price.recurring === 'object' ? (price.recurring as Record<string,
unknown>) : null
      const interval = typeof recurring?.interval === 'string' ? recurring.interval : ''
      intervalRank = interval === 'month' ? 2 : interval ? 1 : 0
    }

    const unitAmount = typeof price.unit_amount === 'number' && Number.isFinite(price.unit_amount) ? price.unit_amount :
Number.POSITIVE_INFINITY
    candidates.push({ id, unitAmount, intervalRank })
  }

  if (candidates.length === 0) return null
  candidates.sort((a, b) => {
    if (b.intervalRank !== a.intervalRank) return b.intervalRank - a.intervalRank
    if (a.unitAmount !== b.unitAmount) return a.unitAmount - b.unitAmount
    return a.id.localeCompare(b.id)
  })
```

```
  return candidates[0]?.id ?? null
}

function toIsoDateFromUnixSeconds(value: unknown) {
  const n = typeof value === 'number' ? value : typeof value === 'string' ? Number(value) : NaN
  if (!Number.isFinite(n) || n <= 0) return null
  return new Date(n * 1000).toISOString().slice(0, 10)
}

function toIsoDatePlusDays(days: number) {
  const n = Number(days)
  if (!Number.isFinite(n) || n <= 0) return null
  const d = new Date()
  d.setHours(0, 0, 0, 0)
  d.setDate(d.getDate() + Math.floor(n))
  return d.toISOString().slice(0, 10)
}

function mapStripeSubscriptionToPagamentoStatus(statusRaw: unknown) {
  const s = typeof statusRaw === 'string' ? statusRaw.trim().toLowerCase() : ''
  if (s === 'active' || s === 'trialing') return 'ativo'
  if (s === 'past_due' || s === 'unpaid') return 'inadimplente'
  if (s === 'canceled' || s === 'incomplete_expired') return 'cancelado'
  if (s === 'incomplete' || s === 'paused') return 'suspenso'
  return 'suspenso'
}

function resolveLimiteFuncionariosFromPlano(planoRaw: unknown) {
  const plano = typeof planoRaw === 'string' ? planoRaw.trim().toLowerCase() : ''
  if (!plano) return undefined
  if (plano === 'enterprise') return 10
  if (plano === 'team') return 4
  if (plano === 'pro') return 4
  if (plano === 'basic') return 1
  if (plano === 'free') return 1
  return undefined
}

function resolveLimiteAgendamentosMesFromPlano(planoRaw: unknown) {
  const plano = typeof planoRaw === 'string' ? planoRaw.trim().toLowerCase() : ''
  if (!plano) return undefined
  if (plano === 'enterprise') return null
  if (plano === 'team') return 300
  if (plano === 'pro') return 180
  if (plano === 'basic') return 60
  if (plano === 'free') return 30
  return undefined
}

function resolvePlanoFromMetadata(metadataRaw: unknown) {
  const metadata = (metadataRaw ?? null) as Record<string, unknown> | null
  const itemRaw = typeof metadata?.item === 'string' ? metadata.item : typeof metadata?.plano === 'string' ?
metadata.plano : null
  const item = typeof itemRaw === 'string' ? itemRaw.trim().toLowerCase() : ''
  if (!item) return { item: null as string | null, plano: null as string | null }
  const plano = ['basic', 'pro', 'team', 'enterprise'].includes(item) ? item : null
  return { item, plano }
}

function parsePositiveInt(value: unknown) {
  const n = typeof value === 'number' ? value : typeof value === 'string' ? Number(value) : NaN
  if (!Number.isFinite(n)) return null
  const i = Math.floor(n)
  if (i <= 0) return null
  return i
}

function clampInt(value: number, min: number, max: number) {
  const n = Math.floor(value)
  if (n < min) return min
  if (n > max) return max
  return n
```

```
}

function resolveFuncionariosTotalFromMetadata(metadataRaw: unknown) {
  const metadata = (metadataRaw ?? null) as Record<string, unknown> | null
  const raw = metadata?.funcionarios_total ?? metadata?.funcionarios ?? metadata?.qtd_funcionarios
  const parsed = parsePositiveInt(raw)
  if (!parsed) return null
  return clampInt(parsed, 1, 200)
}

function resolveExtraEmployeesQtyFromSubscription(subscriptionRaw: unknown, opts: { extraPriceId?: string;
extraProductId?: string }) {
  if (!subscriptionRaw || typeof subscriptionRaw !== 'object') return null
  const subscription = subscriptionRaw as Record<string, unknown>
  const items = subscription.items
  if (!items || typeof items !== 'object') return null
  const data = (items as Record<string, unknown>).data
  if (!Array.isArray(data)) return null

  const extraPriceId = (opts.extraPriceId ?? '').trim()
  const extraProductId = (opts.extraProductId ?? '').trim()
  if (!extraPriceId && !extraProductId) return null

  let sum = 0
  for (const it of data) {
    if (!it || typeof it !== 'object') continue
    const row = it as Record<string, unknown>
    const qty = parsePositiveInt(row.quantity) ?? 0
    if (qty <= 0) continue

    const priceRaw = row.price
    const priceObj = priceRaw && typeof priceRaw === 'object' ? (priceRaw as Record<string, unknown>) : null
    const priceId = typeof priceRaw === 'string' ? priceRaw : typeof priceObj?.id === 'string' ? priceObj.id : ''

    const productRaw = priceObj?.product
    const productObj = productRaw && typeof productRaw === 'object' ? (productRaw as Record<string, unknown>) : null
    const productId = typeof productRaw === 'string' ? productRaw : typeof productObj?.id === 'string' ? productObj.id :
''

    const matchesPrice = extraPriceId && priceId === extraPriceId
    const matchesProduct = extraProductId && productId === extraProductId
    if (!matchesPrice && !matchesProduct) continue

    sum += qty
  }

  return sum
}

function resolveLimiteFuncionariosFromStripe(input: {
  plano: string | null
  metadata: unknown
  subscription?: unknown
  extraPriceId?: string
  extraProductId?: string
}) {
  const plano = typeof input.plano === 'string' ? input.plano.trim().toLowerCase() : ''
  if (!plano) return resolveLimiteFuncionariosFromPlano(input.plano)
  if (plano === 'basic' || plano === 'free') return resolveLimiteFuncionariosFromPlano(plano)

  const base = resolveLimiteFuncionariosFromPlano(plano)
  if (base === null) return null
  const baseIncluded = typeof base === 'number' && Number.isFinite(base) && base > 0 ? Math.floor(base) : 1

  const fromMeta = resolveFuncionariosTotalFromMetadata(input.metadata)
  const extraQty = resolveExtraEmployeesQtyFromSubscription(input.subscription, {
    extraPriceId: input.extraPriceId,
    extraProductId: input.extraProductId,
  })

  const max = plano === 'pro' || plano === 'team' ? 6 : plano === 'enterprise' ? 10 : 200
```

```
  if (typeof extraQty === 'number') {
    const total = baseIncluded + Math.max(0, extraQty)
    return clampInt(total, 1, max)
  }

  if (fromMeta) return clampInt(Math.max(baseIncluded, fromMeta), 1, max)
  return resolveLimiteFuncionariosFromPlano(plano)
}

function resolvePaymentStatus(value: unknown) {
  const v = typeof value === 'string' ? value.trim().toLowerCase() : ''
  if (!v) return null
  if (v === 'paid' || v === 'no_payment_required') return 'paid'
  if (v === 'unpaid') return 'unpaid'
  return null
}

async function findUsuarioIdByStripe(adminClient: ReturnType<typeof createClient>, input: { subscriptionId?: string |
null; customerId?: string | null }) {
  const subscriptionId = (input.subscriptionId ?? '').trim()
  const customerId = (input.customerId ?? '').trim()

  if (subscriptionId) {
    const { data, error } = await adminClient
      .from('usuarios')
      .select('id')
      .eq('stripe_subscription_id', subscriptionId)
      .maybeSingle()
    if (!error && data?.id) return { ok: true as const, usuarioId: String((data as { id: string }).id) }
  }
  if (customerId) {
    const { data, error } = await adminClient
      .from('usuarios')
      .select('id')
      .eq('stripe_customer_id', customerId)
      .maybeSingle()
    if (!error && data?.id) return { ok: true as const, usuarioId: String((data as { id: string }).id) }
  }

  return { ok: false as const }
}

async function applyUsuarioPaymentUpdate(
  adminClient: ReturnType<typeof createClient>,
  usuarioId: string,
  update: {
    plano?: string | null
    limite_funcionarios?: number | null
    limite_agendamentos_mes?: number | null
    status_pagamento?: string | null
    data_vencimento?: string | null
    free_trial_consumido?: boolean | null
    stripe_customer_id?: string | null
    stripe_subscription_id?: string | null
    stripe_checkout_session_id?: string | null
    stripe_last_event_id?: string | null
    stripe_last_event_at?: string | null
  }
) {
  const payload: Record<string, unknown> = {}
  if (typeof update.plano === 'string' && update.plano.trim()) payload.plano = update.plano.trim().toLowerCase()
  if (typeof update.limite_funcionarios === 'number' && Number.isFinite(update.limite_funcionarios))
payload.limite_funcionarios = update.limite_funcionarios
  if (update.limite_funcionarios === null) payload.limite_funcionarios = null
  if (typeof update.limite_agendamentos_mes === 'number' && Number.isFinite(update.limite_agendamentos_mes))
payload.limite_agendamentos_mes = update.limite_agendamentos_mes
  if (update.limite_agendamentos_mes === null) payload.limite_agendamentos_mes = null
  if (typeof update.status_pagamento === 'string' && update.status_pagamento.trim()) payload.status_pagamento =
update.status_pagamento.trim()
  if (typeof update.data_vencimento === 'string' && update.data_vencimento.trim()) payload.data_vencimento =
update.data_vencimento.trim()
  if (update.free_trial_consumido === true) payload.free_trial_consumido = true
```

```
  if (update.free_trial_consumido === false) payload.free_trial_consumido = false
  if (typeof update.stripe_customer_id === 'string' && update.stripe_customer_id.trim()) payload.stripe_customer_id =
update.stripe_customer_id.trim()
  if (typeof update.stripe_subscription_id === 'string' && update.stripe_subscription_id.trim())
payload.stripe_subscription_id = update.stripe_subscription_id.trim()
  if (typeof update.stripe_checkout_session_id === 'string' && update.stripe_checkout_session_id.trim())
payload.stripe_checkout_session_id = update.stripe_checkout_session_id.trim()
  if (typeof update.stripe_last_event_id === 'string' && update.stripe_last_event_id.trim())
payload.stripe_last_event_id = update.stripe_last_event_id.trim()
  if (typeof update.stripe_last_event_at === 'string' && update.stripe_last_event_at.trim())
payload.stripe_last_event_at = update.stripe_last_event_at.trim()

  if (Object.keys(payload).length === 0) return { ok: true as const }

  const { error } = await adminClient.from('usuarios').update(payload).eq('id', usuarioId)
  if (error) return { ok: false as const, error: error.message }
  return { ok: true as const }
}

async function stripeRequest(params: URLSearchParams, key: string) {
  const res = await fetch('https://api.stripe.com/v1/checkout/sessions', {
    method: 'POST',
    headers: {
      Authorization: `Bearer ${key}`,
      'Content-Type': 'application/x-www-form-urlencoded',
    },
    body: params.toString(),
  })
  const text = await res.text()
  let parsed: unknown = null
  try {
    parsed = text ? (JSON.parse(text) as unknown) : null
  } catch {
    parsed = text
  }
  return { ok: res.ok, status: res.status, body: parsed }
}

function parseNumeric(value: unknown) {
  if (typeof value === 'number' && Number.isFinite(value)) return value
  if (typeof value === 'string') {
    const n = Number(value)
    if (Number.isFinite(n)) return n
  }
  return null
}

function toCentsBRL(value: number) {
  const n = Number(value)
  if (!Number.isFinite(n)) return null
  const cents = Math.round(n * 100)
  if (!Number.isFinite(cents) || cents <= 0) return null
  return cents
}

function buildPublicBookingReturnPath(input: { slug: string; unidadeSlug?: string | null }) {
  const slug = (input.slug ?? '').trim().replace(/^\/+|\/+$/g, '')
  if (!slug) return null
  const u = (input.unidadeSlug ?? '').trim().replace(/^\/+|\/+$/g, '')
  return u ? `/agendar/${encodeURIComponent(slug)}/${encodeURIComponent(u)}` : `/agendar/${encodeURIComponent(slug)}`
}

function isStripePaidSession(session: Record<string, unknown>) {
  const paymentStatus = resolvePaymentStatus(session.payment_status)
  if (paymentStatus === 'paid' || paymentStatus === 'no_payment_required') return true
  return false
}

async function callPublicCreateAgendamentoWithFallback(
  adminClient: ReturnType<typeof createClient>,
  args: Record<string, unknown>
) {
```

```
  const lowerMsg = (e: { message?: string }) => String(e?.message ?? '').toLowerCase()
  const hasExtras = 'p_extras' in args && args.p_extras !== null && args.p_extras !== undefined
  const shouldFallback = (msg: string) => {
    const missingSignature =
      msg.includes('public_create_agendamento_publico') && (msg.includes('does not exist') || msg.includes('function')
|| msg.includes('rpc'))
    const badParam =
      msg.includes('parameter') ||
      msg.includes('unknown') ||
      msg.includes('p_unidade_id') ||
      msg.includes('p_status') ||
      msg.includes('p_extras')
    return missingSignature || badParam
  }

  const dropKeys = (obj: Record<string, unknown>, keys: string[]) => {
    const out: Record<string, unknown> = { ...obj }
    for (const k of keys) {
      if (k in out) delete out[k]
    }
    return out
  }

  const baseVariants: Record<string, unknown>[] = [
    { ...args },
    dropKeys(args, ['p_unidade_id']),
    dropKeys(args, ['p_status']),
    dropKeys(args, ['p_unidade_id', 'p_status']),
  ]

  const variants: Record<string, unknown>[] = hasExtras
    ? baseVariants
    : [
        ...baseVariants,
        dropKeys(args, ['p_extras']),
        dropKeys(args, ['p_unidade_id', 'p_extras']),
        dropKeys(args, ['p_status', 'p_extras']),
        dropKeys(args, ['p_unidade_id', 'p_status', 'p_extras']),
      ]

  let lastError: { message?: string } | null = null
  for (const v of variants) {
    const { data, error } = await adminClient.rpc('public_create_agendamento_publico', v)
    if (!error) return { ok: true as const, data }
    lastError = error
    const msg = lowerMsg(error)
    if (!shouldFallback(msg)) return { ok: false as const, error }
  }

  return { ok: false as const, error: lastError as { message: string } }
}

function isMissingColumnError(error: { message?: string } | null, column: string) {
  const msg = String(error?.message ?? '')
  const lower = msg.toLowerCase()
  const col = column.toLowerCase()
  if (!lower.includes(col)) return false
  if (lower.includes('column') && lower.includes('does not exist')) return true
  if (lower.includes('schema cache') && lower.includes(col)) return true
  if (lower.includes('could not find') && lower.includes(col)) return true
  return false
}

async function upsertTaxaAgendamentoPagamentoWithFallback(
  adminClient: ReturnType<typeof createClient>,
  row: Record<string, unknown>,
  onConflict: string
) {
  const first = await adminClient.from('taxa_agendamento_pagamentos').upsert(row, { onConflict })
  if (!first.error) return { ok: true as const }
  if (!isMissingColumnError(first.error, 'unidade_id')) return { ok: false as const, error: first.error }
  const withoutUnidade: Record<string, unknown> = { ...row }
```

```
  delete withoutUnidade.unidade_id
  const second = await adminClient.from('taxa_agendamento_pagamentos').upsert(withoutUnidade, { onConflict })
  if (second.error) return { ok: false as const, error: second.error }
  return { ok: true as const }
}

async function updateTaxaAgendamentoPagamentoWithFallback(
  adminClient: ReturnType<typeof createClient>,
  id: string,
  patch: Record<string, unknown>
) {
  const first = await adminClient.from('taxa_agendamento_pagamentos').update(patch).eq('id', id)
  if (!first.error) return { ok: true as const }
  if (!isMissingColumnError(first.error, 'unidade_id')) return { ok: false as const, error: first.error }
  const withoutUnidade: Record<string, unknown> = { ...patch }
  delete withoutUnidade.unidade_id
  const second = await adminClient.from('taxa_agendamento_pagamentos').update(withoutUnidade).eq('id', id)
  if (second.error) return { ok: false as const, error: second.error }
  return { ok: true as const }
}

async function insertTaxaAgendamentoPagamentoWithFallback(adminClient: ReturnType<typeof createClient>, row:
Record<string, unknown>) {
  const first = await adminClient.from('taxa_agendamento_pagamentos').insert(row)
  if (!first.error) return { ok: true as const }
  if (!isMissingColumnError(first.error, 'unidade_id')) return { ok: false as const, error: first.error }
  const withoutUnidade: Record<string, unknown> = { ...row }
  delete withoutUnidade.unidade_id
  const second = await adminClient.from('taxa_agendamento_pagamentos').insert(withoutUnidade)
  if (second.error) return { ok: false as const, error: second.error }
  return { ok: true as const }
}

async function handlePublicBookingFeeCheckout(input: {
  adminClient: ReturnType<typeof createClient>
  stripeKey: string
  siteUrl: string
  payload: Payload & { action: 'create_booking_fee_checkout' }
}) {
  const usuarioId = normalizeUuid(input.payload.usuario_id)
  const servicoId = normalizeUuid(input.payload.servico_id)
  const funcionarioId = normalizeUuid(input.payload.funcionario_id)
  const unidadeId = normalizeUuid(input.payload.unidade_id)
  const data = String(input.payload.data ?? '').trim()
  const horaInicio = String(input.payload.hora_inicio ?? '').trim()
  const clienteNome = String(input.payload.cliente_nome ?? '').trim()
  const clienteTelefone = String(input.payload.cliente_telefone ?? '').trim()
  const clienteEndereco = String(input.payload.cliente_endereco ?? '').trim()
  const slug = String(input.payload.slug ?? '').trim()
  const unidadeSlug = String(input.payload.unidade_slug ?? '').trim()

  const extrasRaw = input.payload.extras
  const extrasObj = extrasRaw && typeof extrasRaw === 'object' && !Array.isArray(extrasRaw) ? (extrasRaw as
Record<string, unknown>) : null
  const extras = clienteEndereco ? { ...(extrasObj ?? {}), endereco: clienteEndereco } : extrasObj
  const extrasFinal = extras && Object.keys(extras).length > 0 ? extras : null

  if (!usuarioId || !servicoId || !data || !horaInicio || !clienteNome || !clienteTelefone || !slug) {
    return jsonResponse(400, { error: 'invalid_payload' })
  }

  const returnPath = buildPublicBookingReturnPath({ slug, unidadeSlug: unidadeSlug || null })
  if (!returnPath) return jsonResponse(400, { error: 'invalid_slug' })

  const { data: usuarioRow, error: usuarioErr } = await input.adminClient
    .from('usuarios')
    .select('id,slug,ativo')
    .eq('slug', slug)
    .maybeSingle()
  if (usuarioErr || !usuarioRow || usuarioRow.ativo !== true) {
    return jsonResponse(404, { error: 'usuario_not_found' })
  }
```

```
  if (String(usuarioRow.id) !== usuarioId) {
    return jsonResponse(400, { error: 'usuario_mismatch' })
  }

  if (unidadeSlug) {
    const { data: unRow, error: unErr } = await input.adminClient
      .from('unidades')
      .select('id,slug,ativo,usuario_id')
      .eq('usuario_id', usuarioId)
      .eq('slug', unidadeSlug)
      .maybeSingle()
    if (unErr) return jsonResponse(400, { error: 'unidade_error', message: unErr.message })
    if (!unRow || unRow.ativo !== true) return jsonResponse(404, { error: 'unidade_not_found' })
    if (!unidadeId) {
      return jsonResponse(400, { error: 'missing_unidade_id' })
    }
    if (String(unRow.id) !== unidadeId) {
      return jsonResponse(400, { error: 'unidade_mismatch' })
    }
  }

  const { data: servicoRow, error: servicoErr } = await input.adminClient
    .from('servicos')
    .select('id,usuario_id,ativo,nome,taxa_agendamento')
    .eq('id', servicoId)
    .eq('usuario_id', usuarioId)
    .maybeSingle()
  if (servicoErr) return jsonResponse(400, { error: 'servico_error', message: servicoErr.message })
  if (!servicoRow || servicoRow.ativo !== true) return jsonResponse(404, { error: 'servico_not_found' })

  const taxa = parseNumeric((servicoRow as Record<string, unknown>).taxa_agendamento) ?? 0
  const taxaNorm = Math.max(0, taxa)
  const cents = toCentsBRL(taxaNorm)
  if (!cents) return jsonResponse(400, { error: 'invalid_fee' })

  const { data: creditIdRaw, error: creditErr } = await input.adminClient.rpc('consume_taxa_agendamento_credito', {
    p_usuario_id: usuarioId,
    p_cliente_telefone: clienteTelefone,
  })
  if (creditErr) {
    return jsonResponse(400, { error: 'credit_error', message: creditErr.message })
  }
  const creditId = normalizeUuid(creditIdRaw)
  if (creditId) {
    const createArgs: Record<string, unknown> = {
      p_usuario_id: usuarioId,
      p_data: data,
      p_hora_inicio: horaInicio,
      p_servico_id: servicoId,
      p_cliente_nome: clienteNome,
      p_cliente_telefone: clienteTelefone,
      p_funcionario_id: funcionarioId,
      p_status: 'confirmado',
    }
    if (extrasFinal) createArgs.p_extras = extrasFinal
    if (unidadeId) createArgs.p_unidade_id = unidadeId

    const createRes = await callPublicCreateAgendamentoWithFallback(input.adminClient, createArgs)
    if (!createRes.ok) {
      return jsonResponse(400, { error: 'create_agendamento_failed', message: createRes.error.message })
    }
    const agendamentoId = normalizeUuid(createRes.data)
    if (!agendamentoId) return jsonResponse(400, { error: 'invalid_agendamento_id' })

    await updateTaxaAgendamentoPagamentoWithFallback(input.adminClient, creditId, {
      status: 'usado',
      usado_em: new Date().toISOString(),
      utilizado_em_agendamento_id: agendamentoId,
      servico_id: servicoId,
      funcionario_id: funcionarioId,
      unidade_id: unidadeId,
    })
```

```
    return jsonResponse(200, { ok: true, kind: 'credit', agendamento_id: agendamentoId })
  }

  const createArgs: Record<string, unknown> = {
    p_usuario_id: usuarioId,
    p_data: data,
    p_hora_inicio: horaInicio,
    p_servico_id: servicoId,
    p_cliente_nome: clienteNome,
    p_cliente_telefone: clienteTelefone,
    p_funcionario_id: funcionarioId,
    p_status: 'pendente',
  }
  if (extrasFinal) createArgs.p_extras = extrasFinal
  if (unidadeId) createArgs.p_unidade_id = unidadeId

  const createRes = await callPublicCreateAgendamentoWithFallback(input.adminClient, createArgs)
  if (!createRes.ok) {
    return jsonResponse(400, { error: 'create_agendamento_failed', message: createRes.error.message })
  }
  const agendamentoId = normalizeUuid(createRes.data)
  if (!agendamentoId) return jsonResponse(400, { error: 'invalid_agendamento_id' })

  const successUrl = `${input.siteUrl}${returnPath}?paid=1&session_id={CHECKOUT_SESSION_ID}`
  const cancelUrl = `${input.siteUrl}${returnPath}?canceled=1`

  const params = new URLSearchParams()
  params.set('mode', 'payment')
  params.set('success_url', successUrl)
  params.set('cancel_url', cancelUrl)
  params.set('client_reference_id', agendamentoId)
  params.set('metadata[kind]', 'booking_fee')
  params.set('metadata[usuario_id]', usuarioId)
  params.set('metadata[agendamento_id]', agendamentoId)
  params.set('metadata[servico_id]', servicoId)
  if (funcionarioId) params.set('metadata[funcionario_id]', funcionarioId)
  if (unidadeId) params.set('metadata[unidade_id]', unidadeId)
  params.set('payment_intent_data[metadata][kind]', 'booking_fee')
  params.set('payment_intent_data[metadata][usuario_id]', usuarioId)
  params.set('payment_intent_data[metadata][agendamento_id]', agendamentoId)
  params.set('payment_intent_data[metadata][servico_id]', servicoId)
  if (funcionarioId) params.set('payment_intent_data[metadata][funcionario_id]', funcionarioId)
  if (unidadeId) params.set('payment_intent_data[metadata][unidade_id]', unidadeId)

  params.set('payment_method_types[0]', 'card')
  params.set('line_items[0][price_data][currency]', 'brl')
  params.set('line_items[0][price_data][product_data][name]', `Taxa de agendamento - ${String((servicoRow as Record<string, unknown>).nome ?? 'Serviço')}`)
  params.set('line_items[0][price_data][unit_amount]', String(cents))
  params.set('line_items[0][quantity]', '1')
  params.set('expand[0]', 'payment_intent')

  const stripeRes = await stripeRequest(params, input.stripeKey)
  if (!stripeRes.ok || !stripeRes.body || typeof stripeRes.body !== 'object') {
    await input.adminClient.from('agendamentos').delete().eq('id', agendamentoId).eq('usuario_id', usuarioId)
    const stripeMessage = (() => {
      if (!stripeRes.body || typeof stripeRes.body !== 'object') return null
      const body = stripeRes.body as Record<string, unknown>
      const err = body.error
      if (!err || typeof err !== 'object') return null
      const msg = (err as Record<string, unknown>).message
      return typeof msg === 'string' && msg.trim() ? msg.trim() : null
    })()
    return jsonResponse(400, { error: 'stripe_error', message: stripeMessage ?? 'Falha ao criar checkout.', stripe: stripeRes.body })
  }

  const sessionObj = stripeRes.body as Record<string, unknown>
  const checkoutUrl = typeof sessionObj.url === 'string' ? sessionObj.url : null
  const sessionId = typeof sessionObj.id === 'string' ? sessionObj.id : null
  const piRaw = sessionObj.payment_intent
```

```
  const paymentIntentId = typeof piRaw === 'string' ? piRaw : piRaw && typeof piRaw === 'object' && typeof (piRaw as Record<string, unknown>).id === 'string' ? String((piRaw as Record<string, unknown>).id) : null
  if (!checkoutUrl || !sessionId) {
    await input.adminClient.from('agendamentos').delete().eq('id', agendamentoId).eq('usuario_id', usuarioId)
    return jsonResponse(400, { error: 'missing_url' })
  }

  const feeValue = taxaNorm
  const persist = await upsertTaxaAgendamentoPagamentoWithFallback(
    input.adminClient,
    {
      usuario_id: usuarioId,
      agendamento_id: agendamentoId,
      servico_id: servicoId,
      funcionario_id: funcionarioId,
      unidade_id: unidadeId,
      cliente_nome: clienteNome,
      cliente_telefone: clienteTelefone,
      valor: feeValue,
      moeda: 'brl',
      status: 'pendente',
      stripe_checkout_session_id: sessionId,
      stripe_payment_intent_id: paymentIntentId,
    },
    'stripe_checkout_session_id'
  )
  if (!persist.ok) {
    await input.adminClient.from('agendamentos').delete().eq('id', agendamentoId).eq('usuario_id', usuarioId)
    return jsonResponse(400, { error: 'db_error', message: persist.error.message })
  }

  return jsonResponse(200, {
    ok: true,
    kind: 'checkout',
    checkout_url: checkoutUrl,
    session_id: sessionId,
    intent_id: paymentIntentId,
    agendamento_id: agendamentoId,
  })
}

async function finalizeBookingFeeFromStripeSession(input: {
  adminClient: ReturnType<typeof createClient>
  stripeKey: string
  sessionId: string
}) {
  const sessionRes = await stripeApiRequest(`/checkout/sessions/${encodeURIComponent(input.sessionId)}?expand[]=payment_intent`, {
    method: 'GET',
    key: input.stripeKey,
  })
  if (!sessionRes.ok || !sessionRes.body || typeof sessionRes.body !== 'object') {
    return { ok: false as const, error: 'stripe_error' as const, stripe: sessionRes.body }
  }

  const session = sessionRes.body as Record<string, unknown>
  if (!isStripePaidSession(session)) {
    return { ok: false as const, error: 'not_paid' as const }
  }

  const metadata = (session.metadata ?? null) as Record<string, unknown> | null
  const kind = typeof metadata?.kind === 'string' ? metadata.kind.trim().toLowerCase() : ''
  if (kind !== 'booking_fee') {
    return { ok: false as const, error: 'invalid_kind' as const }
  }
  const usuarioId = normalizeUuid(metadata?.usuario_id)
  const agendamentoId = normalizeUuid(metadata?.agendamento_id)
  const servicoId = normalizeUuid(metadata?.servico_id)
  const funcionarioId = normalizeUuid(metadata?.funcionario_id)
  const unidadeId = normalizeUuid(metadata?.unidade_id)

  if (!usuarioId || !agendamentoId) {
```

```
    return { ok: false as const, error: 'missing_metadata' as const }
  }

  const piRaw = session.payment_intent
  const paymentIntentId =
    typeof piRaw === 'string'
      ? piRaw
      : piRaw && typeof piRaw === 'object' && typeof (piRaw as Record<string, unknown>).id === 'string'
        ? String((piRaw as Record<string, unknown>).id)
        : null

  const nowIso = new Date().toISOString()

  await input.adminClient.from('agendamentos').update({ status: 'confirmado' }).eq('id', agendamentoId).eq('usuario_id',
usuarioId)

  const amountTotal = typeof session.amount_total === 'number' ? session.amount_total : typeof session.amount_total ===
'string' ? Number(session.amount_total) : null
  const valor = typeof amountTotal === 'number' && Number.isFinite(amountTotal) ? amountTotal / 100 : null

  const { data: existing } = await input.adminClient
    .from('taxa_agendamento_pagamentos')
    .select('id')
    .eq('stripe_checkout_session_id', input.sessionId)
    .maybeSingle()

  if (existing?.id) {
    const upd = await updateTaxaAgendamentoPagamentoWithFallback(input.adminClient, String(existing.id), {
      status: 'pago',
      pago_em: nowIso,
      stripe_payment_intent_id: paymentIntentId,
      servico_id: servicoId,
      funcionario_id: funcionarioId,
      unidade_id: unidadeId,
    })
    if (!upd.ok) return { ok: false as const, error: 'db_error' as const }
  } else {
    const ins = await insertTaxaAgendamentoPagamentoWithFallback(input.adminClient, {
      usuario_id: usuarioId,
      agendamento_id: agendamentoId,
      servico_id: servicoId,
      funcionario_id: funcionarioId,
      unidade_id: unidadeId,
      valor: valor ?? 0,
      moeda: 'brl',
      status: 'pago',
      stripe_checkout_session_id: input.sessionId,
      stripe_payment_intent_id: paymentIntentId,
      pago_em: nowIso,
    })
    if (!ins.ok) return { ok: false as const, error: 'db_error' as const }
  }

  return { ok: true as const, agendamentoId }
}

Deno.serve(async (req) => {
  if (req.method === 'OPTIONS') return jsonResponse(200, { ok: true })
  if (req.method !== 'POST') return jsonResponse(405, { error: 'method_not_allowed' })

  const supabaseUrl = cleanEnvValue(Deno.env.get('SUPABASE_URL') ?? '')
  const anonKey = cleanEnvValue(Deno.env.get('SUPABASE_ANON_KEY') ?? '')
  const serviceRoleKey=[REDACTED]'SERVICE_ROLE_KEY') ?? Deno.env.get('SUPABASE_SERVICE_ROLE_KEY') ?? '')
  const stripeKey = cleanEnvValue(Deno.env.get('STRIPE_SECRET_KEY') ?? '')
  const stripeWebhookSecret=[REDACTED]'STRIPE_WEBHOOK_SECRET') ?? '')
  const stripeKeyMode =
    stripeKey.startsWith('sk_live_') || stripeKey.startsWith('rk_live_')
      ? 'live'
      : stripeKey.startsWith('sk_test_') || stripeKey.startsWith('rk_test_')
        ? 'test'
        : null
```

```
  const bookingFeeEnabled = cleanEnvValue(Deno.env.get('ENABLE_BOOKING_FEE') ?? '') === '1'

  if (!supabaseUrl || !anonKey || !serviceRoleKey) {
    return jsonResponse(500, { error: 'missing_env' })
  }
  if (!stripeKey) {
    return jsonResponse(500, { error: 'missing_env', message: 'STRIPE_SECRET_KEY não configurada.' })
  }

  const stripeKeyHasQuery = stripeKey.includes('?') || stripeKey.includes('&') || stripeKey.includes('=') ||
stripeKey.includes('://')
  if (!stripeKeyMode || stripeKeyHasQuery) {
    return jsonResponse(500, {
      error: 'invalid_env',
      message: 'STRIPE_SECRET_KEY inválida. Use sk_live_... (ou rk_live_...) sem aspas, URL ou parâmetros.',
      stripe_key_mode: stripeKeyMode,
    })
  }

  const signatureHeader = req.headers.get('stripe-signature') ?? req.headers.get('Stripe-Signature')
  const isStripeWebhook = Boolean(signatureHeader)

  const adminClient = createClient(supabaseUrl, serviceRoleKey)

  if (isStripeWebhook) {
    if (!stripeWebhookSecret) {
      return jsonResponse(500, { error: 'missing_env', message: 'STRIPE_WEBHOOK_SECRET não configurada.' })
    }
    const rawPayload = await req.text()
    const verified = await verifyStripeSignature({ payload: rawPayload, header: String(signatureHeader), secret=
[REDACTED]
    if (!verified.ok) {
      return jsonResponse(400, { error: 'invalid_signature', reason: verified.error })
    }

    let event: StripeEvent
    try {
      event = (rawPayload ? (JSON.parse(rawPayload) as StripeEvent) : ({} as StripeEvent))
    } catch {
      return jsonResponse(400, { error: 'invalid_json' })
    }

    const eventId = typeof event?.id === 'string' ? event.id : null
    const eventType = typeof event?.type === 'string' ? event.type : null
    const obj = (event?.data?.object ?? null) as Record<string, unknown> | null
    if (!eventType || !obj) return jsonResponse(200, { ok: true })

    const nowIso = new Date().toISOString()

    if (eventType === 'checkout.session.completed' || eventType === 'checkout.session.async_payment_succeeded') {
      const metadata = (obj.metadata ?? null) as Record<string, unknown> | null
      const kind = typeof metadata?.kind === 'string' ? metadata.kind.trim().toLowerCase() : ''
      const sessionId = typeof obj.id === 'string' ? obj.id : null

      if (kind === 'booking_fee' && sessionId) {
        if (!bookingFeeEnabled) return jsonResponse(200, { ok: true })
        const done = await finalizeBookingFeeFromStripeSession({ adminClient, stripeKey, sessionId })
        if (!done.ok) return jsonResponse(200, { ok: true })
        return jsonResponse(200, { ok: true })
      }

      const usuarioId = normalizeUuid(metadata?.usuario_id)
      const { plano } = resolvePlanoFromMetadata(metadata)

      const paymentStatus = resolvePaymentStatus(obj.payment_status)
      const customerId = typeof obj.customer === 'string' ? obj.customer : null
      const subscriptionId = typeof obj.subscription === 'string' ? obj.subscription : null
      const sessionId2 = sessionId

      let finalUsuarioId = usuarioId
      if (!finalUsuarioId) {
        const found = await findUsuarioIdByStripe(adminClient, { subscriptionId, customerId })
```

```
        finalUsuarioId = found.ok ? found.usuarioId : null
      }

      if (finalUsuarioId && plano) {
        let venc: string | null = null
        let statusPagamento: string | null = null
        let subscriptionObj: Record<string, unknown> | null = null

        if (subscriptionId) {
          statusPagamento = 'ativo'
          const subRes = await stripeApiRequest(`/subscriptions/${encodeURIComponent(subscriptionId)}?expand[]=items.data.price.product`, {
            method: 'GET',
            key: stripeKey,
          })
          if (subRes.ok && subRes.body && typeof subRes.body === 'object') {
            subscriptionObj = subRes.body as Record<string, unknown>
            statusPagamento = mapStripeSubscriptionToPagamentoStatus(subscriptionObj.status)
            venc = toIsoDateFromUnixSeconds(subscriptionObj.current_period_end)
          }
        } else if (paymentStatus === 'paid') {
          statusPagamento = 'ativo'
          venc = toIsoDatePlusDays(30)
        }

        const extraPriceId = (Deno.env.get('STRIPE_PRICE_FUNCIONARIO_ADICIONAL') ?? '').trim()
        const extraProductId = 'prod_Tik80yLbCUqUhZ'
        const limiteFuncionarios = resolveLimiteFuncionariosFromStripe({
          plano,
          metadata,
          subscription: subscriptionObj,
          extraPriceId: extraPriceId || undefined,
          extraProductId: extraProductId || undefined,
        })
        const limiteAgendamentosMes = resolveLimiteAgendamentosMesFromPlano(plano)

        if (statusPagamento) {
          await applyUsuarioPaymentUpdate(adminClient, finalUsuarioId, {
            plano,
            limite_funcionarios: limiteFuncionarios,
            limite_agendamentos_mes: limiteAgendamentosMes,
            status_pagamento: statusPagamento,
            data_vencimento: venc,
            free_trial_consumido: true,
            stripe_customer_id: customerId,
            stripe_subscription_id: subscriptionId,
            stripe_checkout_session_id: sessionId2,
            stripe_last_event_id: eventId,
            stripe_last_event_at: nowIso,
          })
        }
      }

      return jsonResponse(200, { ok: true })
    }

    if (eventType === 'invoice.payment_succeeded' || eventType === 'invoice.payment_failed') {
      const subscriptionId = typeof obj.subscription === 'string' ? obj.subscription : null
      const customerId = typeof obj.customer === 'string' ? obj.customer : null
      const found = await findUsuarioIdByStripe(adminClient, { subscriptionId, customerId })
      if (!found.ok) return jsonResponse(200, { ok: true })

      const venc = toIsoDateFromUnixSeconds(obj.current_period_end ?? obj.period_end)
      const statusPagamento = eventType === 'invoice.payment_succeeded' ? 'ativo' : 'inadimplente'

      await applyUsuarioPaymentUpdate(adminClient, found.usuarioId, {
        status_pagamento: statusPagamento,
        data_vencimento: venc,
        stripe_customer_id: customerId,
        stripe_subscription_id: subscriptionId,
        stripe_last_event_id: eventId,
        stripe_last_event_at: nowIso,
```

```
      })

      return jsonResponse(200, { ok: true })
    }

    if (eventType === 'customer.subscription.updated' || eventType === 'customer.subscription.deleted') {
      const subscriptionId = typeof obj.id === 'string' ? obj.id : null
      const customerId = typeof obj.customer === 'string' ? obj.customer : null
      const metadata = (obj.metadata ?? null) as Record<string, unknown> | null
      const usuarioIdFromMeta = normalizeUuid(metadata?.usuario_id)
      const itemRaw = typeof metadata?.item === 'string' ? metadata.item : typeof metadata?.plano === 'string' ?
metadata.plano : null
      const item = typeof itemRaw === 'string' ? itemRaw.trim().toLowerCase() : null
      const plano = item && ['basic', 'pro', 'team', 'enterprise'].includes(item) ? item : null
      const extraPriceId = (Deno.env.get('STRIPE_PRICE_FUNCIONARIO_ADICIONAL') ?? '').trim()
      const extraProductId = 'prod_Tik80yLbCUqUhZ'
      const limiteFuncionarios = resolveLimiteFuncionariosFromStripe({
        plano,
        metadata,
        subscription: obj,
        extraPriceId: extraPriceId || undefined,
        extraProductId: extraProductId || undefined,
      })
      const limiteAgendamentosMes = resolveLimiteAgendamentosMesFromPlano(plano)

      let finalUsuarioId = usuarioIdFromMeta
      if (!finalUsuarioId) {
        const found = await findUsuarioIdByStripe(adminClient, { subscriptionId, customerId })
        finalUsuarioId = found.ok ? found.usuarioId : null
      }
      if (!finalUsuarioId) return jsonResponse(200, { ok: true })

      const statusPagamento = mapStripeSubscriptionToPagamentoStatus(obj.status)
      const venc = toIsoDateFromUnixSeconds(obj.current_period_end)

      await applyUsuarioPaymentUpdate(adminClient, finalUsuarioId, {
        plano: plano ?? undefined,
        limite_funcionarios: plano ? limiteFuncionarios : undefined,
        limite_agendamentos_mes: plano ? limiteAgendamentosMes : undefined,
        status_pagamento: statusPagamento,
        data_vencimento: venc,
        free_trial_consumido: plano ? true : undefined,
        stripe_customer_id: customerId,
        stripe_subscription_id: subscriptionId,
        stripe_last_event_id: eventId,
        stripe_last_event_at: nowIso,
      })

      return jsonResponse(200, { ok: true })
    }

    return jsonResponse(200, { ok: true })
  }

  let payload: Payload
  try {
    payload = (await req.json()) as Payload
  } catch {
    return jsonResponse(400, { error: 'invalid_json' })
  }

  if (payload && (payload as Payload).action === 'create_booking_fee_checkout') {
    if (!bookingFeeEnabled) return jsonResponse(400, { error: 'feature_disabled' })
    const origin = cleanUrl(req.headers.get('origin') ?? '')
    const siteUrl = cleanUrl(Deno.env.get('SITE_URL') ?? Deno.env.get('APP_URL') ?? origin)
    if (!siteUrl) return jsonResponse(500, { error: 'missing_env', message: 'Defina SITE_URL/APP_URL para montar
success_url.' })

    return await handlePublicBookingFeeCheckout({
      adminClient,
      stripeKey,
      siteUrl,
```

```
      payload: payload as Payload & { action: 'create_booking_fee_checkout' },
    })
  }

  if (payload && (payload as Payload).action === 'sync_booking_fee_session') {
    if (!bookingFeeEnabled) return jsonResponse(400, { error: 'feature_disabled' })
    const sessionId = String((payload as Payload & { action: 'sync_booking_fee_session' }).session_id ?? '').trim()
    if (!sessionId) return jsonResponse(400, { error: 'invalid_session_id' })
    const done = await finalizeBookingFeeFromStripeSession({ adminClient, stripeKey, sessionId })
    if (!done.ok) return jsonResponse(400, { error: done.error })
    return jsonResponse(200, { ok: true, agendamento_id: done.agendamentoId })
  }

  const userClient = createClient(supabaseUrl, anonKey, {
    global: {
      headers: {
        Authorization: req.headers.get('Authorization') ?? '',
      },
    },
  })

  const { data: userData, error: userErr } = await userClient.auth.getUser()
  if (userErr || !userData?.user) {
    return jsonResponse(401, { error: 'unauthorized', message: userErr?.message ?? 'invalid_session' })
  }

  const action = payload ? String((payload as Payload).action ?? '').trim() : ''
  if (!action || (action !== 'create_checkout' && action !== 'sync_checkout_session' && action !==
'create_billing_portal')) {
    return jsonResponse(400, { error: 'invalid_action' })
  }

  if (payload.action === 'sync_checkout_session') {
    const sessionId = String(payload.session_id ?? '').trim()
    if (!sessionId) return jsonResponse(400, { error: 'invalid_session_id' })

    const usuarioIdFromPayload = normalizeUuid(payload.usuario_id)

    const callerId = userData.user.id
    const { data: saRow } = await userClient.from('super_admin').select('id').eq('id', callerId).maybeSingle()
    const isSuperAdmin = Boolean(saRow)

    const sessionRes = await stripeApiRequest(`/checkout/sessions/${encodeURIComponent(sessionId)}?
expand[]=subscription`, { method: 'GET', key: stripeKey })
    if (!sessionRes.ok || !sessionRes.body || typeof sessionRes.body !== 'object') {
      return jsonResponse(400, { error: 'stripe_error', message: 'Falha ao consultar checkout no Stripe.', stripe:
sessionRes.body })
    }

    const session = sessionRes.body as Record<string, unknown>
    const metadata = (session.metadata ?? null) as Record<string, unknown> | null
    const usuarioIdFromMeta = normalizeUuid(metadata?.usuario_id)
    const usuarioIdFromClientRef = normalizeUuid(session.client_reference_id)
    const finalUsuarioId = usuarioIdFromPayload ?? usuarioIdFromMeta ?? usuarioIdFromClientRef
    if (!finalUsuarioId) return jsonResponse(400, { error: 'missing_usuario_id' })
    if (!isSuperAdmin && callerId !== finalUsuarioId) return jsonResponse(403, { error: 'forbidden' })

    const { item, plano } = resolvePlanoFromMetadata(metadata)

    const customerId = typeof session.customer === 'string' ? session.customer : null
    const subscriptionRaw = session.subscription
    const subscriptionId =
      typeof subscriptionRaw === 'string'
        ? subscriptionRaw
        : subscriptionRaw && typeof subscriptionRaw === 'object' && typeof (subscriptionRaw as Record<string,
unknown>).id === 'string'
          ? String((subscriptionRaw as Record<string, unknown>).id)
          : null

    let statusPagamento: string | null = null
    let venc: string | null = null
```

```
    const paymentStatus = resolvePaymentStatus(session.payment_status)

    if (plano && subscriptionRaw && typeof subscriptionRaw === 'object') {
      const subObj = subscriptionRaw as Record<string, unknown>
      statusPagamento = mapStripeSubscriptionToPagamentoStatus(subObj.status)
      venc = toIsoDateFromUnixSeconds(subObj.current_period_end)
    } else if (plano && subscriptionId) {
      const subRes = await stripeApiRequest(`/subscriptions/${encodeURIComponent(subscriptionId)}`, { method: 'GET',
key: stripeKey })
      if (subRes.ok && subRes.body && typeof subRes.body === 'object') {
        const sub = subRes.body as Record<string, unknown>
        statusPagamento = mapStripeSubscriptionToPagamentoStatus(sub.status)
        venc = toIsoDateFromUnixSeconds(sub.current_period_end)
      } else {
        statusPagamento = 'ativo'
      }
    } else if (plano && paymentStatus === 'paid') {
      statusPagamento = 'ativo'
      venc = toIsoDatePlusDays(30)
    }

    const nowIso = new Date().toISOString()

    const extraPriceId = (Deno.env.get('STRIPE_PRICE_FUNCIONARIO_ADICIONAL') ?? '').trim()
    const extraProductId = 'prod_Tik80yLbCUqUhZ'
    const limiteFuncionarios = resolveLimiteFuncionariosFromStripe({
      plano,
      metadata,
      subscription: subscriptionRaw && typeof subscriptionRaw === 'object' ? subscriptionRaw : null,
      extraPriceId: extraPriceId || undefined,
      extraProductId: extraProductId || undefined,
    })
    const limiteAgendamentosMes = resolveLimiteAgendamentosMesFromPlano(plano)

    await applyUsuarioPaymentUpdate(adminClient, finalUsuarioId, {
      plano: plano ?? undefined,
      limite_funcionarios: plano ? limiteFuncionarios : undefined,
      limite_agendamentos_mes: plano ? limiteAgendamentosMes : undefined,
      status_pagamento: statusPagamento ?? undefined,
      data_vencimento: venc ?? undefined,
      free_trial_consumido: plano ? true : undefined,
      stripe_customer_id: customerId,
      stripe_subscription_id: subscriptionId,
      stripe_checkout_session_id: sessionId,
      stripe_last_event_id: `sync_session:${sessionId}`,
      stripe_last_event_at: nowIso,
    })

    return jsonResponse(200, { ok: true, usuario_id: finalUsuarioId, item, plano, status_pagamento: statusPagamento,
data_vencimento: venc })
  }

  if (payload.action === 'create_billing_portal') {
    const usuarioId = normalizeUuid(payload.usuario_id)
    if (!usuarioId) return jsonResponse(400, { error: 'invalid_usuario_id' })

    const callerId = userData.user.id
    const { data: saRow } = await userClient.from('super_admin').select('id').eq('id', callerId).maybeSingle()
    const isSuperAdmin = Boolean(saRow)
    if (!isSuperAdmin && callerId !== usuarioId) {
      return jsonResponse(403, { error: 'forbidden' })
    }

    const { data: usuarioRow, error: usuarioErr } = await adminClient
      .from('usuarios')
      .select('stripe_customer_id,email,nome_negocio')
      .eq('id', usuarioId)
      .maybeSingle()
    if (usuarioErr) {
      return jsonResponse(400, { error: 'db_error', message: usuarioErr.message })
    }
```

```
    const stripeMessageFromBody = (body: unknown) => {
      if (!body || typeof body !== 'object') return null
      const obj = body as Record<string, unknown>
      const err = obj.error
      if (!err || typeof err !== 'object') return null
      const msg = (err as Record<string, unknown>).message
      return typeof msg === 'string' && msg.trim() ? msg.trim() : null
    }

    const stripeCustomerNotFound = (body: unknown) => {
      if (!body || typeof body !== 'object') return false
      const obj = body as Record<string, unknown>
      const err = obj.error
      if (!err || typeof err !== 'object') return false
      const e = err as Record<string, unknown>
      const msg = typeof e.message === 'string' ? e.message.trim().toLowerCase() : ''
      const code = typeof e.code === 'string' ? e.code.trim().toLowerCase() : ''
      const param = typeof e.param === 'string' ? e.param.trim().toLowerCase() : ''
      if (msg.includes('no such customer')) return true
      if (code === 'resource_missing' && param === 'customer') return true
      return false
    }

    const nowIso = new Date().toISOString()
    const email = typeof (usuarioRow as Record<string, unknown> | null)?.email === 'string' ? String((usuarioRow as
Record<string, unknown>).email).trim() : ''
    const nomeNegocio =
      typeof (usuarioRow as Record<string, unknown> | null)?.nome_negocio === 'string' ? String((usuarioRow as
Record<string, unknown>).nome_negocio).trim() : ''

    const createStripeCustomer = async () => {
      const customerParams = new URLSearchParams()
      if (email) customerParams.set('email', email)
      if (nomeNegocio) customerParams.set('name', nomeNegocio)
      customerParams.set('metadata[usuario_id]', usuarioId)

      const customerRes = await stripeApiRequest('/customers', { method: 'POST', key: stripeKey, params: customerParams
})
      if (!customerRes.ok || !customerRes.body || typeof customerRes.body !== 'object') {
        return { ok: false as const, status: customerRes.status, body: customerRes.body }
      }
      const id = typeof (customerRes.body as Record<string, unknown>).id === 'string' ? String((customerRes.body as
Record<string, unknown>).id).trim() : ''
      if (!id) {
        return { ok: false as const, status: customerRes.status, body: customerRes.body }
      }

      const persisted = await applyUsuarioPaymentUpdate(adminClient, usuarioId, {
        stripe_customer_id: id,
        stripe_last_event_id: `billing_portal:customer_created:${id}`,
        stripe_last_event_at: nowIso,
      })
      if (!persisted.ok) {
        return { ok: false as const, status: 400, body: { error: 'db_error', message: persisted.error } }
      }

      return { ok: true as const, id }
    }

    let customerId = typeof (usuarioRow as Record<string, unknown> | null)?.stripe_customer_id === 'string' ?
String((usuarioRow as Record<string, unknown>).stripe_customer_id).trim() : ''
    if (!customerId) {
      const created = await createStripeCustomer()
      if (!created.ok) {
        return jsonResponse(400, {
          error: 'stripe_error',
          message: 'Falha ao criar o cliente no Stripe para abrir o portal.',
          stripe_status: created.status,
          stripe_key_mode: stripeKeyMode,
          stripe: created.body,
        })
      }
```

```
      customerId = created.id
    }

    const origin = cleanUrl(req.headers.get('origin') ?? '')
    const siteUrl = cleanUrl(Deno.env.get('SITE_URL') ?? Deno.env.get('APP_URL') ?? origin)
    if (!siteUrl) {
      return jsonResponse(500, { error: 'missing_env', message: 'Defina SITE_URL/APP_URL para montar return_url.' })
    }

    const params = new URLSearchParams()
    params.set('customer', customerId)
    params.set('return_url', `${siteUrl}/pagamento`)

    const portalConfig = cleanEnvValue(Deno.env.get('STRIPE_BILLING_PORTAL_CONFIGURATION') ?? '')
    if (portalConfig) params.set('configuration', portalConfig)

    const portalRes = await stripeApiRequest('/billing_portal/sessions', { method: 'POST', key: stripeKey, params })
    if (!portalRes.ok || !portalRes.body || typeof portalRes.body !== 'object') {
      if (stripeCustomerNotFound(portalRes.body)) {
        const created = await createStripeCustomer()
        if (!created.ok) {
          return jsonResponse(400, {
            error: 'stripe_error',
            message: 'Falha ao recriar o cliente no Stripe para abrir o portal.',
            stripe_status: created.status,
            stripe_key_mode: stripeKeyMode,
            stripe: created.body,
          })
        }

        const retryParams = new URLSearchParams()
        retryParams.set('customer', created.id)
        retryParams.set('return_url', `${siteUrl}/pagamento`)
        if (portalConfig) retryParams.set('configuration', portalConfig)
        const retryRes = await stripeApiRequest('/billing_portal/sessions', { method: 'POST', key: stripeKey, params:
retryParams })
        if (!retryRes.ok || !retryRes.body || typeof retryRes.body !== 'object') {
          const stripeMessage = stripeMessageFromBody(retryRes.body)
          return jsonResponse(400, {
            error: 'stripe_error',
            message: stripeMessage ? `Falha ao abrir o portal do Stripe: ${stripeMessage}` : 'Falha ao abrir o portal do
Stripe.',
            stripe_status: retryRes.status,
            stripe_key_mode: stripeKeyMode,
            stripe: retryRes.body,
          })
        }

        const url = typeof (retryRes.body as Record<string, unknown>).url === 'string' ? String((retryRes.body as
Record<string, unknown>).url) : ''
        if (!url.trim()) {
          return jsonResponse(400, {
            error: 'stripe_error',
            message: 'Stripe não retornou a URL do portal.',
            stripe_status: retryRes.status,
            stripe_key_mode: stripeKeyMode,
            stripe: retryRes.body,
          })
        }
        return jsonResponse(200, { ok: true, url })
      }

      const stripeMessage = stripeMessageFromBody(portalRes.body)
      return jsonResponse(400, {
        error: 'stripe_error',
        message: stripeMessage ? `Falha ao abrir o portal do Stripe: ${stripeMessage}` : 'Falha ao abrir o portal do
Stripe.',
        stripe_status: portalRes.status,
        stripe_key_mode: stripeKeyMode,
        stripe: portalRes.body,
      })
    }
```

```
    const url = typeof (portalRes.body as Record<string, unknown>).url === 'string' ? String((portalRes.body as Record<string, unknown>).url) : ''
    if (!url.trim()) {
      return jsonResponse(400, {
        error: 'stripe_error',
        message: 'Stripe não retornou a URL do portal.',
        stripe_status: portalRes.status,
        stripe_key_mode: stripeKeyMode,
        stripe: portalRes.body,
      })
    }

    return jsonResponse(200, { ok: true, url })
  }

  const usuarioId = normalizeUuid(payload.usuario_id)
  if (!usuarioId) return jsonResponse(400, { error: 'invalid_usuario_id' })

  const item = String(payload.plano ?? '').trim().toLowerCase()
  if (!item || item === 'free') return jsonResponse(400, { error: 'invalid_plano' })

  const callerId = userData.user.id
  const { data: saRow } = await userClient.from('super_admin').select('id').eq('id', callerId).maybeSingle()
  const isSuperAdmin = Boolean(saRow)
  if (!isSuperAdmin && callerId !== usuarioId) {
    return jsonResponse(403, { error: 'forbidden' })
  }

  const planKeys = new Set(['basic', 'pro', 'enterprise'])
  const serviceKeys = new Set(['setup_completo', 'consultoria_hora'])
  if (!planKeys.has(item) && !serviceKeys.has(item)) {
    return jsonResponse(400, { error: 'invalid_plano', message: `Item inválido: ${item}.` })
  }

  const productIdMap: Record<string, string> = {
    basic: 'prod_Tik9tEMnGcTjdq',
    pro: 'prod_Tik8lu4o69znQA',
    enterprise: 'prod_Tik9hxnnoWGI6a',
    consultoria_hora: 'prod_TikBiK2IspRhUo',
    setup_completo: 'prod_TikBQXciLFeI6O',
  }

  const productId = productIdMap[item] ?? null
  if (!productId) {
    return jsonResponse(400, {
      error: 'missing_product',
      message: `Produto não configurado para item ${item}.`,
    })
  }

  const productRes = await stripeApiRequest(`/products/${encodeURIComponent(productId)}`, { method: 'GET', key: stripeKey })
  if (!productRes.ok || !productRes.body || typeof productRes.body !== 'object') {
    const stripeMessage = (() => {
      if (!productRes.body || typeof productRes.body !== 'object') return null
      const body = productRes.body as Record<string, unknown>
      const err = body.error
      if (!err || typeof err !== 'object') return null
      const msg = (err as Record<string, unknown>).message
      return typeof msg === 'string' && msg.trim() ? msg.trim() : null
    })()
    return jsonResponse(400, {
      error: 'stripe_error',
      message: stripeMessage
        ? `Falha ao consultar produto no Stripe (item=${item}, product=${productId}): ${stripeMessage}`
        : `Falha ao consultar produto no Stripe (item=${item}, product=${productId}).`,
      stripe_status: productRes.status,
      stripe_key_mode: stripeKeyMode,
      stripe: productRes.body,
    })
  }
```

```
  const product = productRes.body as Record<string, unknown>
  if (product.active !== true) {
    return jsonResponse(400, { error: 'inactive_product', message: `Produto inativo no Stripe (item=${item},
product=${productId}).` })
  }
  const dp = product.default_price
  const price = typeof dp === 'string' ? dp : dp && typeof dp === 'object' && typeof (dp as Record<string, unknown>).id
=== 'string' ? String((dp as Record<string, unknown>).id) : null
  if (!price) {
    return jsonResponse(400, { error: 'missing_price', message: `Produto sem default_price no Stripe (item=${item},
product=${productId}).` })
  }

  const origin = cleanUrl(req.headers.get('origin') ?? '')
  const siteUrl = cleanUrl(Deno.env.get('SITE_URL') ?? Deno.env.get('APP_URL') ?? origin)
  if (!siteUrl) {
    return jsonResponse(500, { error: 'missing_env', message: 'Defina SITE_URL/APP_URL para montar success_url.' })
  }

  const { data: usuarioRow, error: usuarioErr } = await adminClient
    .from('usuarios')
    .select('id,email')
    .eq('id', usuarioId)
    .maybeSingle()
  if (usuarioErr || !usuarioRow) {
    return jsonResponse(404, { error: 'usuario_not_found' })
  }

  const rawMetodo = (payload.metodo ?? null) as unknown
  const metodo = rawMetodo === 'pix' ? 'pix' : 'card'
  const isPlan = planKeys.has(item)
  const mode = isPlan && metodo !== 'pix' ? 'subscription' : 'payment'

  const includedPro = 4
  const maxPro = 6
  let funcionariosTotal = 1
  if (item === 'pro') {
    const parsed = parsePositiveInt((payload as Record<string, unknown> | null)?.funcionarios_total)
    if (!parsed) {
      funcionariosTotal = includedPro
    } else if (parsed > maxPro) {
      return jsonResponse(400, { error: 'invalid_funcionarios_total', message: 'Para mais de 6 profissionais, use o
plano EMPRESA.' })
    } else {
      funcionariosTotal = clampInt(parsed, includedPro, maxPro)
    }
  }
  const extraQty = item === 'pro' ? Math.max(0, funcionariosTotal - includedPro) : 0

  const successUrl = `${siteUrl}/pagamento?
checkout=success&usuario_id=${encodeURIComponent(usuarioId)}&item=${encodeURIComponent(item)}&plano=${encodeURIComponent
(item)}&session_id={CHECKOUT_SESSION_ID}`
  const cancelUrl = `${siteUrl}/pagamento?
checkout=cancel&usuario_id=${encodeURIComponent(usuarioId)}&item=${encodeURIComponent(item)}&plano=${encodeURIComponent(
item)}`

  const params = new URLSearchParams()
  params.set('mode', mode)
  params.set('success_url', successUrl)
  params.set('cancel_url', cancelUrl)
  params.set('client_reference_id', usuarioId)
  params.set('metadata[usuario_id]', usuarioId)
  params.set('metadata[item]', item)
  params.set('metadata[plano]', item)
  params.set('metadata[metodo]', metodo)
  params.set('metadata[billing_mode]', mode)
  params.set('metadata[created_by]', isSuperAdmin ? 'super_admin' : 'usuario')
  params.set('metadata[funcionarios_total]', String(funcionariosTotal))
  if (mode === 'subscription') {
    params.set('subscription_data[metadata][usuario_id]', usuarioId)
    params.set('subscription_data[metadata][item]', item)
```

```
    params.set('subscription_data[metadata][plano]', item)
    params.set('subscription_data[metadata][metodo]', metodo)
    params.set('subscription_data[metadata][funcionarios_total]', String(funcionariosTotal))
  }

  const validatePriceForProduct = async (
    candidatePriceId: string,
    expectedProductId: string,
    expectedMode: 'subscription' | 'payment'
  ) => {
    const id = (candidatePriceId ?? '').trim()
    if (!id) return false
    const res = await stripeApiRequest(`/prices/${encodeURIComponent(id)}?expand[]=product`, { method: 'GET', key:
stripeKey })
    if (!res.ok || !res.body || typeof res.body !== 'object') return false
    const priceObj = res.body as Record<string, unknown>
    if (priceObj.active !== true) return false
    if (expectedMode === 'subscription' && priceObj.type !== 'recurring') return false
    if (expectedMode === 'payment' && priceObj.type !== 'one_time') return false
    const productRaw = priceObj.product
    const productObj = productRaw && typeof productRaw === 'object' ? (productRaw as Record<string, unknown>) : null
    const pid = typeof productRaw === 'string' ? productRaw : typeof productObj?.id === 'string' ? String(productObj.id)
: ''
    if (!pid || pid !== expectedProductId) return false
    if (!productObj || productObj.active !== true) return false
    return true
  }

  let finalPrice: string | null = null
  const expectedType = mode === 'subscription' ? 'recurring' : 'one_time'

  if (metodo === 'pix' && isPlan) {
    const envKey = `STRIPE_PRICE_${item.toUpperCase()}_PIX`
    const altEnvKey = item === 'enterprise' ? 'STRIPE_PRICE_EMPRESA_PIX' : null
    const fromEnv = ((Deno.env.get(envKey) ?? '').trim() || (altEnvKey ? (Deno.env.get(altEnvKey) ?? '').trim() :
'')).trim()

    if (fromEnv) {
      const ok = await validatePriceForProduct(fromEnv, productId, 'payment')
      if (!ok) {
        return jsonResponse(400, {
          error: 'invalid_price',
          message: `O Price informado em ${envKey} não está ativo, não é one-time, ou não pertence ao produto
${productId}.`,
        })
      }
      finalPrice = fromEnv
    } else {
      const resolved = await resolveActivePriceIdForProduct({
        productId,
        key: stripeKey,
        expectedMode: 'payment',
        currency: 'brl',
      })
      if (resolved) {
        finalPrice = resolved
      } else {
        return jsonResponse(400, {
          error: 'missing_env',
          message:
            item === 'enterprise'
              ? `Não encontrei um Price one-time (BRL) ativo no Stripe para o produto ${productId}. Crie um Price one-
time no Stripe ou configure ${envKey} (ou ${altEnvKey}).`
              : `Não encontrei um Price one-time (BRL) ativo no Stripe para o produto ${productId}. Crie um Price one-
time no Stripe ou configure ${envKey}.`,
        })
      }
    }
  } else {
    const envKey = `STRIPE_PRICE_${item.toUpperCase()}`
    const altEnvKey = item === 'enterprise' ? 'STRIPE_PRICE_EMPRESA' : null
    const fromEnv = ((Deno.env.get(envKey) ?? '').trim() || (altEnvKey ? (Deno.env.get(altEnvKey) ?? '').trim() :
```

```
'')).trim()

  if (fromEnv) {
   const ok = await validatePriceForProduct(fromEnv, productId, mode)
   if (ok) finalPrice = fromEnv
  }

  if (!finalPrice) {
   const defaultOk = await validatePriceForProduct(price, productId, mode)
   if (defaultOk) {
    finalPrice = price
   } else {
    const resolved = await resolveActivePriceIdForProduct({
     productId,
     key: stripeKey,
     expectedMode: mode,
     currency: 'brl',
    })
    if (resolved) {
     finalPrice = resolved
    } else {
     return jsonResponse(400, {
      error: 'invalid_default_price',
      message: `default_price inválido para o modo ${mode} (esperado ${expectedType}) (item=${item},
product=${productId}, price=${price}).`,
     })
    }
   }
  }
 }

 if (!finalPrice) {
  return jsonResponse(400, { error: 'missing_price', message: `Não foi possível resolver um Price para item=${item}.`
})
 }

 params.set('payment_method_types[0]', metodo)
 params.set('line_items[0][price]', finalPrice)
 params.set('line_items[0][quantity]', '1')

 if (item === 'pro' && extraQty > 0) {
  const extraProductId = 'prod_Tik80yLbCUqUhZ'
  const extraPriceEnvKey = mode === 'subscription' ? 'STRIPE_PRICE_FUNCIONARIO_ADICIONAL' :
'STRIPE_PRICE_FUNCIONARIO_ADICIONAL_PIX'
  let extraPrice = (Deno.env.get(extraPriceEnvKey) ?? '').trim()
  if (!extraPrice) {
   const resolved = await resolveActivePriceIdForProduct({
    productId: extraProductId,
    key: stripeKey,
    expectedMode: mode,
    currency: 'brl',
   })
   if (resolved) extraPrice = resolved
  }

  if (!extraPrice) {
   return jsonResponse(400, {
    error: 'missing_env',
    message: `Não encontrei um Price ativo (BRL) para o produto ${extraProductId}. Configure ${extraPriceEnvKey} com
um Price no Stripe.`,
   })
  }

  const ok = await validatePriceForProduct(extraPrice, extraProductId, mode)
  if (!ok) {
   const expectedType = mode === 'subscription' ? 'recurring' : 'one_time'
   return jsonResponse(400, {
    error: 'invalid_price',
    message: `O Price informado em ${extraPriceEnvKey} não está ativo, não é ${expectedType}, ou não pertence ao
produto ${extraProductId}.`,
   })
  }
```

```
    params.set('line_items[1][price]', extraPrice)
    params.set('line_items[1][quantity]', String(extraQty))
  }

  const email = typeof usuarioRow.email === 'string' ? usuarioRow.email.trim() : ''
  if (email) params.set('customer_email', email)

  const res = await stripeRequest(params, stripeKey)
  if (!res.ok) {
    const msg = (() => {
      if (!res.body || typeof res.body !== 'object') return null
      const body = res.body as Record<string, unknown>
      const err = body.error
      if (err && typeof err === 'object') {
        const m = (err as Record<string, unknown>).message
        return typeof m === 'string' ? m : null
      }
      return null
    })()
    return jsonResponse(400, { error: 'stripe_error', message: msg ?? 'Falha ao criar checkout.', stripe: res.body })
  }

  const body = (res.body ?? null) as unknown
  const url = typeof (body as Record<string, unknown> | null)?.url === 'string' ? String((body as Record<string,
unknown>).url) : null
  if (!url) return jsonResponse(400, { error: 'missing_url' })

  return jsonResponse(200, { url })
})
```

smagenda/supabase/functions/resend-domain/config.toml

```
verify_jwt = false
```

smagenda/supabase/functions/resend-domain/index.ts

```
import { createClient } from 'https://esm.sh/@supabase/supabase-js@2.89.0'

type ResendDomainRecord = {
  record?: string
  name?: string
  type?: string
  ttl?: string
  status?: string
  value?: string
  priority?: number
}

type ResendDomain = {
  object?: string
  id?: string
  name?: string
  status?: string
  created_at?: string
  region?: string
  capabilities?: { sending?: string; receiving?: string } | null
  records?: ResendDomainRecord[]
}

type Payload = {
  domain?: string
  domain_id?: string
  action?: 'status' | 'verify' | 'send_test'
  from?: string
  to?: string
  subject?: string
  text?: string
  html?: string
}

function jsonResponse(status: number, body: unknown) {
  return new Response(JSON.stringify(body), {
    status,
    headers: {
      'Content-Type': 'application/json',
      'Access-Control-Allow-Origin': '*',
      'Access-Control-Allow-Headers': 'authorization, x-client-info, apikey, content-type',
      'Access-Control-Allow-Methods': 'POST, OPTIONS',
    },
  })
}

async function dohQuery(name: string, type: string) {
  const url = new URL('https://cloudflare-dns.com/dns-query')
  url.searchParams.set('name', name)
  url.searchParams.set('type', type)
  const res = await fetch(url.toString(), {
    headers: {
      Accept: 'application/dns-json',
    },
  })
  const json = (await res.json().catch(() => null)) as
    | {
        Status?: number
        Answer?: Array<{ name?: string; type?: number; TTL?: number; data?: string }>
      }
    | null
  return { ok: res.ok, status: res.status, json }
}

function normalizeFqdn(name: string, domain: string) {
  const n = (name ?? '').trim()
  const d = (domain ?? '').trim().toLowerCase()
  if (!n || n === '@') return d
  const nn = n.endsWith('.') ? n.slice(0, -1) : n
  const lower = nn.toLowerCase()
```

```
  if (lower === d || lower.endsWith(`.${d}`)) return lower
  return `${lower}.${d}`
}

function stripOuterQuotes(v: string) {
  const s = v.trim()
  if (s.startsWith('"') && s.endsWith('"') && s.length >= 2) return s.slice(1, -1)
  return s
}

function normalizeSpf(value: string) {
  return stripOuterQuotes(value)
    .trim()
    .replace(/\s+/g, ' ')
}

function normalizeDkim(value: string) {
  return stripOuterQuotes(value)
    .trim()
    .replace(/\s+/g, '')
}

function parseMxAnswerData(data: string) {
  const trimmed = data.trim()
  const parts = trimmed.split(/\s+/g)
  const prioRaw = parts[0] ?? ''
  const hostRaw = parts.slice(1).join(' ')
  const prio = Number(prioRaw)
  const host = hostRaw.endsWith('.') ? hostRaw.slice(0, -1) : hostRaw
  return { prio: Number.isFinite(prio) ? prio : null, host: host.toLowerCase() }
}

async function resendRequest(path: string, method: string, apiKey=[REDACTED]
  const res = await fetch(`https://api.resend.com${path}`, {
    method,
    headers: {
      Authorization: `Bearer ${apiKey}`,
      'Content-Type': 'application/json',
    },
    body: body === undefined ? undefined : JSON.stringify(body),
  })
  const json = (await res.json().catch(() => null)) as unknown
  return { ok: res.ok, status: res.status, json }
}

function splitEmailList(raw: string) {
  const cleaned = raw
    .split(/[;,\n]/g)
    .map((s) => s.trim())
    .filter(Boolean)
  return cleaned
}

function extractEmailAddress(raw: string) {
  const s = raw.trim()
  const m = s.match(/<([^>]+)>/)
  return (m?.[1] ?? s).trim()
}

function isLikelyEmail(s: string) {
  const v = s.trim()
  if (!v.includes('@')) return false
  const [local, domain] = v.split('@')
  if (!local || !domain) return false
  return domain.includes('.') && !domain.startsWith('.') && !domain.endsWith('.')
}

Deno.serve(async (req) => {
  if (req.method === 'OPTIONS') return jsonResponse(200, { ok: true })
  if (req.method !== 'POST') return jsonResponse(405, { error: 'method_not_allowed' })

  const supabaseUrl = Deno.env.get('SUPABASE_URL') ?? ''
```

```
  const anonKey = Deno.env.get('SUPABASE_ANON_KEY') ?? ''
  const serviceRoleKey=[REDACTED]'SERVICE_ROLE_KEY') ?? Deno.env.get('SUPABASE_SERVICE_ROLE_KEY') ?? ''
  const resendApiKey=[REDACTED]'RESEND_API_KEY') ?? ''

  if (!supabaseUrl || !anonKey || !serviceRoleKey) return jsonResponse(500, { error: 'missing_env' })
  if (!resendApiKey) return jsonResponse(500, { error: 'missing_resend_api_key' })

  const userClient = createClient(supabaseUrl, anonKey, {
    global: {
      headers: {
        Authorization: req.headers.get('Authorization') ?? '',
      },
    },
  })
  const adminClient = createClient(supabaseUrl, serviceRoleKey)

  const { data: userData, error: userErr } = await userClient.auth.getUser()
  if (userErr || !userData?.user) return jsonResponse(401, { error: 'unauthorized' })

  const uid = userData.user.id
  const { data: saRow, error: saErr } = await adminClient.from('super_admin').select('id').eq('id', uid).maybeSingle()
  if (saErr) {
    const lower = saErr.message.toLowerCase()
    const missingTable = lower.includes('could not find the table') || lower.includes('schema cache')
    if (missingTable) {
      return jsonResponse(400, {
        error: 'schema_incomplete',
        message: 'Tabela public.super_admin não existe. Execute o SQL de políticas (Super Admin) em
/admin/configuracoes.',
      })
    }
    return jsonResponse(403, { error: 'not_allowed', message: saErr.message })
  }
  if (!saRow) return jsonResponse(403, { error: 'not_allowed' })

  let payload: Payload
  try {
    payload = (await req.json()) as Payload
  } catch {
    return jsonResponse(400, { error: 'invalid_json' })
  }

  const action = payload.action ?? 'status'
  const domainName = (payload.domain ?? '').trim().toLowerCase()
  const domainIdInput = (payload.domain_id ?? '').trim()
  if (!domainName && !domainIdInput) return jsonResponse(400, { error: 'missing_domain' })

  const listRes = await resendRequest('/domains', 'GET', resendApiKey)
  if (!listRes.ok) return jsonResponse(502, { error: 'resend_list_failed', details: listRes.json })

  const listJson = listRes.json as { data?: ResendDomain[] } | ResendDomain[] | null
  const domains = Array.isArray(listJson) ? listJson : Array.isArray(listJson?.data) ? listJson.data : []

  const selected = domainIdInput
    ? domains.find((d) => d.id === domainIdInput) ?? null
    : domains.find((d) => (d.name ?? '').toLowerCase() === domainName) ?? null

  if (!selected?.id) {
    return jsonResponse(404, {
      error: 'domain_not_found',
      domains: domains.map((d) => ({ id: d.id ?? null, name: d.name ?? null, status: d.status ?? null })),
    })
  }

  if (action === 'send_test') {
    const fromRaw = (payload.from ?? '').trim()
    const toRaw = (payload.to ?? '').trim()
    const subject = (payload.subject ?? '').trim() || 'Teste de email (Resend) — SMagenda'
    const text = (payload.text ?? '').trim() || `Teste de envio via Resend em ${new Date().toISOString()}`
    const html = (payload.html ?? '').trim() || `<p>${text}</p>`

    if (!fromRaw || !toRaw) return jsonResponse(400, { error: 'missing_email_fields' })
```

```
    const fromEmail = extractEmailAddress(fromRaw)
    if (!isLikelyEmail(fromEmail)) return jsonResponse(400, { error: 'invalid_from' })

    const toList = splitEmailList(toRaw)
    if (toList.length === 0 || toList.some((t) => !isLikelyEmail(extractEmailAddress(t)))) {
      return jsonResponse(400, { error: 'invalid_to' })
    }

    const selectedDomain = (selected.name ?? '').trim().toLowerCase()
    const fromDomain = fromEmail.split('@')[1]?.trim().toLowerCase() ?? ''
    if (selectedDomain && fromDomain && fromDomain !== selectedDomain && !fromDomain.endsWith(`.${selectedDomain}`)) {
      return jsonResponse(400, { error: 'from_domain_mismatch', message: `O domínio do remetente deve ser
${selectedDomain}.` })
    }

    const emailRes = await resendRequest('/emails', 'POST', resendApiKey, {
      from: fromRaw,
      to: toList,
      subject,
      text,
      html,
    })
    if (!emailRes.ok) return jsonResponse(502, { error: 'resend_send_failed', details: emailRes.json })
    return jsonResponse(200, { ok: true, email: emailRes.json })
  }

  if (action === 'verify') {
    const verifyRes = await resendRequest(`/domains/${selected.id}/verify`, 'POST', resendApiKey)
    if (!verifyRes.ok) return jsonResponse(502, { error: 'resend_verify_failed', details: verifyRes.json })
  }

  const getRes = await resendRequest(`/domains/${selected.id}`, 'GET', resendApiKey)
  if (!getRes.ok) return jsonResponse(502, { error: 'resend_get_failed', details: getRes.json })

  const domain = (getRes.json ?? null) as ResendDomain | null
  const records = Array.isArray(domain?.records) ? domain?.records : []

  const nsRes = await dohQuery(domainName || (domain?.name ?? ''), 'NS').catch(() => ({ ok: false, status: 0, json: null
}))
  const nameservers = Array.isArray(nsRes.json?.Answer)
    ? nsRes.json?.Answer.map((a) => (a.data ?? '').trim()).filter(Boolean)
    : []

  const dnsChecks = await Promise.all(
    records.map(async (r) => {
      const name = (r.name ?? '').trim()
      const type = (r.type ?? '').trim().toUpperCase()
      const expectedValue = typeof r.value === 'string' ? r.value : ''
      const expectedPriority = typeof r.priority === 'number' ? r.priority : null
      const fqdn = normalizeFqdn(name, domainName || (domain?.name ?? ''))

      if (!fqdn || (type !== 'TXT' && type !== 'MX')) {
        return {
          record: r.record ?? null,
          name,
          type,
          fqdn: fqdn || null,
          expected: { value: expectedValue || null, priority: expectedPriority },
          dns: { found: false, match: false, values: [] as string[] },
        }
      }

      const query = await dohQuery(fqdn, type).catch(() => ({ ok: false, status: 0, json: null }))
      const answers = Array.isArray(query.json?.Answer) ? query.json?.Answer : []
      const values = answers.map((a) => (a.data ?? '').trim()).filter(Boolean)

      let match = false
      let hint: string | null = null

      if (type === 'TXT') {
        const expectedNorm = expectedValue.toLowerCase().includes('v=spf1') ? normalizeSpf(expectedValue) :
```

```
normalizeDkim(expectedValue)
        const gotNorms = values.map((v) => {
          const raw = stripOuterQuotes(v)
          const asJoined = raw.replace(/"\s+"/g, '')
          return expectedValue.toLowerCase().includes('v=spf1') ? normalizeSpf(asJoined) : normalizeDkim(asJoined)
        })
        match = gotNorms.some((g) => g === expectedNorm)
        if (!match && values.length > 0 && type === 'TXT' && expectedValue.toLowerCase().includes('v=spf1')) {
          const got = gotNorms[0] ?? ''
          if (got.includes('v=spf1') && got.includes('include:amazonses.com') && !got.endsWith('~all') &&
!got.endsWith('-all')) {
            hint = 'Seu SPF parece estar incompleto. Ele precisa terminar com ~all (ou -all) como no Resend.'
          }
        }
      }

      if (type === 'MX') {
        const expectedHost = (expectedValue ?? '').trim().toLowerCase().replace(/\.$/, '')
        const expectedPrio = expectedPriority
        const parsed = values.map((v) => parseMxAnswerData(v))
        match = parsed.some((p) => p.host === expectedHost && (expectedPrio === null || p.prio === expectedPrio))
        const maybeAppended = parsed.find((p) => p.host.endsWith(`.${(domainName || (domain?.name ??
'')).toLowerCase()}`))
        if (!match && maybeAppended) {
          hint = 'Seu provedor pode estar anexando o domínio no MX. Tente salvar o valor com ponto final (ex.: feedback-
smtp.sa-east-1.amazonses.com.).'
        }
      }

      return {
        record: r.record ?? null,
        name,
        type,
        fqdn,
        expected: { value: expectedValue || null, priority: expectedPriority },
        dns: { found: values.length > 0, match, values },
        hint,
        resendStatus: r.status ?? null,
        ttl: r.ttl ?? null,
      }
    })
  )

  return jsonResponse(200, {
    ok: true,
    domain: {
      id: domain?.id ?? selected.id,
      name: domain?.name ?? selected.name ?? null,
      status: domain?.status ?? selected.status ?? null,
      region: domain?.region ?? null,
      capabilities: domain?.capabilities ?? null,
      records,
    },
    dns: {
      checked_at: new Date().toISOString(),
      nameservers,
      records: dnsChecks,
    },
  })
})
```

smagenda/supabase/functions/whatsapp-lembretes/config.toml

```
verify_jwt = false
```

smagenda/supabase/functions/whatsapp-lembretes/index.ts

```
import { createClient } from 'https://esm.sh/@supabase/supabase-js@2.89.0'

type UsuarioRow = {
  id: string
  whatsapp_instance_name: string | null
  slug: string | null
  enviar_confirmacao: boolean | null
  enviar_lembrete: boolean | null
  lembrete_horas_antes: number | null
  mensagem_lembrete: string | null
  mensagem_confirmacao: string | null
  nome_negocio: string | null
  telefone: string | null
  endereco: string | null
}

type BillingUsuarioRow = {
  id: string
  whatsapp_instance_name: string | null
  slug: string | null
  nome_negocio: string | null
  telefone: string | null
  whatsapp_habilitado?: boolean | null
  status_pagamento?: string | null
  data_vencimento?: string | null
  plano?: string | null
  ativo?: boolean | null
}

type SuperAdminConfigRow = {
  id: string
  whatsapp_api_url: string | null
  whatsapp_api_key=[REDACTED]
}

type ServicoRow = { nome: string | null; preco: number | null }

type FuncionarioRow = { nome_completo: string | null; telefone: string | null }

type UnidadeRow = { nome: string | null; endereco: string | null; telefone: string | null }

type AgendamentoRow = {
  id: string
  usuario_id: string
  cliente_nome: string | null
  cliente_telefone: string | null
  data: string
  hora_inicio: string | null
  status: string
  lembrete_enviado: boolean | null
  confirmacao_enviada?: boolean | null
  extras?: unknown | null
  servico: ServicoRow | null
  funcionario?: FuncionarioRow | null
  unidade?: UnidadeRow | null
}

type UsuarioPartialRow = {
  id: string
  whatsapp_instance_name: string | null
  slug: string | null
  enviar_confirmacao: boolean | null
  mensagem_confirmacao: string | null
  nome_negocio: string | null
  telefone: string | null
  endereco: string | null
}

const defaultConfirmacao = `Olá {nome}!\n\nSeu agendamento foi confirmado:\n📅 {data} às {hora}\n✂️ {servico}\n💰{preco}\n\nLocal: {endereco}\n\nNos vemos em breve!\n{nome_negocio}`
```

```
const defaultFuncionarioNovoAgendamento = `Novo agendamento:\n📅 {data} às {hora}\n👤 {cliente_nome}\n✂️ {servico}\n📍{endereco}`

const FN_VERSION = 'whatsapp-lembretes@2025-12-31'

function jsonResponse(status: number, body: unknown) {
  return new Response(JSON.stringify(body), {
    status,
    headers: {
      'Content-Type': 'application/json',
      'Access-Control-Allow-Origin': '*',
      'Access-Control-Allow-Headers': 'authorization, x-client-info, apikey, content-type, x-cron-secret',
      'Access-Control-Allow-Methods': 'POST, OPTIONS',
      'Access-Control-Expose-Headers': 'x-smagenda-fn',
      'x-smagenda-fn': FN_VERSION,
    },
  })
}

function cleanUrl(input: string) {
  const raw = String(input ?? '').trim()
  if (!raw) return ''
  const withProto = (() => {
    if (/^https?:\/\//i.test(raw)) return raw
    if (/^(localhost|127\.0\.0\.1)(:|$)/i.test(raw)) return `http://${raw}`
    return `https://${raw}`
  })()
  return withProto.replace(/\/+$/, '')
}

function sanitizeInstanceName(input: string) {
  const s = input.trim().toLowerCase()
  const normalized = s
    .normalize('NFKD')
    .replace(/[\u0300-\u036f]/g, '')
    .replace(/[^a-z0-9_-]/g, '-')
    .replace(/-+/g, '-')
    .replace(/^-+|-+$/g, '')
  return normalized.slice(0, 50) || 'smagenda'
}

function normalizePlanoLabel(planoRaw: unknown) {
  const raw = typeof planoRaw === 'string' ? planoRaw.trim() : ''
  if (!raw) return ''
  const p = raw.toLowerCase()
  if (p === 'enterprise') return 'EMPRESA'
  if (p === 'pro' || p === 'team') return 'PRO'
  if (p === 'basic') return 'BASIC'
  if (p === 'free') return 'FREE'
  return raw.toUpperCase()
}

function sanitizePhoneDigits(input: string) {
  return String(input ?? '').replace(/\D/g, '')
}

function buildRecipientCandidates(raw: string) {
  const digits = sanitizePhoneDigits(raw)
  if (!digits) return []

  const bases: string[] = [digits]
  if (!digits.startsWith('55') && (digits.length === 10 || digits.length === 11)) {
    bases.push(`55${digits}`)
  }

  const candidates: string[] = []
  for (const b of bases) {
    candidates.push(b)
    candidates.push(`+${b}`)
    candidates.push(`${b}@s.whatsapp.net`)
    candidates.push(`${b}@c.us`)
```

```
  }

  return Array.from(new Set(candidates.map((v) => v.trim()).filter(Boolean)))
}

function maskRecipient(input: string) {
  const digits = String(input ?? '').replace(/\D/g, '')
  if (!digits) return ''
  const last4 = digits.slice(-4)
  return `${'*'.repeat(Math.max(0, digits.length - 4))}${last4}`
}

function extractTextFragments(input: unknown): string[] {
  if (typeof input === 'string') return [input]
  if (!input) return []
  if (Array.isArray(input)) return input.flatMap((v) => extractTextFragments(v))
  if (typeof input !== 'object') return []
  const obj = input as Record<string, unknown>
  const keys = ['message', 'error', 'details', 'response', 'data', 'description']
  const out: string[] = []
  for (const key of keys) {
    if (obj[key] !== undefined) out.push(...extractTextFragments(obj[key]))
  }
  if (out.length) return out
  return Object.values(obj).flatMap((v) => extractTextFragments(v))
}

function unwrapNotFoundLast(details: unknown): unknown {
  if (!details || typeof details !== 'object') return details
  const obj = details as Record<string, unknown>
  if (obj.error === 'not_found' && obj.last !== undefined) return obj.last
  return details
}

function isRecipientFormatError(details: unknown) {
  const unwrapped = unwrapNotFoundLast(details)
  const text = extractTextFragments(unwrapped).join(' | ').toLowerCase()
  if (!text) return false

  if (text.includes('quote command returned error')) return true
  if (text.includes('invalid jid') || text.includes('jid')) return true
  if (text.includes('not a whatsapp user') || text.includes('is not on whatsapp')) return true
  if (text.includes('invalid number') || text.includes('number invalid')) return true
  return false
}

function normalizeEvolutionText(input: string) {
  return String(input ?? '').replace(/\r\n/g, '\n')
}

function stripUnsupportedEvolutionChars(input: string) {
  const s = normalizeEvolutionText(input)
  const withoutSurrogates = s.replace(/[\uD800-\uDFFF]/g, '')
  const withoutVariationSelectors = withoutSurrogates.replace(/[\uFE0E\uFE0F]/g, '')
  return withoutVariationSelectors.replace(/[^\t\n\r\u0020-\u007E\u00A0-\u00FF]/g, '')
}

function buildTextCandidates(input: string) {
  const raw = normalizeEvolutionText(input)
  const stripped = stripUnsupportedEvolutionChars(raw)
  const candidates = raw === stripped ? [raw] : [raw, stripped]
  return Array.from(new Set(candidates.map((v) => v.trim()).filter(Boolean)))
}

async function evolutionSendTextWithFallback(opts: {
  apiUrl: string
  apiKey=[REDACTED]
  instanceName: string
  phoneRaw: string
  text: string
}) {
  const recipients = buildRecipientCandidates(opts.phoneRaw)
```

```
  const texts = buildTextCandidates(opts.text)
  const attempts: Array<{ recipient: string; text_variant: string; status: number; ok: boolean }> = []

  let last: Awaited<ReturnType<typeof evolutionRequestAuto>> | null = null

  for (const recipient of recipients) {
    for (let i = 0; i < texts.length; i += 1) {
      const text = texts[i]!
      const variant = i === 0 ? 'raw' : 'stripped'
      const res = await evolutionRequestAuto({
        baseUrl: opts.apiUrl,
        apiKey=[REDACTED]
        path: `/message/sendText/${opts.instanceName}`,
        method: 'POST',
        body: { number: recipient, text },
      })

      attempts.push({ recipient: maskRecipient(recipient), text_variant: variant, status: res.status, ok: res.ok })
      last = res
      if (res.ok) return { ...res, attempts }

      if (res.status === 401 || res.status === 403 || res.status === 404) {
        return { ...res, attempts }
      }

      if (!isRecipientFormatError(res.body)) {
        return { ...res, attempts }
      }
    }
  }

  return { ...(last ?? { ok: false, status: 502, body: { error: 'evolution_send_failed' }, baseUrlUsed: opts.apiUrl }),
attempts }
}

function formatBRL(value: number) {
  try {
    return new Intl.NumberFormat('pt-BR', { style: 'currency', currency: 'BRL' }).format(value)
  } catch {
    return `R$ ${value.toFixed(2)}`
  }
}

function formatBRDate(isoDate: string) {
  const parts = isoDate.split('-')
  if (parts.length !== 3) return isoDate
  return `${parts[2]}/${parts[1]}/${parts[0]}`
}

function interpolateTemplate(template: string, vars: Record<string, string>) {
  return template.replace(/\{([a-zA-Z0-9_]+)\}/g, (_, key: string) => vars[key] ?? `{${key}}`)
}

function readExtrasEndereco(extras: unknown) {
  if (!extras || typeof extras !== 'object') return ''
  const v = (extras as Record<string, unknown>).endereco
  if (typeof v !== 'string') return ''
  const t = v.trim()
  return t ? t : ''
}

function computeDaysLeft(todayIso: string, vencIso: string) {
  const today = new Date(`${todayIso}T00:00:00Z`)
  const venc = new Date(`${vencIso}T00:00:00Z`)
  if (!Number.isFinite(today.getTime()) || !Number.isFinite(venc.getTime())) return null
  const diffMs = venc.getTime() - today.getTime()
  return Math.round(diffMs / (24 * 60 * 60 * 1000))
}

async function evolutionRequest(opts: { baseUrl: string; apiKey=[REDACTED]
  const rawKey = String(opts.apiKey ?? '').trim()
  const normalizedKey = rawKey.replace(/^['"`\s]+|['"`\s]+$/g, '')
```

```
  const keyCandidates = Array.from(new Set([normalizedKey, rawKey].map((v) => v.trim()).filter(Boolean)))
  const contentType = opts.body === undefined ? {} : { 'Content-Type': 'application/json' }

  const baseUrl = `${cleanUrl(opts.baseUrl)}${opts.path.startsWith('/') ? '' : '/'}${opts.path}`
  const redactUrl = (input: string) => {
    try {
      const u = new URL(input)
      const sensitive = ['apikey', 'apiKey', 'api_key', 'token', 'access_token', 'key']
      for (const k of sensitive) {
        if (u.searchParams.has(k)) u.searchParams.set(k, 'REDACTED')
      }
      return u.toString()
    } catch {
      return input.replace(/(apikey|apiKey|api_key|token|access_token|key)=([^&]+)/g, '$1=REDACTED')
    }
  }

  const appendQuery = (input: string, param: string, value: string) => {
    try {
      const u = new URL(input)
      u.searchParams.set(param, value)
      return u.toString()
    } catch {
      const sep = input.includes('?') ? '&' : '?'
      return `${input}${sep}${encodeURIComponent(param)}=${encodeURIComponent(value)}`
    }
  }

  const urlVariants: Array<{ url: string; urlRedacted: string }> = [{ url: baseUrl, urlRedacted: redactUrl(baseUrl) }]
  for (const key of keyCandidates) {
    const candidates = [appendQuery(baseUrl, 'apikey', key), appendQuery(baseUrl, 'apiKey', key), appendQuery(baseUrl,
'token', key)]
    for (const u of candidates) {
      const redacted = redactUrl(u)
      if (urlVariants.some((v) => v.urlRedacted === redacted)) continue
      urlVariants.push({ url: u, urlRedacted: redacted })
    }
  }

  const headerVariants: Array<Record<string, string>> = keyCandidates.length
    ? keyCandidates.flatMap((key) => [
        { apikey=[REDACTED]
        { apiKey=[REDACTED]
        { 'x-api-key': key, ...contentType },
        { Authorization: `Bearer ${key}`, ...contentType },
        { Authorization: key, ...contentType },
      ])
    : [{ ...contentType }]

  const doFetchOnce = async (args: { url: string; urlRedacted: string; headers: Record<string, string> }) => {
    const controller = new AbortController()
    const timeout = setTimeout(() => controller.abort(), 15000)
    try {
      const res = await fetch(args.url, {
        method: opts.method,
        headers: args.headers,
        body: opts.body === undefined ? undefined : JSON.stringify(opts.body),
        signal: controller.signal,
      })
      const text = await res.text()
      let json: unknown = null
      try {
        json = text ? JSON.parse(text) : null
      } catch {
        json = text
      }
      return { ok: res.ok, status: res.status, body: json }
    } catch (e: unknown) {
      const message = e instanceof Error ? e.message : 'Falha ao conectar na Evolution API'
      return { ok: false, status: 502, body: { error: 'evolution_fetch_failed', message, url: args.urlRedacted } }
    } finally {
      clearTimeout(timeout)
```

```
      }
    }

    const tryAll = async () => {
      let last: Awaited<ReturnType<typeof doFetchOnce>> | null = null
      for (const u of urlVariants) {
        for (const h of headerVariants) {
          last = await doFetchOnce({ url: u.url, urlRedacted: u.urlRedacted, headers: h })
          if (last.status !== 401) return last
        }
      }
      return last ?? doFetchOnce({ url: baseUrl, urlRedacted: redactUrl(baseUrl), headers: headerVariants[0]! })
    }

    const first = await tryAll()
    if (!first.ok && first.status === 502) {
      const second = await tryAll()
      if (second.ok || second.status !== 502) return second
    }
    return first
  }

  function buildEvolutionBaseUrls(rawBaseUrl: string) {
    const baseUrl = cleanUrl(rawBaseUrl)
    const lower = baseUrl.toLowerCase()
    if (lower.includes('/v1/') || lower.endsWith('/v1') || lower.includes('/v2/') || lower.endsWith('/v2')) {
      return [baseUrl]
    }
    return [baseUrl, `${baseUrl}/v2`, `${baseUrl}/v1`]
  }

  async function evolutionRequestAuto(opts: { baseUrl: string; apiKey=[REDACTED]
    const candidates = buildEvolutionBaseUrls(opts.baseUrl)
    let last: { ok: boolean; status: number; body: unknown; baseUrlUsed: string } | null = null
    for (const baseUrl of candidates) {
      const res = await evolutionRequest({ ...opts, baseUrl })
      last = { ...res, baseUrlUsed: baseUrl }
      if (res.status === 404) continue
      return last
    }
    return last ?? { ok: false, status: 404, body: null, baseUrlUsed: candidates[0] ?? opts.baseUrl }
  }

  function toVirtualTimeZoneDate(date: Date, timeZone: string) {
    const parts = new Intl.DateTimeFormat('en-CA', {
      timeZone,
      year: 'numeric',
      month: '2-digit',
      day: '2-digit',
      hour: '2-digit',
      minute: '2-digit',
      second: '2-digit',
      hour12: false,
    }).formatToParts(date)
    const map: Record<string, string> = {}
    for (const p of parts) {
      if (p.type !== 'literal') map[p.type] = p.value
    }
    const iso = `${map.year}-${map.month}-${map.day}T${map.hour}:${map.minute}:${map.second}Z`
    return new Date(iso)
  }

  Deno.serve(async (req) => {
    if (req.method === 'OPTIONS') return jsonResponse(200, { ok: true })
    if (req.method !== 'POST') return jsonResponse(405, { error: 'method_not_allowed' })

    let bodyJson: unknown = null
    try {
      bodyJson = await req.json()
    } catch {
      bodyJson = null
    }
```

```
  const requiredSecret=[REDACTED]'CRON_SECRET') ?? ''
  if (requiredSecret) {
    const provided = req.headers.get('x-cron-secret') ?? ''
    if (provided !== requiredSecret) return jsonResponse(401, { error: 'unauthorized' })
  }

  const supabaseUrl = Deno.env.get('SUPABASE_URL') ?? ''
  const serviceRoleKey=[REDACTED]'SERVICE_ROLE_KEY') ?? Deno.env.get('SUPABASE_SERVICE_ROLE_KEY') ?? ''
  if (!supabaseUrl || !serviceRoleKey) return jsonResponse(500, { error: 'missing_env' })

  const adminClient = createClient(supabaseUrl, serviceRoleKey)

  const isMissingColumnError = (message: string) => message.toLowerCase().includes('does not exist') &&
message.toLowerCase().includes('column')
  const isMissingTableError = (message: string) => message.includes('schema cache') || message.includes("Could not find
the table 'public.")
  const isMissingRelationshipError = (message: string) => message.toLowerCase().includes('could not find a relationship
between')

  const { data: saRaw, error: saErr } = await adminClient
    .from('super_admin')
    .select('id,whatsapp_api_url,whatsapp_api_key')
    .not('whatsapp_api_url', 'is', null)
    .not('whatsapp_api_key', 'is', null)
    .limit(1)
    .maybeSingle()

  if (saErr) {
    if (isMissingColumnError(saErr.message) || isMissingTableError(saErr.message)) {
      return jsonResponse(400, { error: 'schema_incomplete', message: saErr.message })
    }
    return jsonResponse(400, { error: 'load_super_admin_failed', message: saErr.message })
  }

  const sa = (saRaw ?? null) as unknown as SuperAdminConfigRow | null
  const globalApiUrl = sa?.whatsapp_api_url ?? null
  const globalApiKey=[REDACTED]
  if (!globalApiUrl || !globalApiKey) return jsonResponse(200, { ok: true, skipped_all: 'not_configured', results: [] })

  const action = (() => {
    if (!bodyJson || typeof bodyJson !== 'object') return null
    const raw = (bodyJson as Record<string, unknown>).action
    return typeof raw === 'string' ? raw.trim().toLowerCase() : null
  })()

  if (action === 'billing_status_changed') {
    const usuarioId = (() => {
      if (!bodyJson || typeof bodyJson !== 'object') return null
      const raw = (bodyJson as Record<string, unknown>).usuario_id
      if (typeof raw !== 'string') return null
      const v = raw.trim()
      return v ? v : null
    })()
    const status = (() => {
      if (!bodyJson || typeof bodyJson !== 'object') return null
      const raw = (bodyJson as Record<string, unknown>).status_pagamento
      return typeof raw === 'string' ? raw.trim().toLowerCase() : null
    })()

    if (!usuarioId || !status) return jsonResponse(400, { error: 'invalid_payload' })

    const { data: uRaw, error: uErr } = await adminClient
      .from('usuarios')

.select('id,whatsapp_instance_name,slug,nome_negocio,telefone,whatsapp_habilitado,status_pagamento,data_vencimento,plano
,ativo')
      .eq('id', usuarioId)
      .maybeSingle()
    if (uErr) return jsonResponse(400, { error: 'load_user_failed', message: uErr.message })
    if (!uRaw) return jsonResponse(404, { error: 'usuario_not_found' })
```

```
    const u = uRaw as unknown as BillingUsuarioRow
    if (u.ativo === false && status !== 'suspenso' && status !== 'cancelado') return jsonResponse(200, { ok: true,
skipped: 'usuario_inativo' })
    if (u.whatsapp_habilitado !== true) return jsonResponse(200, { ok: true, skipped: 'whatsapp_disabled' })
    if (!sanitizePhoneDigits(u.telefone ?? '')) return jsonResponse(200, { ok: true, skipped: 'missing_phone' })

    const instanceNameRaw = u.whatsapp_instance_name ?? u.slug ?? ''
    const instanceName = sanitizeInstanceName(instanceNameRaw)

    const venc = (u.data_vencimento ?? '').trim()
    const vencPart = venc ? `\n\nVencimento: ${formatBRDate(venc)}` : ''
    const plano = normalizePlanoLabel(u.plano)
    const planoPart = plano ? `\nPlano: ${plano}` : ''
    const business = (u.nome_negocio ?? '').trim()
    const header = business ? `Olá! Aqui é o SMagenda (${business}).` : 'Olá! Aqui é o SMagenda.'

    const text =
      status === 'inadimplente'
        ? `${header}\n\nIdentificamos falha no pagamento da sua assinatura.${vencPart}${planoPart}\n\nPara regularizar e
manter o acesso, finalize o pagamento em: ${cleanUrl(Deno.env.get('SITE_URL') ?? Deno.env.get('APP_URL') ??
'')}/pagamento`
        : status === 'suspenso'
          ? `${header}\n\nSeu acesso está suspenso no momento.${vencPart}${planoPart}\n\nRegularize o pagamento para
reativar em: ${cleanUrl(Deno.env.get('SITE_URL') ?? Deno.env.get('APP_URL') ?? '')}/pagamento`
          : status === 'cancelado'
            ? `${header}\n\nSua assinatura foi cancelada.${planoPart}\n\nCaso queira reativar, acesse:
${cleanUrl(Deno.env.get('SITE_URL') ?? Deno.env.get('APP_URL') ?? '')}/pagamento`
            : `${header}\n\nAtualização do pagamento: ${status}.${vencPart}${planoPart}`

    const sendRes = await evolutionSendTextWithFallback({ apiUrl: globalApiUrl, apiKey=[REDACTED]'', text })
    if (!sendRes.ok) return jsonResponse(502, { ok: false, error: 'evolution_error', details: sendRes.body, attempts:
sendRes.attempts })
    return jsonResponse(200, { ok: true, sent: true, usuario_id: u.id, status_pagamento: status })
  }

  if (action === 'billing_daily') {
    const timeZone = 'America/Sao_Paulo'
    const now = new Date()
    const nowVirtual = toVirtualTimeZoneDate(now, timeZone)
    const todayIso = nowVirtual.toISOString().slice(0, 10)

    const { data: usuarios, error: usuariosErr } = await adminClient
      .from('usuarios')

.select('id,whatsapp_instance_name,slug,nome_negocio,telefone,whatsapp_habilitado,status_pagamento,data_vencimento,plano
,ativo')
      .eq('ativo', true)
      .not('data_vencimento', 'is', null)
      .in('status_pagamento', ['ativo', 'trial', 'inadimplente', 'suspenso'])
      .limit(2000)
    if (usuariosErr) return jsonResponse(400, { error: 'load_users_failed', message: usuariosErr.message })

    const updated: Array<{ usuario_id: string; next_status: string; next_ativo: boolean | null }> = []
    const sent: Array<{ usuario_id: string; kind: string; days_left: number }> = []
    const skipped: Array<{ usuario_id: string; reason: string }> = []

    for (const row of usuarios ?? []) {
      const u = row as unknown as BillingUsuarioRow
      const venc = (u.data_vencimento ?? '').trim()
      if (!venc) {
        skipped.push({ usuario_id: u.id, reason: 'missing_vencimento' })
        continue
      }

      const daysLeft = computeDaysLeft(todayIso, venc)
      if (daysLeft === null) {
        skipped.push({ usuario_id: u.id, reason: 'invalid_vencimento' })
        continue
      }

      const currentStatus = String(u.status_pagamento ?? '').trim().toLowerCase()
      const baseUrl = cleanUrl(Deno.env.get('SITE_URL') ?? Deno.env.get('APP_URL') ?? '')
```

```
      const pagamentoUrl = baseUrl ? `${baseUrl}/pagamento` : '/pagamento'

      const stage = (() => {
        if (daysLeft === 3 || daysLeft === 1 || daysLeft === 0) return { kind: currentStatus === 'trial' ? 'trial' :
'vencendo' }
        if (daysLeft === -3) return { kind: 'atraso_3' }
        if (daysLeft === -7) return { kind: 'atraso_7' }
        if (daysLeft === -14) return { kind: 'suspensao_14' }
        if (daysLeft === -30) return { kind: 'cancelamento_30' }
        return null
      })()

      const shouldUpdateTrialExpired = currentStatus === 'trial' && daysLeft < 0
      if (shouldUpdateTrialExpired) {
        const { error: updErr } = await adminClient
          .from('usuarios')
          .update({ status_pagamento: 'inadimplente' })
          .eq('id', u.id)
          .eq('status_pagamento', 'trial')
        if (!updErr) updated.push({ usuario_id: u.id, next_status: 'inadimplente', next_ativo: null })
      }

      if (stage?.kind === 'atraso_3' && currentStatus === 'ativo') {
        const { error: updErr } = await adminClient
          .from('usuarios')
          .update({ status_pagamento: 'inadimplente' })
          .eq('id', u.id)
          .eq('status_pagamento', 'ativo')
        if (!updErr) updated.push({ usuario_id: u.id, next_status: 'inadimplente', next_ativo: null })
      }

      if (stage?.kind === 'suspensao_14') {
        const { error: updErr } = await adminClient
          .from('usuarios')
          .update({ status_pagamento: 'suspenso', ativo: false })
          .eq('id', u.id)
          .neq('status_pagamento', 'cancelado')
        if (!updErr) updated.push({ usuario_id: u.id, next_status: 'suspenso', next_ativo: false })

        await adminClient.from('funcionarios').update({ ativo: false }).eq('usuario_master_id', u.id)
      }

      if (stage?.kind === 'cancelamento_30') {
        const { error: updErr } = await adminClient
          .from('usuarios')
          .update({ status_pagamento: 'cancelado', ativo: false })
          .eq('id', u.id)
        if (!updErr) updated.push({ usuario_id: u.id, next_status: 'cancelado', next_ativo: false })

        await adminClient.from('funcionarios').update({ ativo: false }).eq('usuario_master_id', u.id)
      }

      if (!stage) {
        skipped.push({ usuario_id: u.id, reason: 'not_due_window' })
        continue
      }

      if (u.whatsapp_habilitado !== true) {
        skipped.push({ usuario_id: u.id, reason: 'whatsapp_disabled' })
        continue
      }

      const phone = u.telefone ?? ''
      if (!sanitizePhoneDigits(phone)) {
        skipped.push({ usuario_id: u.id, reason: 'invalid_phone' })
        continue
      }

      const instanceNameRaw = u.whatsapp_instance_name ?? u.slug ?? ''
      const instanceName = sanitizeInstanceName(instanceNameRaw)
      const plano = normalizePlanoLabel(u.plano)
      const planoPart = plano ? `\nPlano: ${plano}` : ''
```

```
      const business = (u.nome_negocio ?? '').trim()
      const header = business ? `Olá! Aqui é o SMagenda (${business}).` : 'Olá! Aqui é o SMagenda.'
      const vencPart = venc ? `\nVencimento: ${formatBRDate(venc)}` : ''

      const text = (() => {
        if (stage.kind === 'trial') {
          const when = daysLeft === 0 ? 'hoje' : daysLeft === 1 ? 'amanhã' : 'em 3 dias'
          return `${header}\n\nSeu período de teste termina ${when}.${vencPart}${planoPart}\n\nPara manter o acesso,
escolha um plano em: ${pagamentoUrl}`
        }
        if (stage.kind === 'vencendo') {
          const when = daysLeft === 0 ? 'hoje' : daysLeft === 1 ? 'amanhã' : 'em 3 dias'
          return `${header}\n\nSeu pagamento vence ${when}.${vencPart}${planoPart}\n\nPara evitar interrupções, confira
em: ${pagamentoUrl}`
        }
        if (stage.kind === 'atraso_3') {
          return `${header}\n\nSeu pagamento está em atraso há 3 dias.${vencPart}${planoPart}\n\nPara regularizar e
manter o acesso, acesse: ${pagamentoUrl}`
        }
        if (stage.kind === 'atraso_7') {
          return `${header}\n\nSeu pagamento está em atraso há 7 dias.${vencPart}${planoPart}\n\nPara evitar suspensão,
acesse: ${pagamentoUrl}`
        }
        if (stage.kind === 'suspensao_14') {
          return `${header}\n\nSeu acesso foi suspenso por pagamento em atraso.${vencPart}${planoPart}\n\nRegularize
para reativar: ${pagamentoUrl}`
        }
        return `${header}\n\nSua assinatura foi cancelada por pagamento em atraso.${planoPart}\n\nCaso queira reativar,
acesse: ${pagamentoUrl}`
      })()

      const sendRes = await evolutionSendTextWithFallback({ apiUrl: globalApiUrl, apiKey=[REDACTED]
      if (!sendRes.ok) {
        skipped.push({ usuario_id: u.id, reason: 'evolution_error' })
        continue
      }

      sent.push({ usuario_id: u.id, kind: stage.kind, days_left: daysLeft })
    }

    return jsonResponse(200, { ok: true, action: 'billing_daily', today: todayIso, updated, sent, skipped })
  }

  if (action === 'funcionario_novo_agendamento') {
    const agendamentoId = (() => {
      if (!bodyJson || typeof bodyJson !== 'object') return null
      const raw = (bodyJson as Record<string, unknown>).agendamento_id
      if (typeof raw !== 'string') return null
      const id = raw.trim()
      return id ? id : null
    })()

    if (!agendamentoId) return jsonResponse(400, { error: 'invalid_payload' })

    const agSelFull =

'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,hora_fim,status,extras,servico:servicos(nome,preco),funcio
nario:funcionarios(nome_completo,telefone),unidade:unidades(nome,endereco,telefone)'
    const agSelBase =
'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,hora_fim,status,extras,servico:servicos(nome,preco)'

    const agFirst = await adminClient.from('agendamentos').select(agSelFull).eq('id', agendamentoId).maybeSingle()
    const agRes =
      agFirst.error && (isMissingTableError(agFirst.error.message) || isMissingRelationshipError(agFirst.error.message)
|| isMissingColumnError(agFirst.error.message))
        ? await adminClient.from('agendamentos').select(agSelBase).eq('id', agendamentoId).maybeSingle()
        : null

    const agData = (agRes ? agRes.data : agFirst.data) as unknown as AgendamentoRow | null
    const agErr = agRes ? agRes.error : agFirst.error
    if (agErr) {
      if (isMissingColumnError(agErr.message) || isMissingTableError(agErr.message)) return jsonResponse(400, { error:
```

```
'schema_incomplete', message: agErr.message })
      return jsonResponse(400, { error: 'load_agendamento_failed', message: agErr.message })
    }
    if (!agData) return jsonResponse(404, { error: 'agendamento_not_found' })

    const status = String(agData.status ?? '').trim().toLowerCase()
    if (status === 'cancelado') return jsonResponse(200, { ok: true, skipped: 'cancelado' })

    const funcionarioTelefone = (agData.funcionario?.telefone ?? '').trim()
    if (!sanitizePhoneDigits(funcionarioTelefone)) return jsonResponse(200, { ok: true, skipped:
'missing_funcionario_phone' })

    const userSel = 'id,whatsapp_instance_name,slug,nome_negocio,telefone,endereco'
    const userRes = await adminClient.from('usuarios').select(userSel).eq('id', agData.usuario_id).maybeSingle()
    if (userRes.error) {
      if (isMissingColumnError(userRes.error.message) || isMissingTableError(userRes.error.message)) {
        return jsonResponse(400, { error: 'schema_incomplete', message: userRes.error.message })
      }
      return jsonResponse(400, { error: 'load_user_failed', message: userRes.error.message })
    }
    if (!userRes.data) return jsonResponse(404, { error: 'usuario_not_found' })

    const uRow = userRes.data as unknown as UsuarioPartialRow
    const instanceNameRaw = uRow.whatsapp_instance_name ?? uRow.slug ?? ''
    const instanceName = sanitizeInstanceName(instanceNameRaw)

    const clienteEndereco = readExtrasEndereco(agData.extras)
    const unidadeEndereco = (agData.unidade?.endereco ?? '').trim()
    const endereco = clienteEndereco || unidadeEndereco || (uRow.endereco ?? '')

    const vars = {
      data: formatBRDate(agData.data),
      hora: String(agData.hora_inicio ?? ''),
      cliente_nome: String(agData.cliente_nome ?? ''),
      servico: String(agData.servico?.nome ?? ''),
      endereco,
      nome_negocio: String(uRow.nome_negocio ?? ''),
      cliente_telefone: String(agData.cliente_telefone ?? ''),
      profissional_nome: String(agData.funcionario?.nome_completo ?? ''),
    }

    const text = interpolateTemplate(defaultFuncionarioNovoAgendamento, vars).trim()
    if (!text) return jsonResponse(200, { ok: true, skipped: 'missing_message_data' })

    const sendRes = await evolutionSendTextWithFallback({
      apiUrl: globalApiUrl,
      apiKey=[REDACTED]
      instanceName,
      phoneRaw: funcionarioTelefone,
      text,
    })

    if (!sendRes.ok) return jsonResponse(502, { ok: false, error: 'evolution_error', details: sendRes.body, attempts:
sendRes.attempts })
    return jsonResponse(200, { ok: true, sent: true, usuario_id: uRow.id, agendamento_id: agendamentoId })
  }

  const agendamentoId = (() => {
    if (!bodyJson || typeof bodyJson !== 'object') return null
    const raw = (bodyJson as Record<string, unknown>).agendamento_id
    if (typeof raw !== 'string') return null
    const id = raw.trim()
    return id ? id : null
  })()

  if (agendamentoId) {
    let canUpdateConfirmacao = true
    const agSelFull =

'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,confirmacao_enviada,extras,servico:servicos(nome,pr
eco),funcionario:funcionarios(nome_completo,telefone),unidade:unidades(nome,endereco,telefone)'
    const agSelFullNoFlag =
```

```
'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,extras,servico:servicos(nome,preco),funcionario:fun
cionarios(nome_completo,telefone),unidade:unidades(nome,endereco,telefone)'
    const agSelBase =
'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,confirmacao_enviada,extras,servico:servicos(nome,pr
eco)'
    const agSelBaseNoFlag =
'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,extras,servico:servicos(nome,preco)'

    const first = await adminClient.from('agendamentos').select(agSelFull).eq('id', agendamentoId).maybeSingle()
    const agData = await (async () => {
      if (!first.error) return first

      const msg = first.error.message
      const missingFlag = isMissingColumnError(msg)
      const missingJoin = isMissingTableError(msg) || isMissingRelationshipError(msg)

      if (missingFlag) {
        canUpdateConfirmacao = false
        const second = await adminClient.from('agendamentos').select(agSelFullNoFlag).eq('id',
agendamentoId).maybeSingle()
        if (!second.error) return second
        if (isMissingTableError(second.error.message) || isMissingRelationshipError(second.error.message) ||
isMissingColumnError(second.error.message)) {
          return await adminClient.from('agendamentos').select(agSelBaseNoFlag).eq('id', agendamentoId).maybeSingle()
        }
        return second
      }

      if (missingJoin || isMissingColumnError(msg)) {
        const second = await adminClient.from('agendamentos').select(agSelBase).eq('id', agendamentoId).maybeSingle()
        if (!second.error) return second
        if (isMissingColumnError(second.error.message)) {
          canUpdateConfirmacao = false
          return await adminClient.from('agendamentos').select(agSelBaseNoFlag).eq('id', agendamentoId).maybeSingle()
        }
        return second
      }

      return first
    })()

    if (agData.error) {
      return jsonResponse(400, { error: 'load_agendamento_failed', message: agData.error.message })
    }
    if (!agData.data) return jsonResponse(404, { error: 'agendamento_not_found' })

    const agRow = agData.data as unknown as AgendamentoRow
    if (agRow.status !== 'confirmado') return jsonResponse(200, { ok: true, skipped: 'not_confirmed' })
    if (agRow.confirmacao_enviada === true) return jsonResponse(200, { ok: true, skipped: 'already_sent' })

    const userSel =
'id,whatsapp_instance_name,slug,enviar_confirmacao,mensagem_confirmacao,nome_negocio,telefone,endereco'
    const userRes = await adminClient.from('usuarios').select(userSel).eq('id', agRow.usuario_id).maybeSingle()
    if (userRes.error) {
      if (isMissingColumnError(userRes.error.message) || isMissingTableError(userRes.error.message)) {
        return jsonResponse(400, { error: 'schema_incomplete', message: userRes.error.message })
      }
      return jsonResponse(400, { error: 'load_user_failed', message: userRes.error.message })
    }
    if (!userRes.data) return jsonResponse(404, { error: 'usuario_not_found' })

    const uRow = userRes.data as unknown as UsuarioPartialRow
    if (uRow.enviar_confirmacao === false) return jsonResponse(200, { ok: true, skipped: 'disabled' })

    const instanceNameRaw = uRow.whatsapp_instance_name ?? uRow.slug ?? ''
    const instanceName = sanitizeInstanceName(instanceNameRaw)

    const tmpl = (uRow.mensagem_confirmacao ?? '').trim() ? (uRow.mensagem_confirmacao ?? '') : defaultConfirmacao

    const clienteEndereco = readExtrasEndereco(agRow.extras)
    const unidadeEndereco = (agRow.unidade?.endereco ?? '').trim()
```

```
    const endereco = clienteEndereco || unidadeEndereco || (uRow.endereco ?? '')
    const telefoneProfissional = (agRow.funcionario?.telefone ?? '').trim() || (uRow.telefone ?? '')

    const vars = {
      nome: agRow.cliente_nome ?? '',
      data: formatBRDate(agRow.data),
      hora: agRow.hora_inicio ?? '',
      servico: agRow.servico?.nome ?? '',
      preco: agRow.servico?.preco != null ? formatBRL(Number(agRow.servico.preco)) : '',
      cliente_endereco: clienteEndereco,
      endereco,
      nome_negocio: uRow.nome_negocio ?? '',
      telefone_profissional: telefoneProfissional,
      profissional_nome: agRow.funcionario?.nome_completo ?? '',
      unidade_nome: agRow.unidade?.nome ?? '',
      unidade_endereco: unidadeEndereco,
      unidade_telefone: agRow.unidade?.telefone ?? '',
    }
    const text = interpolateTemplate(tmpl, vars).trim()
    const phoneRaw = agRow.cliente_telefone ?? ''
    if (!text || !sanitizePhoneDigits(phoneRaw)) return jsonResponse(200, { ok: true, skipped: 'missing_message_data' })

    const sendRes = await evolutionSendTextWithFallback({
      apiUrl: globalApiUrl,
      apiKey=[REDACTED]
      instanceName,
      phoneRaw,
      text,
    })

    if (!sendRes.ok) return jsonResponse(502, { ok: false, error: 'evolution_error', details: sendRes.body, attempts:
sendRes.attempts })

    if (canUpdateConfirmacao) {
      const { error: updErr } = await adminClient
        .from('agendamentos')
        .update({ confirmacao_enviada: true, confirmacao_enviada_em: new Date().toISOString() })
        .eq('id', agRow.id)
        .eq('usuario_id', uRow.id)
        .eq('confirmacao_enviada', false)

      if (updErr) {
        return jsonResponse(502, { ok: false, error: 'save_confirmacao_flag_failed', message: updErr.message })
      }
    }

    return jsonResponse(200, { ok: true, sent: true, usuario_id: uRow.id, agendamento_id: agRow.id })
  }

  const { data: users, error: usersErr } = await adminClient
    .from('usuarios')

.select('id,whatsapp_instance_name,slug,enviar_confirmacao,enviar_lembrete,lembrete_horas_antes,mensagem_confirmacao,men
sagem_lembrete,nome_negocio,telefone,endereco')
    .or('enviar_lembrete.eq.true,enviar_confirmacao.eq.true')
    .limit(500)

  if (usersErr) {
    if (isMissingColumnError(usersErr.message)) return jsonResponse(400, { error: 'schema_incomplete', message:
usersErr.message })
    return jsonResponse(400, { error: 'load_users_failed', message: usersErr.message })
  }

  const timeZone = 'America/Sao_Paulo'
  const now = new Date()
  const nowVirtual = toVirtualTimeZoneDate(now, timeZone)
  const results: Array<{
    usuario_id: string
    confirm_sent: number
    confirm_skipped: number
    confirm_failed: number
    reminder_sent: number
```

```
    reminder_skipped: number
    reminder_failed: number
  }> = []

  for (const u of users ?? []) {
    const uRow = u as unknown as UsuarioRow

    const instanceNameRaw = uRow.whatsapp_instance_name ?? uRow.slug ?? ''
    const instanceName = sanitizeInstanceName(instanceNameRaw)
    let confirmSent = 0
    let confirmSkipped = 0
    let confirmFailed = 0

    let reminderSent = 0
    let reminderSkipped = 0
    let reminderFailed = 0

    const shouldConfirm = uRow.enviar_confirmacao !== false
    if (shouldConfirm) {
      const confirmStart = new Date(nowVirtual)
      confirmStart.setDate(confirmStart.getDate() - 7)
      const confirmEnd = new Date(nowVirtual)
      confirmEnd.setDate(confirmEnd.getDate() + 90)

      const confirmStartDate = confirmStart.toISOString().slice(0, 10)
      const confirmEndDate = confirmEnd.toISOString().slice(0, 10)

      const confirmSelFull =

'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,confirmacao_enviada,extras,servico:servicos(nome,preco),funcionario:funcionarios(nome_completo,telefone),unidade:unidades(nome,endereco,telefone)'
      const confirmSelBase =
'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,confirmacao_enviada,extras,servico:servicos(nome,preco)'

      const confirmFirst = await adminClient
        .from('agendamentos')
        .select(confirmSelFull)
        .eq('usuario_id', uRow.id)
        .eq('status', 'confirmado')
        .eq('confirmacao_enviada', false)
        .gte('data', confirmStartDate)
        .lte('data', confirmEndDate)
        .limit(200)

      const confirmSecond =
        confirmFirst.error &&
        (isMissingTableError(confirmFirst.error.message) || isMissingRelationshipError(confirmFirst.error.message) ||
isMissingColumnError(confirmFirst.error.message))
          ? await adminClient
              .from('agendamentos')
              .select(confirmSelBase)
              .eq('usuario_id', uRow.id)
              .eq('status', 'confirmado')
              .eq('confirmacao_enviada', false)
              .gte('data', confirmStartDate)
              .lte('data', confirmEndDate)
              .limit(200)
          : null

      const confirmAgs = (confirmSecond ? confirmSecond.data : confirmFirst.data) as unknown as AgendamentoRow[] | null
      const confirmErr = confirmSecond ? confirmSecond.error : confirmFirst.error

      if (confirmErr) {
        confirmFailed++
      } else {
        for (const ag of confirmAgs ?? []) {
          const agRow = ag as unknown as AgendamentoRow
          if (agRow.confirmacao_enviada === true) {
            confirmSkipped++
            continue
          }
```

```
          const tmpl = (uRow.mensagem_confirmacao ?? '').trim() ? (uRow.mensagem_confirmacao ?? '') : defaultConfirmacao

          const clienteEndereco = readExtrasEndereco(agRow.extras)
          const unidadeEndereco = (agRow.unidade?.endereco ?? '').trim()
          const endereco = clienteEndereco || unidadeEndereco || (uRow.endereco ?? '')
          const telefoneProfissional = (agRow.funcionario?.telefone ?? '').trim() || (uRow.telefone ?? '')

          const vars = {
            nome: agRow.cliente_nome ?? '',
            data: formatBRDate(agRow.data),
            hora: agRow.hora_inicio ?? '',
            servico: agRow.servico?.nome ?? '',
            preco: agRow.servico?.preco != null ? formatBRL(Number(agRow.servico.preco)) : '',
            cliente_endereco: clienteEndereco,
            endereco,
            nome_negocio: uRow.nome_negocio ?? '',
            telefone_profissional: telefoneProfissional,
            profissional_nome: agRow.funcionario?.nome_completo ?? '',
            unidade_nome: agRow.unidade?.nome ?? '',
            unidade_endereco: unidadeEndereco,
            unidade_telefone: agRow.unidade?.telefone ?? '',
          }
          const text = interpolateTemplate(tmpl, vars).trim()
          const phoneRaw = agRow.cliente_telefone ?? ''
          if (!text || !sanitizePhoneDigits(phoneRaw)) {
            confirmSkipped++
            continue
          }

          const sendRes = await evolutionSendTextWithFallback({
            apiUrl: globalApiUrl,
            apiKey=[REDACTED]
            instanceName,
            phoneRaw,
            text,
          })

          if (!sendRes.ok) {
            confirmFailed++
            continue
          }

          const { error: updErr } = await adminClient
            .from('agendamentos')
            .update({ confirmacao_enviada: true, confirmacao_enviada_em: new Date().toISOString() })
            .eq('id', agRow.id)
            .eq('usuario_id', uRow.id)
            .eq('confirmacao_enviada', false)

          if (updErr) {
            confirmFailed++
            continue
          }

          confirmSent++
        }
      }
    }

    if (uRow.enviar_lembrete === true) {
      const hours = Number(uRow.lembrete_horas_antes ?? 24)
      const hoursSafe = Number.isFinite(hours) && hours >= 1 && hours <= 168 ? hours : 24

      const targetStart = new Date(nowVirtual.getTime() + hoursSafe * 60 * 60 * 1000)
      const targetEnd = new Date(targetStart.getTime() + 60 * 60 * 1000)

      const startDate = targetStart.toISOString().slice(0, 10)
      const endDate = targetEnd.toISOString().slice(0, 10)

      const reminderSelFull =
```

```
'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,lembrete_enviado,extras,servico:servicos(nome,preco
),funcionario:funcionarios(nome_completo,telefone),unidade:unidades(nome,endereco,telefone)'
      const reminderSelBase =
'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,lembrete_enviado,extras,servico:servicos(nome,preco
)'

      const reminderFirst = await adminClient
        .from('agendamentos')
        .select(reminderSelFull)
        .eq('usuario_id', uRow.id)
        .eq('status', 'confirmado')
        .eq('lembrete_enviado', false)
        .gte('data', startDate)
        .lte('data', endDate)
        .limit(200)

      const reminderSecond =
        reminderFirst.error &&
        (isMissingTableError(reminderFirst.error.message) || isMissingRelationshipError(reminderFirst.error.message) ||
isMissingColumnError(reminderFirst.error.message))
          ? await adminClient
              .from('agendamentos')
              .select(reminderSelBase)
              .eq('usuario_id', uRow.id)
              .eq('status', 'confirmado')
              .eq('lembrete_enviado', false)
              .gte('data', startDate)
              .lte('data', endDate)
              .limit(200)
          : null

      const ags = (reminderSecond ? reminderSecond.data : reminderFirst.data) as unknown as AgendamentoRow[] | null
      const agErr = reminderSecond ? reminderSecond.error : reminderFirst.error

      if (agErr) {
        reminderFailed++
      } else {
        for (const ag of ags ?? []) {
          const agRow = ag as unknown as AgendamentoRow

          if (agRow.lembrete_enviado === true) {
            reminderSkipped++
            continue
          }

          const data = agRow.data
          const horaInicio = agRow.hora_inicio ?? ''
          const startVirtual = data && horaInicio ? new Date(`${data}T${horaInicio}:00Z`) : null
          if (!startVirtual || !(startVirtual >= targetStart && startVirtual < targetEnd)) {
            reminderSkipped++
            continue
          }

          const tmpl = uRow.mensagem_lembrete ?? ''

          const clienteEndereco = readExtrasEndereco(agRow.extras)
          const unidadeEndereco = (agRow.unidade?.endereco ?? '').trim()
          const endereco = clienteEndereco || unidadeEndereco || (uRow.endereco ?? '')
          const telefoneProfissional = (agRow.funcionario?.telefone ?? '').trim() || (uRow.telefone ?? '')

          const vars = {
            nome: agRow.cliente_nome ?? '',
            data: formatBRDate(agRow.data),
            hora: agRow.hora_inicio ?? '',
            servico: agRow.servico?.nome ?? '',
            preco: agRow.servico?.preco != null ? formatBRL(Number(agRow.servico.preco)) : '',
            cliente_endereco: clienteEndereco,
            endereco,
            nome_negocio: uRow.nome_negocio ?? '',
            telefone_profissional: telefoneProfissional,
            profissional_nome: agRow.funcionario?.nome_completo ?? '',
            unidade_nome: agRow.unidade?.nome ?? '',
```

```
            unidade_endereco: unidadeEndereco,
            unidade_telefone: agRow.unidade?.telefone ?? '',
          }
          const text = interpolateTemplate(tmpl, vars).trim()
          const phoneRaw = agRow.cliente_telefone ?? ''
          if (!text || !sanitizePhoneDigits(phoneRaw)) {
            reminderSkipped++
            continue
          }

          const sendRes = await evolutionSendTextWithFallback({
            apiUrl: globalApiUrl,
            apiKey=[REDACTED]
            instanceName,
            phoneRaw,
            text,
          })

          if (!sendRes.ok) {
            reminderFailed++
            continue
          }

          const { error: updErr } = await adminClient
            .from('agendamentos')
            .update({ lembrete_enviado: true, lembrete_enviado_em: new Date().toISOString() })
            .eq('id', agRow.id)
            .eq('usuario_id', uRow.id)
            .eq('lembrete_enviado', false)

          if (updErr) {
            reminderFailed++
            continue
          }

          reminderSent++
        }
      }
    }

    results.push({
      usuario_id: uRow.id,
      confirm_sent: confirmSent,
      confirm_skipped: confirmSkipped,
      confirm_failed: confirmFailed,
      reminder_sent: reminderSent,
      reminder_skipped: reminderSkipped,
      reminder_failed: reminderFailed,
    })
  }

  return jsonResponse(200, { ok: true, results })
})
```

smagenda/supabase/functions/whatsapp/config.toml

```
verify_jwt = false
```

smagenda/supabase/functions/whatsapp/index.ts

```
import { createClient } from 'https://esm.sh/@supabase/supabase-js@2.89.0'

type Payload =
  | { action: 'connect'; number?: string | null }
  | { action: 'status' }
  | { action: 'disconnect' }
  | { action: 'send_test'; number: string; text: string }
  | { action: 'send_confirmacao'; agendamento_id: string }
  | { action: 'send_cancelamento'; agendamento_id: string }
  | { action: 'config_status' }
  | { action: 'admin_diagnostics' }
  | { action: 'admin_status' }
  | { action: 'admin_connect'; number?: string | null }
  | { action: 'admin_disconnect' }
  | { action: 'admin_send_aviso'; cliente_ids: string[]; text: string }

type UsuarioAuthRow = { id: string; tipo_conta: string | null }

type FuncionarioAuthRow = { id: string; usuario_master_id: string | null }

type SuperAdminAuthRow = { id: string }

type SuperAdminConfigRow = {
  id: string
  whatsapp_api_url: string | null
  whatsapp_api_key=[REDACTED]
  whatsapp_instance_name: string | null
}

type ClienteAvisoRow = { id: string; telefone: string | null; nome_negocio: string | null }

type UsuarioBaseRow = {
  id: string
  slug: string | null
  nome_negocio: string | null
  telefone: string | null
  endereco: string | null
  whatsapp_api_url: string | null
  whatsapp_api_key=[REDACTED]
  whatsapp_habilitado?: boolean | null
}

type UsuarioExtraRow = {
  whatsapp_instance_name: string | null
  enviar_confirmacao: boolean | null
  enviar_lembrete: boolean | null
  enviar_cancelamento?: boolean | null
  lembrete_horas_antes: number | null
  mensagem_confirmacao: string | null
  mensagem_lembrete: string | null
  mensagem_cancelamento?: string | null
}

type ServicoRow = { nome: string | null; preco: number | null }

type FuncionarioRow = { nome_completo: string | null; telefone: string | null }

type UnidadeRow = { nome: string | null; endereco: string | null; telefone: string | null }

type AgendamentoRow = {
  id: string
  usuario_id: string
  cliente_nome: string | null
  cliente_telefone: string | null
  data: string
  hora_inicio: string | null
  status: string
  funcionario_id?: string | null
  unidade_id?: string | null
  extras?: unknown | null
```

```
  servico: ServicoRow | null
  funcionario?: FuncionarioRow | null
  unidade?: UnidadeRow | null
}

type AgendamentoConfirmacaoRow = { confirmacao_enviada: boolean | null }

const defaultConfirmacao = `Olá {nome}!\n\nSeu agendamento foi confirmado:\n📅 {data} às {hora}\n✂️ {servico}\n💰
{preco}\n\nLocal: {endereco}\n\nNos vemos em breve!\n{nome_negocio}`

const defaultCancelamento = `Olá {nome}!\n\nSeu agendamento foi cancelado:\n📅 {data} às {hora}\n✂️ {servico}\n\nSe
precisar remarcar, é só me chamar.\n{nome_negocio}`

const FN_VERSION = 'whatsapp@2026-01-14.1'

function jsonResponse(status: number, body: unknown) {
  const withFn = (() => {
    if (!body || typeof body !== 'object') return body
    if (Array.isArray(body)) return body
    const obj = body as Record<string, unknown>
    if (typeof obj.fn === 'string' && obj.fn.trim()) return body
    return { ...obj, fn: FN_VERSION }
  })()

  return new Response(JSON.stringify(withFn), {
    status,
    headers: {
      'Content-Type': 'application/json',
      'Access-Control-Allow-Origin': '*',
      'Access-Control-Allow-Headers': 'authorization, x-user-jwt, x-client-info, apikey, content-type',
      'Access-Control-Allow-Methods': 'POST, OPTIONS',
      'Access-Control-Expose-Headers': 'x-smagenda-fn',
      'x-smagenda-fn': FN_VERSION,
    },
  })
}

function extractJwt(req: Request) {
  const xJwt = req.headers.get('x-user-jwt') ?? ''
  if (xJwt.trim()) return xJwt.trim()
  const auth = req.headers.get('Authorization') ?? ''
  const m = auth.match(/^Bearer\s+(.+)$/i)
  if (m?.[1]) return m[1].trim()
  return ''
}

function decodeJwtPayload(jwt: string): Record<string, unknown> | null {
  const parts = jwt.split('.')
  if (parts.length !== 3) return null
  const payload = parts[1] ?? ''
  if (!payload) return null
  const b64 = payload.replace(/-/g, '+').replace(/_/g, '/')
  const padded = b64 + '='.repeat((4 - (b64.length % 4)) % 4)
  try {
    const json = atob(padded)
    const parsed = JSON.parse(json) as unknown
    if (!parsed || typeof parsed !== 'object') return null
    return parsed as Record<string, unknown>
  } catch {
    return null
  }
}

function validateJwtShapeAndProject(jwt: string, supabaseUrl: string) {
  const payload = decodeJwtPayload(jwt)
  if (!payload) return { ok: false as const, reason: 'malformed_jwt' as const }

  const exp = payload.exp
  if (typeof exp === 'number') {
    const now = Math.floor(Date.now() / 1000)
    if (exp <= now) return { ok: false as const, reason: 'expired_jwt' as const }
  }
```

```
  const iss = payload.iss
  if (typeof iss === 'string' && supabaseUrl) {
    const expectedPrefix = `${supabaseUrl.replace(/\/+$/, '')}/auth/v1`
    if (!iss.startsWith(expectedPrefix)) {
      return { ok: false as const, reason: 'jwt_project_mismatch' as const, iss, expectedPrefix }
    }
  }

  return { ok: true as const }
}

function isMissingTableError(message: string) {
  return message.includes('schema cache') || message.includes("Could not find the table 'public.")
}

function isMissingColumnError(message: string) {
  return message.toLowerCase().includes('does not exist') && message.toLowerCase().includes('column')
}

function isMissingRelationshipError(message: string) {
  const m = message.toLowerCase()
  return m.includes('could not find a relationship between')
}

async function loadGlobalWhatsappConfig(dbClient: ReturnType<typeof createClient>) {
  const { data, error } = await dbClient
    .from('super_admin')
    .select('id,whatsapp_api_url,whatsapp_api_key')
    .not('whatsapp_api_url', 'is', null)
    .not('whatsapp_api_key', 'is', null)
    .limit(1)
    .maybeSingle()

  if (error) {
    const msg = String(error.message ?? '')
    const lower = msg.toLowerCase()
    const rls = lower.includes('row level security') || lower.includes('permission denied')
    if (rls) {
      return { ok: false as const, error: 'permission_denied' as const, message: msg }
    }
    if (isMissingTableError(error.message) || isMissingColumnError(error.message)) {
      return { ok: false as const, error: 'schema_incomplete' as const, message: error.message }
    }
    return { ok: false as const, error: 'load_failed' as const, message: error.message }
  }

  const row = (data ?? null) as unknown as { whatsapp_api_url?: string | null; whatsapp_api_key?: string | null } | null
  const apiUrl = typeof row?.whatsapp_api_url === 'string' ? row.whatsapp_api_url : null
  const apiKey=[REDACTED]'string' ? row.whatsapp_api_key=[REDACTED]
  const hasConfig = Boolean(apiUrl && apiKey)

  if (!hasConfig) return { ok: true as const, configured: false as const, apiUrl: null, apiKey=[REDACTED]
  return { ok: true as const, configured: true as const, apiUrl: apiUrl!, apiKey=[REDACTED]
}

function cleanUrl(input: string) {
  const raw = String(input ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
  if (!raw) return ''
  const withProto = (() => {
    if (/^https?:\/\//i.test(raw)) return raw
    if (/^(localhost|127\.0\.0\.1)(:|$)/i.test(raw)) return `http://${raw}`
    return `https://${raw}`
  })()
  return withProto.replace(/\/+$/, '')
}

function isForbiddenEvolutionHost(hostname: string) {
  const h = hostname.trim().toLowerCase()
  if (!h) return true
```

```
  if (h === 'localhost' || h === '0.0.0.0' || h === '127.0.0.1') return true
  if (h.endsWith('.local')) return true
  if (/^10\./.test(h)) return true
  if (/^192\.168\./.test(h)) return true
  const m172 = h.match(/^172\.(\d+)\./)
  if (m172) {
    const n = Number(m172[1])
    if (!Number.isNaN(n) && n >= 16 && n <= 31) return true
  }
  return false
}

function validateEvolutionBaseUrl(raw: string) {
  const cleaned = cleanUrl(raw)
  if (!cleaned) return { ok: false as const, reason: 'missing' as const }
  let u: URL
  try {
    u = new URL(cleaned)
  } catch {
    return { ok: false as const, reason: 'invalid' as const, cleaned }
  }
  if (u.protocol !== 'http:' && u.protocol !== 'https:') return { ok: false as const, reason: 'invalid_protocol' as
const, cleaned }
  if (isForbiddenEvolutionHost(u.hostname)) return { ok: false as const, reason: 'local_address' as const, cleaned }
  return { ok: true as const, cleaned }
}

function sanitizeInstanceName(input: string) {
  const s = input.trim().toLowerCase()
  const normalized = s
    .normalize('NFKD')
    .replace(/[\u0300-\u036f]/g, '')
    .replace(/[^a-z0-9_-]/g, '-')
    .replace(/-+/g, '-')
    .replace(/^-+|-+$/g, '')
  return normalized.slice(0, 50) || 'smagenda'
}

function sanitizePhone(input: string) {
  return String(input ?? '').replace(/\D/g, '')
}

function buildRecipientCandidates(raw: string) {
  const digits = sanitizePhone(raw)
  if (!digits) return []

  const bases: string[] = [digits]
  if (!digits.startsWith('55') && (digits.length === 10 || digits.length === 11)) {
    bases.push(`55${digits}`)
  }

  const candidates: string[] = []
  for (const b of bases) {
    candidates.push(b)
    candidates.push(`+${b}`)
    candidates.push(`${b}@s.whatsapp.net`)
    candidates.push(`${b}@c.us`)
  }

  return Array.from(new Set(candidates.map((v) => v.trim()).filter(Boolean)))
}

function maskRecipient(input: string) {
  const digits = String(input ?? '').replace(/\D/g, '')
  if (!digits) return ''
  const last4 = digits.slice(-4)
  return `${'*'.repeat(Math.max(0, digits.length - 4))}${last4}`
}

function isRecipientFormatError(details: unknown) {
  const unwrapped = unwrapNotFoundLast(details)
  const text = extractTextFragments(unwrapped).join(' | ').toLowerCase()
```

```
  const hasExistsFalse = (input: unknown): boolean => {
    if (!input || typeof input !== 'object') return false
    if (Array.isArray(input)) return input.some((v) => hasExistsFalse(v))
    const obj = input as Record<string, unknown>
    if (obj.exists === false) return true
    return Object.values(obj).some((v) => hasExistsFalse(v))
  }

  if (hasExistsFalse(unwrapped)) return true
  if (!text) return false

  if (text.includes('quote command returned error')) return true
  if (text.includes('invalid jid') || text.includes('jid')) return true
  if (text.includes('not a whatsapp user') || text.includes('is not on whatsapp')) return true
  if (text.includes('invalid number') || text.includes('number invalid')) return true
  return false
}

function normalizeEvolutionText(input: string) {
  return String(input ?? '').replace(/\r\n/g, '\n')
}

function stripUnsupportedEvolutionChars(input: string) {
  const s = normalizeEvolutionText(input)
  const withoutSurrogates = s.replace(/[\uD800-\uDFFF]/g, '')
  const withoutVariationSelectors = withoutSurrogates.replace(/[\uFE0E\uFE0F]/g, '')
  return withoutVariationSelectors.replace(/[^\t\n\r\u0020-\u007E\u00A0-\u00FF]/g, '')
}

function buildTextCandidates(input: string) {
  const raw = normalizeEvolutionText(input)
  const stripped = stripUnsupportedEvolutionChars(raw)
  const candidates = raw === stripped ? [raw] : [raw, stripped]
  return Array.from(new Set(candidates.map((v) => v.trim()).filter(Boolean)))
}

async function evolutionSendTextWithFallback(opts: {
  apiUrl: string
  apiKey=[REDACTED]
  instanceName: string
  phoneRaw: string
  text: string
  stopOn404?: (args: { body: unknown; url: string; baseUrlUsed: string }) => boolean
}) {
  const recipients = buildRecipientCandidates(opts.phoneRaw)
  const texts = buildTextCandidates(opts.text)
  const attempts: Array<{ recipient: string; text_variant: string; status: number; ok: boolean }> = []

  let last: Awaited<ReturnType<typeof evolutionRequestAuto>> | null = null

  for (const recipient of recipients) {
    for (let i = 0; i < texts.length; i += 1) {
      const text = texts[i]!
      const variant = i === 0 ? 'raw' : 'stripped'
      const res = await evolutionRequestAuto({
        baseUrl: opts.apiUrl,
        apiKey=[REDACTED]
        path: `/message/sendText/${opts.instanceName}`,
        method: 'POST',
        body: { number: recipient, text },
        stopOn404: opts.stopOn404,
      })

      attempts.push({ recipient: maskRecipient(recipient), text_variant: variant, status: res.status, ok: res.ok })
      last = res
      if (res.ok) return { ...res, attempts }

      if (res.status === 401 || res.status === 403 || res.status === 404) {
        return { ...res, attempts }
      }
```

```
      if (!isRecipientFormatError(res.body)) {
        return { ...res, attempts }
      }
    }
  }

  return { ...(last ?? { ok: false, status: 502, body: { error: 'evolution_send_failed' } }), attempts }
}

function formatBRL(value: number) {
  try {
    return new Intl.NumberFormat('pt-BR', { style: 'currency', currency: 'BRL' }).format(value)
  } catch {
    return `R$ ${value.toFixed(2)}`
  }
}

function formatBRDate(isoDate: string) {
  const parts = isoDate.split('-')
  if (parts.length !== 3) return isoDate
  return `${parts[2]}/${parts[1]}/${parts[0]}`
}

function readExtrasEndereco(extras: unknown) {
  if (!extras || typeof extras !== 'object') return ''
  const v = (extras as Record<string, unknown>).endereco
  if (typeof v !== 'string') return ''
  const t = v.trim()
  return t ? t : ''
}

function readExtrasEmail(extras: unknown) {
  if (!extras || typeof extras !== 'object') return ''
  const v = (extras as Record<string, unknown>).email
  if (typeof v !== 'string') return ''
  const t = v.trim().toLowerCase()
  if (!t) return ''
  const ok = /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(t)
  return ok ? t : ''
}

function interpolateTemplate(template: string, vars: Record<string, string>) {
  return template.replace(/\{([a-zA-Z0-9_]+)\}/g, (_, key: string) => vars[key] ?? `{${key}}`)
}

async function resendSendEmail(args: {
  apiKey=[REDACTED]
  from: string
  to: string
  subject: string
  text: string
}) {
  const res = await fetch('https://api.resend.com/emails', {
    method: 'POST',
    headers: {
      Authorization: `Bearer ${args.apiKey}`,
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ from: args.from, to: args.to, subject: args.subject, text: args.text }),
  })
  const body = (await res.json().catch(() => null)) as unknown
  return { ok: res.ok, status: res.status, body }
}

async function evolutionRequest(opts: { baseUrl: string; apiKey=[REDACTED]
  const rawKey = String(opts.apiKey ?? '').trim()
  const normalizedKey = rawKey.replace(/^['"`\s]+|['"`\s]+$/g, '')
  const keyCandidates = Array.from(new Set([normalizedKey, rawKey].map((v) => v.trim()).filter(Boolean)))
  const contentType = opts.body === undefined ? {} : { 'Content-Type': 'application/json' }

  const baseUrl = `${cleanUrl(opts.baseUrl)}${opts.path.startsWith('/') ? '' : '/'}${opts.path}`
  const redactUrl = (input: string) => {
```

```
    try {
      const u = new URL(input)
      const sensitive = ['apikey', 'apiKey', 'api_key', 'token', 'access_token', 'key']
      for (const k of sensitive) {
        if (u.searchParams.has(k)) u.searchParams.set(k, 'REDACTED')
      }
      return u.toString()
    } catch {
      return input.replace(/(apikey|apiKey|api_key|token|access_token|key)=([^&]+)/g, '$1=REDACTED')
    }
  }

  const appendQuery = (input: string, param: string, value: string) => {
    try {
      const u = new URL(input)
      u.searchParams.set(param, value)
      return u.toString()
    } catch {
      const sep = input.includes('?') ? '&' : '?'
      return `${input}${sep}${encodeURIComponent(param)}=${encodeURIComponent(value)}`
    }
  }

  const urlVariants: Array<{ kind: string; url: string; urlRedacted: string }> = [{ kind: 'path', url: baseUrl,
urlRedacted: redactUrl(baseUrl) }]
  for (const key of keyCandidates) {
    const candidates: Array<{ kind: string; url: string }> = [
      { kind: `query_apikey`, url: appendQuery(baseUrl, 'apikey', key) },
      { kind: `query_apiKey`, url: appendQuery(baseUrl, 'apiKey', key) },
      { kind: `query_token`, url: appendQuery(baseUrl, 'token', key) },
    ]
    for (const c of candidates) {
      const redacted = redactUrl(c.url)
      if (urlVariants.some((v) => v.urlRedacted === redacted)) continue
      urlVariants.push({ kind: c.kind, url: c.url, urlRedacted: redacted })
    }
  }

  const headerVariants: Array<{ kind: string; headers: Record<string, string> }> = keyCandidates.length
    ? keyCandidates.flatMap((key, i) => {
        const suffix = keyCandidates.length > 1 ? `#${i + 1}` : ''
        return [
          { kind: `apikey${suffix}`, headers: { apikey=[REDACTED]
          { kind: `apiKey${suffix}`, headers: { apiKey=[REDACTED]
          { kind: `x-api-key${suffix}`, headers: { 'x-api-key': key, ...contentType } },
          { kind: `x-api_key${suffix}`, headers: { 'x-api_key': key, ...contentType } },
          { kind: `authorization_bearer${suffix}`, headers: { Authorization: `Bearer ${key}`, ...contentType } },
          { kind: `authorization_raw${suffix}`, headers: { Authorization: key, ...contentType } },
          { kind: `token${suffix}`, headers: { token: key, ...contentType } },
          { kind: `x-access-token${suffix}`, headers: { 'x-access-token': key, ...contentType } },
        ]
      })
    : [{ kind: 'no_key', headers: { ...contentType } }]

  const doFetchOnce = async (args: { url: string; urlRedacted: string; headers: Record<string, string> }) => {
    const controller = new AbortController()
    const timeout = setTimeout(() => controller.abort(), 20000)
    try {
      const res = await fetch(args.url, {
        method: opts.method,
        headers: args.headers,
        body: opts.body === undefined ? undefined : JSON.stringify(opts.body),
        signal: controller.signal,
      })
      const text = await res.text()
      let json: unknown = null
      try {
        json = text ? JSON.parse(text) : null
      } catch {
        json = text
      }
      return { ok: res.ok, status: res.status, body: json, url: args.urlRedacted }
```

```
    } catch (e: unknown) {
      const errAny = e as { name?: unknown; message?: unknown }
      const name = typeof errAny.name === 'string' ? errAny.name : ''
      const message = typeof errAny.message === 'string' ? errAny.message : e instanceof Error ? e.message : 'Falha ao
conectar na Evolution API'
      const lower = message.toLowerCase()
      const isAbort = name === 'AbortError' || lower.includes('aborted') || lower.includes('abort')
      return {
        ok: false,
        status: isAbort ? 504 : 502,
        body: { error: isAbort ? 'evolution_timeout' : 'evolution_fetch_failed', message, url: args.urlRedacted },
        url: args.urlRedacted,
      }
    } finally {
      clearTimeout(timeout)
    }
  }

  const tryAllVariants = async () => {
    const attempts: Array<{ url_kind: string; header_kind: string; status: number }> = []
    let last: Awaited<ReturnType<typeof doFetchOnce>> | null = null
    for (const u of urlVariants) {
      for (const h of headerVariants) {
        const res = await doFetchOnce({ url: u.url, urlRedacted: u.urlRedacted, headers: h.headers })
        attempts.push({ url_kind: u.kind, header_kind: h.kind, status: res.status })
        last = res
        if (res.status !== 401) return res
      }
    }

    if (last && last.status === 401) {
      return {
        ...last,
        body: {
          error: 'evolution_unauthorized',
          url: last.url,
          attempts,
          response: last.body,
        },
      }
    }
    return last ?? doFetchOnce({ url: baseUrl, urlRedacted: redactUrl(baseUrl), headers: headerVariants[0]!.headers })
  }

  return tryAllVariants()
}

function buildEvolutionBaseUrls(rawBaseUrl: string) {
  const baseUrl = cleanUrl(rawBaseUrl)
  const lower = baseUrl.toLowerCase()

  const isIpHostname = (() => {
    try {
      const u = new URL(baseUrl)
      const h = u.hostname.trim()
      if (!h) return false
      return /^\d{1,3}(\.\d{1,3}){3}$/.test(h) || h.includes(':')
    } catch {
      return false
    }
  })()

  const withoutPort = (() => {
    try {
      const u = new URL(baseUrl)
      const port = u.port
      if (!port) return null
      const isDefault = (u.protocol === 'http:' && port === '80') || (u.protocol === 'https:' && port === '443')
      if (isDefault) return null
      u.port = ''
      return cleanUrl(u.toString())
    } catch {
```

```
      return null
    }
  })()

  const altProtocol = (() => {
    if (lower.startsWith('https://')) return `http://${baseUrl.slice('https://'.length)}`
    if (lower.startsWith('http://')) return `https://${baseUrl.slice('http://'.length)}`
    return null
  })()

  const altProtocolWithoutPort = (() => {
    if (!withoutPort) return null
    const l = withoutPort.toLowerCase()
    if (l.startsWith('https://')) return `http://${withoutPort.slice('https://'.length)}`
    if (l.startsWith('http://')) return `https://${withoutPort.slice('http://'.length)}`
    return null
  })()

  const roots = (() => {
    const arr = [baseUrl, altProtocol, withoutPort, altProtocolWithoutPort].filter(Boolean) as string[]
    try {
      const u = new URL(baseUrl)
      const port = u.port
      const isNonDefault = Boolean(port && !((u.protocol === 'http:' && port === '80') || (u.protocol === 'https:' &&
port === '443')))
      if (!isNonDefault || !withoutPort) return Array.from(new Set(arr))
      const preferred = (isIpHostname
        ? [baseUrl, withoutPort, altProtocol, altProtocolWithoutPort]
        : [withoutPort, baseUrl, altProtocolWithoutPort, altProtocol]
      ).filter(Boolean) as string[]
      return Array.from(new Set(preferred.concat(arr)))
    } catch {
      return Array.from(new Set(arr))
    }
  })()
  const out: string[] = []

  for (const root of roots) {
    const l = root.toLowerCase()
    const hasVersion = l.includes('/v1/') || l.endsWith('/v1') || l.includes('/v2/') || l.endsWith('/v2')
    const hasApiPrefix = l.includes('/api/') || l.endsWith('/api')

    if (hasVersion || hasApiPrefix) {
      out.push(root)
      continue
    }

    out.push(root)
    out.push(`${root}/api`)
    out.push(`${root}/api/v2`)
    out.push(`${root}/api/v1`)
    out.push(`${root}/v2`)
    out.push(`${root}/v1`)
  }

  return Array.from(new Set(out))
}

async function evolutionRequestAuto(opts: { baseUrl: string; apiKey=[REDACTED]
  const candidates = buildEvolutionBaseUrls(opts.baseUrl)
  const attempts: Array<{ baseUrl: string; url: string | null; status: number; ok: boolean }> = []
  let last: { ok: boolean; status: number; body: unknown; baseUrlUsed: string; url: string } | null = null
  const startedAt = Date.now()
  const deadlineMs = 35000
  const maxCandidates = 4
  for (const baseUrl of candidates) {
    if (attempts.length >= maxCandidates) break
    if (Date.now() - startedAt > deadlineMs) break
    const res = await evolutionRequest({ ...opts, baseUrl })
    const url = (res as unknown as { url?: string }).url ?? `${cleanUrl(baseUrl)}${opts.path.startsWith('/') ? '' :
'/'}${opts.path}`
    attempts.push({ baseUrl, url, status: res.status, ok: res.ok })
```

```
    last = { ...res, baseUrlUsed: baseUrl, url }
    if (res.status === 404) {
      if (opts.stopOn404?.({ body: res.body, url, baseUrlUsed: baseUrl })) return last
      continue
    }

    if (res.status === 502 || res.status === 504) {
      if (attempts.length >= 2) {
        return {
          ...last,
          body: {
            error: 'evolution_unreachable',
            attempts,
            last: res.body,
          },
        }
      }
      continue
    }

    return last
  }
  if (!last) {
    return { ok: false as const, status: 404, body: null, baseUrlUsed: candidates[0] ?? opts.baseUrl, url: '' }
  }

  if (last.status === 502 || last.status === 504) {
    return {
      ...last,
      body: {
        error: 'evolution_unreachable',
        attempts,
        last: last.body,
      },
    }
  }

  if (last.status === 404) {
    return {
      ...last,
      body: {
        error: 'not_found',
        attempts,
        last: last.body,
      },
    }
  }

  return last
}

function extractTextFragments(input: unknown): string[] {
  if (typeof input === 'string') return [input]
  if (!input) return []
  if (Array.isArray(input)) return input.flatMap((v) => extractTextFragments(v))
  if (typeof input !== 'object') return []
  const obj = input as Record<string, unknown>
  const keys = ['message', 'error', 'details', 'response', 'data', 'description']
  const out: string[] = []
  for (const key of keys) {
    if (obj[key] !== undefined) out.push(...extractTextFragments(obj[key]))
  }
  if (out.length) return out
  return Object.values(obj).flatMap((v) => extractTextFragments(v))
}

function unwrapNotFoundLast(details: unknown): unknown {
  if (!details || typeof details !== 'object') return details
  const obj = details as Record<string, unknown>
  if (obj.error === 'not_found' && obj.last !== undefined) return obj.last
  return details
}
```

```
function isEvolutionRouteNotFound(details: unknown) {
  const unwrapped = unwrapNotFoundLast(details)
  const text = extractTextFragments(unwrapped).join(' | ').toLowerCase()
  if (!text) return false
  if (text.includes('cannot get')) return true
  if (text.includes('<html')) return true
  if (text.includes('not found') && text.includes('/instance/')) return true
  return false
}

function isEvolutionInstanceNotFound(details: unknown, instanceName: string) {
  const unwrapped = unwrapNotFoundLast(details)
  const text = extractTextFragments(unwrapped).join(' | ').toLowerCase()
  if (!text) return false
  const hasInstance = text.includes('instance') || text.includes('instancia') || text.includes('instância')
  const notFound = text.includes('not found') || text.includes('não encontrada') || text.includes('nao encontrada') ||
text.includes('does not exist')
  if (!hasInstance || !notFound) return false
  const i = instanceName.trim().toLowerCase()
  if (!i) return true
  return text.includes(i)
}

function evolutionUrlHint(details: unknown) {
  if (!isEvolutionRouteNotFound(details)) return null
  return 'A URL pública não está apontando para a Evolution API (Cloudflare Tunnel/porta/serviço errado). Acesse a URL
no navegador: deve aparecer “Welcome to the Evolution API”.'
}

function isEvolutionUnauthorized(details: unknown) {
  const unwrapped = unwrapNotFoundLast(details)
  if (!unwrapped || typeof unwrapped !== 'object') return false
  const obj = unwrapped as Record<string, unknown>
  if (obj.status === 401) return true
  const text = extractTextFragments(unwrapped).join(' | ').toLowerCase()
  return text.includes('unauthorized')
}

function evolutionAuthHint(details: unknown) {
  if (!isEvolutionUnauthorized(details)) return null
  return 'A Evolution API está recusando a API Key (401). Confirme que o valor em Configurações > WhatsApp > API Key é
exatamente o mesmo configurado no container da Evolution como AUTHENTICATION_API_KEY. Na v2.3.7, a autenticação funciona
via header "apikey" (query ?apikey=[REDACTED]"apikey".'
}

function isEvolutionCloudflareTunnelError(details: unknown) {
  const unwrapped = unwrapNotFoundLast(details)
  const text = extractTextFragments(unwrapped).join(' | ').toLowerCase()
  if (!text) return false
  if (text.includes('cloudflare tunnel error')) return true
  if (text.includes('error code: 1033')) return true
  if (text.includes('argo tunnel error')) return true
  return false
}

function evolutionCloudflareTunnelHint(details: unknown) {
  if (!isEvolutionCloudflareTunnelError(details)) return null
  return 'O domínio da Evolution API está retornando uma página de erro do Cloudflare Tunnel (1033). Isso significa que
o tunnel/origem está offline ou desconectado. A correção permanente é manter o serviço de origem (container Evolution) e
o processo cloudflared rodando 24/7 (como serviço/systemd/PM2/Docker com restart=always) em um servidor/VPS sempre
ligado; fazer deploy da Edge Function não resolve a causa.'
}

function isEvolutionFetchFailed(details: unknown) {
  const unwrapped = unwrapNotFoundLast(details)
  if (!unwrapped || typeof unwrapped !== 'object') return false
  const obj = unwrapped as Record<string, unknown>
  if (obj.error === 'evolution_fetch_failed') return true
  if (obj.error === 'evolution_timeout') return true
  if (obj.error === 'evolution_unreachable') return true
  const text = extractTextFragments(unwrapped).join(' | ').toLowerCase()
```

```
  if (!text) return false
  return text.includes('signal has been aborted') || text.includes('timed out') || text.includes('timeout') ||
text.includes('fetch failed')
}

function evolutionNetworkHint(details: unknown) {
  if (!isEvolutionFetchFailed(details)) return null
  return 'Falha de rede ao chamar a Evolution API. Confirme que a URL é pública e responde rápido (teste no
navegador/curl). Se estiver usando IP:porta (ex.: 8080), libere a porta no firewall e considere expor via HTTPS/443
(proxy/tunnel) para maior estabilidade.'
}

function isEvolutionQuoteCommandError(details: unknown) {
  const unwrapped = unwrapNotFoundLast(details)
  const text = extractTextFragments(unwrapped).join(' | ').toLowerCase()
  if (!text) return false
  return text.includes('quote command returned error')
}

function evolutionQuoteCommandHint(details: unknown) {
  if (!isEvolutionQuoteCommandError(details)) return null
  return 'A Evolution retornou "Quote command returned error" ao enviar. O sistema tenta automaticamente variações de
formato do número e também remove emojis/caracteres incompatíveis; se persistir, confirme o telefone do cliente com DDD
+ 55 (ex: 5511999999999) e verifique os logs do container da Evolution.'
}

function isEvolutionNumberNotFound(details: unknown) {
  const unwrapped = unwrapNotFoundLast(details)
  const hasExistsFalse = (input: unknown): boolean => {
    if (!input || typeof input !== 'object') return false
    if (Array.isArray(input)) return input.some((v) => hasExistsFalse(v))
    const obj = input as Record<string, unknown>
    if (obj.exists === false) return true
    return Object.values(obj).some((v) => hasExistsFalse(v))
  }
  return hasExistsFalse(unwrapped)
}

function evolutionNumberNotFoundHint(details: unknown) {
  if (!isEvolutionNumberNotFound(details)) return null
  return 'A Evolution indicou que o número não existe no WhatsApp (exists=false). Confirme se o telefone está correto e
tente com DDD + 55 (ex.: 5531999999999).'
}

function evolutionHint(details: unknown) {
  return (
    evolutionUrlHint(details) ??
    evolutionAuthHint(details) ??
    evolutionCloudflareTunnelHint(details) ??
    evolutionNetworkHint(details) ??
    evolutionNumberNotFoundHint(details) ??
    evolutionQuoteCommandHint(details)
  )
}

async function createEvolutionInstance(opts: { apiUrl: string; apiKey=[REDACTED]
  const qrcode = typeof opts.qrcode === 'boolean' ? opts.qrcode : true
  const number = typeof opts.number === 'string' && opts.number.trim() ? sanitizePhone(opts.number) : ''

  const createResA = await evolutionRequestAuto({
    baseUrl: opts.apiUrl,
    apiKey=[REDACTED]
    path: '/instance/create',
    method: 'POST',
    body: { instanceName: opts.instanceName, token: opts.apiKey, qrcode, integration: 'WHATSAPP-BAILEYS', ...(number ? {
number } : {}) },
  })

  const createRes =
    createResA.ok || createResA.status === 409
      ? createResA
      : await evolutionRequestAuto({
```

```
          baseUrl: opts.apiUrl,
          apiKey=[REDACTED]
          path: '/instance/create',
          method: 'POST',
          body: { instanceName: opts.instanceName, qrcode, integration: 'WHATSAPP-BAILEYS', ...(number ? { number } :
{}) },
        })

  return createRes
}

function extractInstanceState(payload: unknown): string | null {
  if (!payload || typeof payload !== 'object') return null
  const root = payload as Record<string, unknown>
  const direct = root.state
  if (typeof direct === 'string') return direct
  const instance = root.instance
  if (instance && typeof instance === 'object') {
    const state = (instance as Record<string, unknown>).state
    if (typeof state === 'string') return state
  }
  const data = root.data
  if (data && typeof data === 'object') {
    const state = (data as Record<string, unknown>).state
    if (typeof state === 'string') return state
  }
  return null
}

function extractQrBase64(payload: unknown): string | null {
  if (!payload || typeof payload !== 'object') return null
  const obj = payload as Record<string, unknown>
  const direct = obj.qrcode ?? obj.qr ?? obj.qrCode ?? obj.qr_code ?? obj.base64
  const normalize = (value: string) => {
    const v = value.trim()
    if (!v) return null
    const idx = v.toLowerCase().indexOf('base64,')
    if (idx >= 0) return v.slice(idx + 'base64,'.length).trim() || null
    return v
  }
  const fromObj = (candidate: unknown): string | null => {
    if (typeof candidate === 'string') return normalize(candidate)
    if (!candidate || typeof candidate !== 'object') return null
    const c = candidate as Record<string, unknown>
    const nested = c.base64 ?? c.qrcode ?? c.qr ?? c.qrCode ?? c.qr_code
    if (typeof nested === 'string') return normalize(nested)
    const nestedBase64 = c.base64
    if (typeof nestedBase64 === 'string') return normalize(nestedBase64)
    return null
  }

  const directValue = fromObj(direct)
  if (directValue) return directValue

  const fromQrcode = fromObj(obj.qrcode)
  if (fromQrcode) return fromQrcode

  const fromData = fromObj(obj.data)
  if (fromData) return fromData

  const fromInstance = fromObj(obj.instance)
  if (fromInstance) return fromInstance

  if (obj.data && typeof obj.data === 'object') {
    const deep = extractQrBase64(obj.data)
    if (deep) return deep
  }
  if (obj.instance && typeof obj.instance === 'object') {
    const deep = extractQrBase64(obj.instance)
    if (deep) return deep
  }
  return null
```

```
}

function extractPairingCode(payload: unknown): string | null {
  if (!payload || typeof payload !== 'object') return null
  const obj = payload as Record<string, unknown>
  const fromStr = (v: unknown) => (typeof v === 'string' && v.trim() ? v.trim() : null)

  const direct = fromStr(obj.pairingCode) ?? fromStr(obj.pairing_code)
  if (direct) return direct

  const scan = (candidate: unknown): string | null => {
    if (!candidate || typeof candidate !== 'object') return null
    const c = candidate as Record<string, unknown>
    return fromStr(c.pairingCode) ?? fromStr(c.pairing_code)
  }

  const fromQrcode = scan(obj.qrcode)
  if (fromQrcode) return fromQrcode
  const fromData = scan(obj.data)
  if (fromData) return fromData
  const fromInstance = scan(obj.instance)
  if (fromInstance) return fromInstance

  if (obj.data && typeof obj.data === 'object') {
    const deep = extractPairingCode(obj.data)
    if (deep) return deep
  }
  if (obj.instance && typeof obj.instance === 'object') {
    const deep = extractPairingCode(obj.instance)
    if (deep) return deep
  }
  return null
}

async function getPrincipal(userClient: ReturnType<typeof createClient>, uid: string) {
  const { data: superAdminRaw, error: superAdminErr } = await userClient.from('super_admin').select('id').eq('id',
uid).maybeSingle()
  if (superAdminErr) {
    if (!isMissingTableError(superAdminErr.message)) {
      return { ok: false as const, error: 'principal_query_error' as const, table: 'super_admin' as const, message:
superAdminErr.message }
    }
  }
  const superAdmin = (superAdminRaw ?? null) as unknown as SuperAdminAuthRow | null
  if (superAdmin?.id) return { ok: true as const, kind: 'super_admin' as const, uid }

  const { data: usuarioRowRaw, error: usuarioErr } = await userClient.from('usuarios').select('id,tipo_conta').eq('id',
uid).maybeSingle()
  if (usuarioErr) {
    if (isMissingTableError(usuarioErr.message)) {
      return { ok: false as const, error: 'schema_incomplete' as const, table: 'usuarios' as const, message:
usuarioErr.message }
    }
    return { ok: false as const, error: 'principal_query_error' as const, table: 'usuarios' as const, message:
usuarioErr.message }
  }
  const usuarioRow = (usuarioRowRaw ?? null) as unknown as UsuarioAuthRow | null
  if (usuarioRow?.id) return { ok: true as const, kind: 'usuario' as const, uid, masterId: uid, tipo_conta:
usuarioRow.tipo_conta ?? '' }

  const { data: funcRowRaw, error: funcErr } = await
userClient.from('funcionarios').select('id,usuario_master_id').eq('id', uid).maybeSingle()
  if (funcErr) {
    if (isMissingTableError(funcErr.message)) {
      return { ok: false as const, error: 'schema_incomplete' as const, table: 'funcionarios' as const, message:
funcErr.message }
    }
    return { ok: false as const, error: 'principal_query_error' as const, table: 'funcionarios' as const, message:
funcErr.message }
  }
  const funcRow = (funcRowRaw ?? null) as unknown as FuncionarioAuthRow | null
  if (funcRow?.id && funcRow.usuario_master_id) return { ok: true as const, kind: 'funcionario' as const, uid, masterId:
```

```
funcRow.usuario_master_id }

  return { ok: false as const, error: 'unauthorized' as const }
}

Deno.serve(async (req) => {
  if (req.method === 'OPTIONS') return jsonResponse(200, { ok: true })
  if (req.method !== 'POST') return jsonResponse(405, { error: 'method_not_allowed' })

  const supabaseUrl = Deno.env.get('SUPABASE_URL') ?? ''
  const anonKey = Deno.env.get('SUPABASE_ANON_KEY') ?? ''
  if (!supabaseUrl || !anonKey) return jsonResponse(500, { error: 'missing_env' })

  const serviceRoleKey=[REDACTED]'SERVICE_ROLE_KEY') ?? Deno.env.get('SUPABASE_SERVICE_ROLE_KEY') ?? ''
  const resendApiKey=[REDACTED]'RESEND_API_KEY') ?? '').trim()
  const resendFromRaw = (Deno.env.get('RESEND_FROM') ?? '').trim()
  const resendFrom = resendFromRaw || 'SMagenda <onboarding@resend.dev>'

  const jwt = extractJwt(req)
  if (!jwt) {
    return jsonResponse(401, {
      error: 'unauthorized',
      message: 'missing_jwt',
      hint: 'Envie o access_token do Supabase no header Authorization: Bearer <token> (ou x-user-jwt).',
    })
  }

  const shapeCheck = validateJwtShapeAndProject(jwt, supabaseUrl)
  if (!shapeCheck.ok) {
    if (shapeCheck.reason === 'jwt_project_mismatch') {
      return jsonResponse(401, { error: 'invalid_jwt', reason: shapeCheck.reason, iss: shapeCheck.iss, expected:
shapeCheck.expectedPrefix })
    }
    return jsonResponse(401, { error: 'invalid_jwt', reason: shapeCheck.reason })
  }

  const baseClient = createClient(supabaseUrl, anonKey)
  const { data: authData, error: authErr } = await baseClient.auth.getUser(jwt)
  if (authErr || !authData?.user?.id) {
    const message = typeof authErr?.message === 'string' ? authErr.message : 'Invalid JWT'
    return jsonResponse(401, { error: 'invalid_jwt', reason: 'supabase_auth_rejected', message })
  }

  const userClient = createClient(supabaseUrl, anonKey, {
    global: {
      headers: {
        Authorization: `Bearer ${jwt}`,
      },
    },
  })

  const serviceClient = serviceRoleKey
    ? createClient(supabaseUrl, serviceRoleKey, {
        auth: { persistSession: false, autoRefreshToken: false },
      })
    : null

  const principalClient = serviceClient ?? userClient
  const dbClient = serviceClient ?? userClient

  let payload: Payload
  try {
    payload = (await req.json()) as Payload
  } catch {
    return jsonResponse(400, { error: 'invalid_json' })
  }

  const uid = authData.user.id
  const isAdminAction =
    payload.action === 'admin_diagnostics' ||
    payload.action === 'admin_status' ||
    payload.action === 'admin_connect' ||
```

```
    payload.action === 'admin_disconnect' ||
    payload.action === 'admin_send_aviso'

  let principal:
    | { ok: true; kind: 'super_admin'; uid: string }
    | { ok: true; kind: 'usuario'; uid: string; masterId: string; tipo_conta: string }
    | { ok: true; kind: 'funcionario'; uid: string; masterId: string }
    | { ok: false; error: string; table?: string; message?: string }

  if (isAdminAction) {
    const { data: saRow, error: saErr } = await dbClient.from('super_admin').select('id').eq('id', uid).maybeSingle()
    if (saErr) {
      if (isMissingTableError(saErr.message) || isMissingColumnError(saErr.message)) {
        return jsonResponse(400, { error: 'schema_incomplete', message: 'Execute o SQL do WhatsApp (Super Admin) em
/admin/configuracoes.' })
      }
      return jsonResponse(403, { error: 'not_allowed', message: saErr.message })
    }
    if (!saRow) {
      return jsonResponse(403, {
        error: 'not_allowed',
        message: 'Conta não cadastrada como Super Admin (public.super_admin).',
        hint: 'Acesse /admin/bootstrap para criar o Super Admin e /admin/configuracoes para aplicar o SQL.',
      })
    }
    principal = { ok: true, kind: 'super_admin', uid }
  } else {
    principal = await getPrincipal(principalClient, uid)
    if (!principal.ok && principal.error === 'unauthorized') {
      const { error: ensureErr } = await userClient.rpc('ensure_usuario_profile')
      if (!ensureErr) {
        principal = await getPrincipal(principalClient, uid)
      } else {
        const msg = typeof ensureErr.message === 'string' ? ensureErr.message : ''
        const lower = msg.toLowerCase()
        const missingFn = lower.includes('ensure_usuario_profile') && (lower.includes('function') ||
lower.includes('rpc'))
        if (missingFn) {
          return jsonResponse(400, { error: 'schema_incomplete', message: 'Crie a função ensure_usuario_profile (trial)
em /admin/configuracoes.' })
        }
      }
    }

    if (!principal.ok) {
      if (principal.error === 'unauthorized') {
        return jsonResponse(401, {
          error: 'unauthorized',
          hint: serviceClient
            ? 'Usuário não existe em public.super_admin/public.usuarios/public.funcionarios.'
            : 'Usuário não existe nas tabelas de perfil ou as políticas RLS estão bloqueando leitura. Execute o SQL de
bootstrap/políticas no Supabase.',
        })
      }
      if (principal.error === 'schema_incomplete') {
        return jsonResponse(400, {
          error: 'schema_incomplete',
          table: (principal as unknown as { table?: unknown }).table ?? null,
          message: (principal as unknown as { message?: unknown }).message ?? null,
        })
      }
      return jsonResponse(403, { error: principal.error, table: (principal as unknown as { table?: unknown }).table ??
null, message: (principal as unknown as { message?: unknown }).message ?? null })
    }
  }

  if (
    payload.action === 'admin_diagnostics' ||
    payload.action === 'admin_status' ||
    payload.action === 'admin_connect' ||
    payload.action === 'admin_disconnect' ||
    payload.action === 'admin_send_aviso'
```

```
  ) {
    if (principal.kind !== 'super_admin') return jsonResponse(403, { error: 'not_allowed' })

    const { data: saRaw, error: saErr } = await dbClient
      .from('super_admin')
      .select('id,whatsapp_api_url,whatsapp_api_key,whatsapp_instance_name')
      .eq('id', principal.uid)
      .maybeSingle()

    if (saErr) {
      if (isMissingColumnError(saErr.message)) {
        return jsonResponse(400, { error: 'schema_incomplete', message: 'Execute o SQL do WhatsApp (Super Admin) em
/admin/configuracoes.' })
      }
      return jsonResponse(403, { error: 'not_allowed' })
    }
    if (!saRaw) return jsonResponse(403, { error: 'not_allowed' })

    const sa = saRaw as unknown as SuperAdminConfigRow
    const apiUrl = sa.whatsapp_api_url
    const apiKey=[REDACTED]
    const instanceNameRaw = sa.whatsapp_instance_name ?? `admin-${principal.uid.slice(0, 8)}`
    const instanceName = sanitizeInstanceName(instanceNameRaw)

    if (payload.action === 'admin_diagnostics') {
      const urlValidation = apiUrl ? validateEvolutionBaseUrl(apiUrl) : { ok: false as const, reason: 'missing_url' as
const }
      const evolution =
        apiUrl && apiKey && (urlValidation as unknown as { ok?: unknown }).ok === true
          ? await (async () => {
              const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)
              const res = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path:
`/instance/connectionState/${instanceName}`, method: 'GET', stopOn404 })
              const hint = res.ok ? null : evolutionHint(res.body)
              const state = res.ok ? extractInstanceState(res.body) : null
              return { ok: res.ok, status: res.status, state, body: res.body, hint }
            })()
          : null

      return jsonResponse(200, {
        ok: true,
        uid: principal.uid,
        hasServiceRoleKey=[REDACTED]
        superAdminRow: { id: sa.id },
        config: {
          hasApiUrl: Boolean(apiUrl),
          hasApiKey=[REDACTED]
          instanceName,
        },
        urlValidation,
        evolution,
      })
    }

    if (!apiUrl || !apiKey) return jsonResponse(400, { error: 'whatsapp_not_configured' })

    const v = validateEvolutionBaseUrl(apiUrl)
    if (!v.ok) {
      const cleaned = (v as unknown as { cleaned?: string }).cleaned ?? null
      const hostname = (() => {
        if (!cleaned) return null
        try {
          return new URL(cleaned).hostname
        } catch {
          return null
        }
      })()
      return jsonResponse(400, {
        error: 'invalid_evolution_url',
        reason: v.reason,
        cleaned,
        hostname,
```

```
    message: 'A URL da Evolution API precisa ser pública (não pode ser localhost/IP privado).',
   })
  }

  if (payload.action === 'admin_status') {
   const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)
   let stateRes = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path:
`/instance/connectionState/${instanceName}`, method: 'GET', stopOn404 })
   if (!stateRes.ok && stateRes.status === 404 && isEvolutionInstanceNotFound(stateRes.body, instanceName)) {
    const createRes = await createEvolutionInstance({ apiUrl, apiKey, instanceName, qrcode: pairingNumber ? false :
true, number: pairingNumber || null })
    if (!createRes.ok && createRes.status !== 409) {
     const hint = evolutionHint(createRes.body)
     return jsonResponse(createRes.status, { error: 'evolution_error', details: createRes.body, hint })
    }
    stateRes = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path:
`/instance/connectionState/${instanceName}`, method: 'GET', stopOn404 })
   }
   if (!stateRes.ok) {
    const hint = evolutionHint(stateRes.body)
    return jsonResponse(stateRes.status, { error: 'evolution_error', details: stateRes.body, hint })
   }
   const state = extractInstanceState(stateRes.body)
   return jsonResponse(200, { ok: true, instanceName, state })
  }

  if (payload.action === 'admin_connect') {
   const pairingNumber = typeof payload.number === 'string' ? sanitizePhone(payload.number) : ''
   const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)
   const stateRes = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path:
`/instance/connectionState/${instanceName}`, method: 'GET', stopOn404 })
   const state = stateRes.ok ? extractInstanceState(stateRes.body) : null
   let qrFromCreate: string | null = null
   let pairingFromCreate: string | null = null

   if (state === 'open') {
    if (!sa.whatsapp_instance_name) {
     const { error: updErr } = await dbClient.from('super_admin').update({ whatsapp_instance_name: instanceName
}).eq('id', principal.uid)
     if (updErr && !isMissingColumnError(updErr.message)) return jsonResponse(400, { error:
'save_instance_name_failed', message: updErr.message })
    }
    return jsonResponse(200, { ok: true, instanceName, state })
   }

   if (!stateRes.ok && stateRes.status === 404 && isEvolutionInstanceNotFound(stateRes.body, instanceName)) {
    const createRes = await createEvolutionInstance({ apiUrl, apiKey, instanceName })
    if (!createRes.ok && createRes.status !== 409) {
     const hint = evolutionHint(createRes.body)
     return jsonResponse(createRes.status, { error: 'evolution_error', details: createRes.body, hint })
    }
    if (createRes.ok) {
     qrFromCreate = extractQrBase64(createRes.body)
     pairingFromCreate = extractPairingCode(createRes.body)
    }
   } else if (!stateRes.ok && stateRes.status === 404) {
    const hint = evolutionHint(stateRes.body)
    return jsonResponse(404, { error: 'evolution_error', details: stateRes.body, hint })
   }

   const connectCandidates: Array<{ method: 'GET' | 'POST'; path: string; body?: unknown }> = pairingNumber
    ? [
      { method: 'GET', path: `/instance/connect/${instanceName}?
number=${encodeURIComponent(pairingNumber)}&qrcode=false` },
      { method: 'GET', path: `/instance/connect/${instanceName}?
number=${encodeURIComponent(pairingNumber)}&qrCode=false` },
      { method: 'GET', path: `/instance/connect/${instanceName}?
number=${encodeURIComponent(pairingNumber)}&pairingCode=true` },
      { method: 'GET', path: `/instance/connect/${instanceName}?number=${encodeURIComponent(pairingNumber)}` },
      { method: 'POST', path: `/instance/connect/${instanceName}`, body: { number: pairingNumber, qrcode: false }
},
      { method: 'POST', path: `/instance/connect/${instanceName}`, body: { number: pairingNumber, qrCode: false }
```

```
},
      { method: 'POST', path: `/instance/connect/${instanceName}`, body: { number: pairingNumber, pairingCode:
true } },
      { method: 'POST', path: `/instance/connect/${instanceName}`, body: { number: pairingNumber, usePairingCode:
true } },
      { method: 'POST', path: `/instance/connect/${instanceName}`, body: { number: pairingNumber } },
      { method: 'GET', path: `/instance/connect?
instance=${encodeURIComponent(instanceName)}&number=${encodeURIComponent(pairingNumber)}&qrcode=false` },
      { method: 'GET', path: `/instance/connect?
instanceName=${encodeURIComponent(instanceName)}&number=${encodeURIComponent(pairingNumber)}&qrcode=false` },
      { method: 'GET', path: `/instance/connect?
instance=${encodeURIComponent(instanceName)}&number=${encodeURIComponent(pairingNumber)}` },
      { method: 'GET', path: `/instance/connect?
instanceName=${encodeURIComponent(instanceName)}&number=${encodeURIComponent(pairingNumber)}` },
      { method: 'POST', path: `/instance/connect`, body: { instanceName, number: pairingNumber, qrcode: false } },
      { method: 'POST', path: `/instance/connect`, body: { instance: instanceName, number: pairingNumber, qrcode:
false } },
      { method: 'POST', path: `/instance/connect`, body: { instanceName, number: pairingNumber, pairingCode: true,
qrcode: false } },
      { method: 'POST', path: `/instance/connect`, body: { instance: instanceName, number: pairingNumber,
pairingCode: true, qrcode: false } },
      { method: 'POST', path: `/instance/connect`, body: { instanceName, number: pairingNumber } },
      { method: 'POST', path: `/instance/connect`, body: { instance: instanceName, number: pairingNumber } },
      { method: 'GET', path: `/instance/connect/${instanceName}` },
      { method: 'POST', path: `/instance/connect/${instanceName}` },
      { method: 'GET', path: `/instance/connect?instance=${encodeURIComponent(instanceName)}` },
      { method: 'GET', path: `/instance/connect?instanceName=${encodeURIComponent(instanceName)}` },
      { method: 'POST', path: `/instance/connect`, body: { instanceName } },
      { method: 'POST', path: `/instance/connect`, body: { instance: instanceName } },
     ]
    : [
      { method: 'GET', path: `/instance/connect/${instanceName}` },
      { method: 'POST', path: `/instance/connect/${instanceName}` },
      { method: 'GET', path: `/instance/connect?instance=${encodeURIComponent(instanceName)}` },
      { method: 'GET', path: `/instance/connect?instanceName=${encodeURIComponent(instanceName)}` },
      { method: 'POST', path: `/instance/connect`, body: { instanceName } },
      { method: 'POST', path: `/instance/connect`, body: { instance: instanceName } },
     ]

   const qrCandidates: Array<{ method: 'GET' | 'POST'; path: string; body?: unknown }> = [
    { method: 'GET', path: `/instance/qrcode/${instanceName}` },
    { method: 'GET', path: `/instance/qrCode/${instanceName}` },
    { method: 'GET', path: `/instance/qr/${instanceName}` },
   ]

   let connectState: string | null = null
   let qrBase64: string | null = qrFromCreate
   let pairingCode: string | null = pairingFromCreate

   for (const c of connectCandidates) {
    const res = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path: c.path, method: c.method, body: c.body
})
    if (!res.ok) {
     if (res.status === 404) continue
     const hint = evolutionHint(res.body)
     return jsonResponse(res.status, { error: 'evolution_error', details: res.body, hint })
    }
    connectState = extractInstanceState(res.body) ?? connectState
    qrBase64 = extractQrBase64(res.body) ?? qrBase64
    pairingCode = extractPairingCode(res.body) ?? pairingCode
    if (pairingNumber) {
     if (pairingCode) break
    } else {
     if (qrBase64 || pairingCode) break
    }
   }

   if (!qrBase64 && !pairingNumber) {
    for (const c of qrCandidates) {
     const res = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path: c.path, method: c.method, body: c.body
})
     if (!res.ok) {
```

```
            if (res.status === 404) continue
            const hint = evolutionHint(res.body)
            return jsonResponse(res.status, { error: 'evolution_error', details: res.body, hint })
          }
          connectState = extractInstanceState(res.body) ?? connectState
          qrBase64 = extractQrBase64(res.body) ?? qrBase64
          pairingCode = extractPairingCode(res.body) ?? pairingCode
          if (qrBase64) break
        }
      }

      if (!connectState) {
        const stateAfterRes = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path:
`/instance/connectionState/${instanceName}`, method: 'GET', stopOn404 })
        if (stateAfterRes.ok) connectState = extractInstanceState(stateAfterRes.body)
      }

      const hint = (() => {
        if (pairingNumber && !pairingCode) {
          return 'A Evolution API não retornou código de pareamento para este número. Se sua versão não suporta
pareamento por código, use “Conectar (QR Code)” em outro aparelho.'
        }
        if (!qrBase64 && !pairingCode && (connectState === 'connecting' || connectState === 'close' || connectState ===
'closed')) {
          return 'A instância ainda não retornou QR Code. Isso costuma acontecer quando a Evolution está iniciando ou
quando a versão do WhatsApp Web no container está desatualizada. Verifique os logs do container e, se necessário,
atualize CONFIG_SESSION_PHONE_VERSION.'
        }
        return null
      })()

      if (!sa.whatsapp_instance_name) {
        const { error: updErr } = await dbClient.from('super_admin').update({ whatsapp_instance_name: instanceName
}).eq('id', principal.uid)
        if (updErr && !isMissingColumnError(updErr.message)) return jsonResponse(400, { error:
'save_instance_name_failed', message: updErr.message })
      }

      const qrSafe = pairingNumber ? null : qrBase64
      return jsonResponse(200, { ok: true, instanceName, state: connectState ?? state, qrBase64: qrSafe, pairingCode,
hint })
    }

    if (payload.action === 'admin_disconnect') {
      const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)
      let logoutRes = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path: `/instance/logout/${instanceName}`,
method: 'DELETE', stopOn404 })
      if (!logoutRes.ok && logoutRes.status === 405) {
        logoutRes = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path: `/instance/logout/${instanceName}`,
method: 'GET', stopOn404 })
      }
      if (!logoutRes.ok && logoutRes.status !== 404) {
        const hint = evolutionHint(logoutRes.body)
        return jsonResponse(logoutRes.status, { error: 'evolution_error', details: logoutRes.body, hint })
      }
      const deleteRes = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path: `/instance/delete/${instanceName}`,
method: 'DELETE', stopOn404 })
      if (!deleteRes.ok && deleteRes.status !== 404) {
        const hint = evolutionHint(deleteRes.body)
        return jsonResponse(deleteRes.status, { error: 'evolution_error', details: deleteRes.body, hint })
      }
      return jsonResponse(200, { ok: true, instanceName })
    }

    if (payload.action === 'admin_send_aviso') {
      const ids = Array.isArray(payload.cliente_ids) ? payload.cliente_ids.map((v) => String(v)).filter(Boolean) : []
      const text = String(payload.text ?? '').trim()
      if (!ids.length || !text) return jsonResponse(400, { error: 'invalid_payload' })
      if (ids.length > 500) return jsonResponse(400, { error: 'too_many_recipients', max: 500 })

      const { data: rows, error: rowsErr } = await
userClient.from('usuarios').select('id,telefone,nome_negocio').in('id', ids)
```

```
    if (rowsErr) return jsonResponse(403, { error: 'not_allowed', message: rowsErr.message })

    const targets = (rows ?? []) as unknown as ClienteAvisoRow[]
    const byId: Record<string, ClienteAvisoRow> = {}
    for (const r of targets) byId[r.id] = r

    const results: Array<{ id: string; nome_negocio: string | null; status: string; details?: unknown }> = []

    let sent = 0
    let failed = 0
    let skippedNoPhone = 0
    let skippedInvalidPhone = 0

    const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)

    for (const id of ids) {
      const row = byId[id] ?? null
      if (!row) {
        failed += 1
        results.push({ id, nome_negocio: null, status: 'not_found' })
        continue
      }

      const rawPhone = row.telefone ?? ''
      if (!rawPhone.trim()) {
        skippedNoPhone += 1
        results.push({ id: row.id, nome_negocio: row.nome_negocio ?? null, status: 'skipped_no_phone' })
        continue
      }

      const phoneRaw = rawPhone
      if (!sanitizePhone(rawPhone)) {
        skippedInvalidPhone += 1
        results.push({ id: row.id, nome_negocio: row.nome_negocio ?? null, status: 'skipped_invalid_phone', details: {
telefone: rawPhone } })
        continue
      }

      let sendRes = await evolutionSendTextWithFallback({ apiUrl, apiKey, instanceName, phoneRaw, text, stopOn404 })

      if (!sendRes.ok && sendRes.status === 404 && isEvolutionInstanceNotFound(sendRes.body, instanceName)) {
        const createRes = await createEvolutionInstance({ apiUrl, apiKey, instanceName })
        if (!createRes.ok && createRes.status !== 409) {
          const hint = evolutionHint(createRes.body)
          failed += 1
          results.push({ id: row.id, nome_negocio: row.nome_negocio ?? null, status: 'failed', details: { error:
'evolution_error', details: createRes.body, hint } })
          continue
        }
        sendRes = await evolutionSendTextWithFallback({ apiUrl, apiKey, instanceName, phoneRaw, text, stopOn404 })
      }

      if (!sendRes.ok) {
        const hint = evolutionHint(sendRes.body)
        failed += 1
        results.push({
          id: row.id,
          nome_negocio: row.nome_negocio ?? null,
          status: 'failed',
          details: { status: sendRes.status, body: sendRes.body, hint, attempts: (sendRes as unknown as { attempts?:
unknown }).attempts ?? null },
        })
        continue
      }

      sent += 1
      results.push({ id: row.id, nome_negocio: row.nome_negocio ?? null, status: 'sent' })
    }

    return jsonResponse(200, {
      ok: true,
      totals: {
```

```
          selected: ids.length,
          found: targets.length,
          sent,
          failed,
          skipped_no_phone: skippedNoPhone,
          skipped_invalid_phone: skippedInvalidPhone,
        },
        results,
      })
    }

    return jsonResponse(400, { error: 'unknown_action' })
  }

  if (principal.kind === 'super_admin') return jsonResponse(403, { error: 'not_allowed' })

  const masterId = principal.masterId

  const { data: userBase, error: userBaseErr } = await dbClient
    .from('usuarios')
    .select('id,slug,nome_negocio,telefone,endereco,whatsapp_habilitado')
    .eq('id', masterId)
    .maybeSingle()

  if (userBaseErr || !userBase) {
    return jsonResponse(403, { error: 'not_allowed' })
  }

  const selectExtraFull =

'whatsapp_instance_name,enviar_confirmacao,enviar_lembrete,enviar_cancelamento,lembrete_horas_antes,mensagem_confirmacao
,mensagem_lembrete,mensagem_cancelamento'
  const selectExtraFallback =
'whatsapp_instance_name,enviar_confirmacao,enviar_lembrete,lembrete_horas_antes,mensagem_confirmacao,mensagem_lembrete'

  const userExtraFirst = await dbClient.from('usuarios').select(selectExtraFull).eq('id', masterId).maybeSingle()
  const userExtraSecond =
    userExtraFirst.error && isMissingColumnError(userExtraFirst.error.message)
      ? await dbClient.from('usuarios').select(selectExtraFallback).eq('id', masterId).maybeSingle()
      : null

  const userExtraRaw = (userExtraSecond ? userExtraSecond.data : userExtraFirst.data) as unknown
  const userExtraErr = userExtraSecond ? userExtraSecond.error : userExtraFirst.error

  const userExtraOk = !userExtraErr || !isMissingColumnError(userExtraErr.message)
  const userExtraSafe = userExtraOk ? (userExtraRaw as unknown) : null

  const userBaseRow = userBase as unknown as UsuarioBaseRow
  const userExtra = (userExtraSafe ?? null) as unknown as UsuarioExtraRow | null

  if (!serviceClient) {
    return jsonResponse(500, {
      error: 'whatsapp_config_requires_service_role',
      message: 'A Edge Function precisa do SERVICE_ROLE_KEY para ler a configuração global do WhatsApp.',
      hint: 'No Supabase: Project Settings → Edge Functions → Secrets. Defina SERVICE_ROLE_KEY (ou
SUPABASE_SERVICE_ROLE_KEY) e faça o deploy da função novamente.',
    })
  }

  const globalCfg = await loadGlobalWhatsappConfig(dbClient)
  if (!globalCfg.ok) {
    if (globalCfg.error === 'schema_incomplete') {
      return jsonResponse(400, { error: 'schema_incomplete', message: 'Execute o SQL do WhatsApp (Super Admin) em
/admin/configuracoes.' })
    }
    if (globalCfg.error === 'permission_denied') {
      return jsonResponse(500, {
        error: 'whatsapp_config_requires_service_role',
        message: 'A Edge Function precisa do SERVICE_ROLE_KEY para ler a configuração global do WhatsApp.',
        hint: 'No Supabase: Project Settings → Edge Functions → Secrets. Defina SERVICE_ROLE_KEY (ou
SUPABASE_SERVICE_ROLE_KEY) e faça o deploy da função novamente.',
      })
```

```
    }
    return jsonResponse(502, { error: 'load_whatsapp_config_failed', message: globalCfg.message })
  }

  const apiUrl = globalCfg.apiUrl
  const apiKey=[REDACTED]
  const enabled = userBaseRow.whatsapp_habilitado
  const whatsappEnabled = enabled === undefined || enabled === null ? true : Boolean(enabled)
  const slug = userBaseRow.slug ?? ''
  const instanceNameRaw = userExtra?.whatsapp_instance_name ?? slug
  const instanceName = sanitizeInstanceName(instanceNameRaw)

  if (payload.action === 'config_status') {
    const configured = Boolean(apiUrl && apiKey)
    if (!configured) return jsonResponse(200, { ok: true, configured: false })
    const v = validateEvolutionBaseUrl(apiUrl)
    return jsonResponse(200, { ok: true, configured: v.ok })
  }

  if (!whatsappEnabled) {
    return jsonResponse(403, { error: 'whatsapp_disabled' })
  }

  if (payload.action === 'connect' || payload.action === 'status' || payload.action === 'disconnect' || payload.action
=== 'send_test') {
    if (principal.kind !== 'usuario' || principal.uid !== masterId) {
      return jsonResponse(403, { error: 'only_master_can_manage' })
    }
  }

  if (payload.action !== 'send_confirmacao' && payload.action !== 'send_cancelamento' && (!apiUrl || !apiKey)) {
    return jsonResponse(400, { error: 'whatsapp_not_configured' })
  }

  if (payload.action !== 'send_confirmacao' && payload.action !== 'send_cancelamento') {
    const v = validateEvolutionBaseUrl(apiUrl ?? '')
    if (!v.ok) {
      const cleaned = (v as unknown as { cleaned?: string }).cleaned ?? null
      const hostname = (() => {
        if (!cleaned) return null
        try {
          return new URL(cleaned).hostname
        } catch {
          return null
        }
      })()
      return jsonResponse(400, {
        error: 'invalid_evolution_url',
        reason: v.reason,
        cleaned,
        hostname,
        message: 'A URL da Evolution API precisa ser pública (não pode ser localhost/IP privado).',
      })
    }
  }

  if (payload.action === 'status') {
    const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)
    let stateRes = await evolutionRequestAuto({
      baseUrl: apiUrl!,
      apiKey=[REDACTED]
      path: `/instance/connectionState/${instanceName}`,
      method: 'GET',
      stopOn404,
    })

      if (!stateRes.ok && stateRes.status === 404 && isEvolutionInstanceNotFound(stateRes.body, instanceName)) {
        const createRes = await createEvolutionInstance({ apiUrl: apiUrl!, apiKey=[REDACTED]
        if (!createRes.ok && createRes.status !== 409) {
          const hint = evolutionHint(createRes.body)
          return jsonResponse(createRes.status, { error: 'evolution_error', details: createRes.body, hint })
        }
```

```
      stateRes = await evolutionRequestAuto({
        baseUrl: apiUrl!,
        apiKey=[REDACTED]
        path: `/instance/connectionState/${instanceName}`,
        method: 'GET',
        stopOn404,
      })
    }

    if (!stateRes.ok) {
      const hint = evolutionHint(stateRes.body)
      return jsonResponse(stateRes.status, { error: 'evolution_error', details: stateRes.body, hint })
    }
    const state = extractInstanceState(stateRes.body)
    return jsonResponse(200, { ok: true, instanceName, state })
  }

  if (payload.action === 'connect') {
    const pairingNumber = typeof payload.number === 'string' ? sanitizePhone(payload.number) : ''
    const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)
    const stateRes = await evolutionRequestAuto({
      baseUrl: apiUrl!,
      apiKey=[REDACTED]
      path: `/instance/connectionState/${instanceName}`,
      method: 'GET',
      stopOn404,
    })
    const state = stateRes.ok ? extractInstanceState(stateRes.body) : null
    let qrFromCreate: string | null = null
    let pairingFromCreate: string | null = null

    if (state === 'open') {
      if (userExtraOk) {
        const { error: updErr } = await dbClient.from('usuarios').update({ whatsapp_instance_name: instanceName
}).eq('id', masterId)
        if (updErr && !isMissingColumnError(updErr.message)) {
          return jsonResponse(400, { error: 'save_instance_name_failed', message: updErr.message })
        }
      }
      return jsonResponse(200, { ok: true, instanceName, state })
    }

    if (!stateRes.ok && stateRes.status === 404 && isEvolutionInstanceNotFound(stateRes.body, instanceName)) {
      const createRes = await createEvolutionInstance({ apiUrl: apiUrl!, apiKey=[REDACTED]
      if (!createRes.ok && createRes.status !== 409) {
        const hint = evolutionHint(createRes.body)
        return jsonResponse(createRes.status, { error: 'evolution_error', details: createRes.body, hint })
      }
      if (createRes.ok) {
        qrFromCreate = extractQrBase64(createRes.body)
        pairingFromCreate = extractPairingCode(createRes.body)
      }
    } else if (!stateRes.ok && stateRes.status === 404) {
      const hint = evolutionHint(stateRes.body)
      return jsonResponse(404, { error: 'evolution_error', details: stateRes.body, hint })
    }

    const connectCandidates: Array<{ method: 'GET' | 'POST'; path: string; body?: unknown }> = pairingNumber
      ? [
          { method: 'GET', path: `/instance/connect/${instanceName}?number=${encodeURIComponent(pairingNumber)}` },
          { method: 'POST', path: `/instance/connect/${instanceName}`, body: { number: pairingNumber } },
          { method: 'GET', path: `/instance/connect?
instance=${encodeURIComponent(instanceName)}&number=${encodeURIComponent(pairingNumber)}` },
          { method: 'GET', path: `/instance/connect?
instanceName=${encodeURIComponent(instanceName)}&number=${encodeURIComponent(pairingNumber)}` },
          { method: 'POST', path: `/instance/connect`, body: { instanceName, number: pairingNumber } },
          { method: 'POST', path: `/instance/connect`, body: { instance: instanceName, number: pairingNumber } },
          { method: 'GET', path: `/instance/connect/${instanceName}` },
          { method: 'POST', path: `/instance/connect/${instanceName}` },
          { method: 'GET', path: `/instance/connect?instance=${encodeURIComponent(instanceName)}` },
          { method: 'GET', path: `/instance/connect?instanceName=${encodeURIComponent(instanceName)}` },
          { method: 'POST', path: `/instance/connect`, body: { instanceName } },
```

```
          { method: 'POST', path: `/instance/connect`, body: { instance: instanceName } },
        ]
      : [
          { method: 'GET', path: `/instance/connect/${instanceName}` },
          { method: 'POST', path: `/instance/connect/${instanceName}` },
          { method: 'GET', path: `/instance/connect?instance=${encodeURIComponent(instanceName)}` },
          { method: 'GET', path: `/instance/connect?instanceName=${encodeURIComponent(instanceName)}` },
          { method: 'POST', path: `/instance/connect`, body: { instanceName } },
          { method: 'POST', path: `/instance/connect`, body: { instance: instanceName } },
        ]

    const qrCandidates: Array<{ method: 'GET' | 'POST'; path: string; body?: unknown }> = [
      { method: 'GET', path: `/instance/qrcode/${instanceName}` },
      { method: 'GET', path: `/instance/qrCode/${instanceName}` },
      { method: 'GET', path: `/instance/qr/${instanceName}` },
    ]

    let connectState: string | null = null
    let qrBase64: string | null = qrFromCreate
    let pairingCode: string | null = pairingFromCreate

    for (const c of connectCandidates) {
      const res = await evolutionRequestAuto({ baseUrl: apiUrl!, apiKey=[REDACTED]
      if (!res.ok) {
        if (res.status === 404) continue
        const hint = evolutionHint(res.body)
        return jsonResponse(res.status, { error: 'evolution_error', details: res.body, hint })
      }

      connectState = extractInstanceState(res.body) ?? connectState
      qrBase64 = extractQrBase64(res.body) ?? qrBase64
      pairingCode = extractPairingCode(res.body) ?? pairingCode

      if (pairingNumber) {
        if (pairingCode) break
      } else {
        if (qrBase64) break
        if (pairingCode) break
      }
    }

    if (!qrBase64 && !pairingNumber) {
      for (const c of qrCandidates) {
        const res = await evolutionRequestAuto({ baseUrl: apiUrl!, apiKey=[REDACTED]
        if (!res.ok) {
          if (res.status === 404) continue
          const hint = evolutionHint(res.body)
          return jsonResponse(res.status, { error: 'evolution_error', details: res.body, hint })
        }

        connectState = extractInstanceState(res.body) ?? connectState
        qrBase64 = extractQrBase64(res.body) ?? qrBase64
        pairingCode = extractPairingCode(res.body) ?? pairingCode
        if (qrBase64) break
      }
    }

    if (!connectState) {
      const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)
      const stateAfterRes = await evolutionRequestAuto({ baseUrl: apiUrl!, apiKey=[REDACTED]'GET', stopOn404 })
      if (stateAfterRes.ok) connectState = extractInstanceState(stateAfterRes.body)
    }

    const hint = (() => {
      if (pairingNumber && !pairingCode) {
        return 'A Evolution API não retornou código de pareamento para este número. Atualize a Evolution e confirme
suporte a pairing code (Baileys); se não suportar, use “Conectar (QR Code)” em outro aparelho.'
      }
      if (!qrBase64 && !pairingCode && (connectState === 'connecting' || connectState === 'close' || connectState ===
'closed')) {
        return 'A instância ainda não retornou QR Code. Isso costuma acontecer quando a Evolution está iniciando ou
quando a versão do WhatsApp Web no container está desatualizada. Verifique os logs do container e, se necessário,
```

```
atualize CONFIG_SESSION_PHONE_VERSION.'
      }
      return null
    })()
    if (userExtraOk) {
      const { error: updErr } = await dbClient.from('usuarios').update({ whatsapp_instance_name: instanceName
}).eq('id', masterId)
      if (updErr && !isMissingColumnError(updErr.message)) {
        return jsonResponse(400, { error: 'save_instance_name_failed', message: updErr.message })
      }
    }
    const qrSafe = pairingNumber ? null : qrBase64
    return jsonResponse(200, { ok: true, instanceName, state: connectState ?? state, qrBase64: qrSafe, pairingCode, hint
})
  }

  if (payload.action === 'disconnect') {
    const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)

    let logoutRes = await evolutionRequestAuto({ baseUrl: apiUrl!, apiKey=[REDACTED]'DELETE', stopOn404 })
    if (!logoutRes.ok && logoutRes.status === 405) {
      logoutRes = await evolutionRequestAuto({ baseUrl: apiUrl!, apiKey=[REDACTED]'GET', stopOn404 })
    }
    if (!logoutRes.ok && logoutRes.status !== 404) {
      const hint = evolutionHint(logoutRes.body)
      return jsonResponse(logoutRes.status, { error: 'evolution_error', details: logoutRes.body, hint })
    }

    const deleteRes = await evolutionRequestAuto({ baseUrl: apiUrl!, apiKey=[REDACTED]'DELETE', stopOn404 })
    if (!deleteRes.ok && deleteRes.status !== 404) {
      const hint = evolutionHint(deleteRes.body)
      return jsonResponse(deleteRes.status, { error: 'evolution_error', details: deleteRes.body, hint })
    }

    return jsonResponse(200, { ok: true, instanceName })
  }

  if (payload.action === 'send_test') {
    const phoneRaw = String(payload.number ?? '')
    const text = String(payload.text ?? '').trim()
    if (!sanitizePhone(phoneRaw) || !text) return jsonResponse(400, { error: 'invalid_payload' })

    const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)
    let sendRes = await evolutionSendTextWithFallback({ apiUrl: apiUrl!, apiKey=[REDACTED]

    if (!sendRes.ok && sendRes.status === 404 && isEvolutionInstanceNotFound(sendRes.body, instanceName)) {
      const createRes = await createEvolutionInstance({ apiUrl: apiUrl!, apiKey=[REDACTED]
      if (!createRes.ok && createRes.status !== 409) {
        const hint = evolutionHint(createRes.body)
        return jsonResponse(createRes.status, { error: 'evolution_error', details: createRes.body, hint })
      }
      sendRes = await evolutionSendTextWithFallback({ apiUrl: apiUrl!, apiKey=[REDACTED]
    }

    if (!sendRes.ok) {
      const hint = evolutionHint(sendRes.body)
      return jsonResponse(sendRes.status, {
        error: 'evolution_error',
        details: sendRes.body,
        hint,
        attempts: (sendRes as unknown as { attempts?: unknown }).attempts ?? null,
      })
    }
    return jsonResponse(200, { ok: true })
  }

  if (payload.action === 'send_confirmacao') {
    const agendamentoId = String(payload.agendamento_id ?? '').trim()
    if (!agendamentoId) return jsonResponse(400, { error: 'invalid_payload' })

    const selectFull =
```

```
'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,funcionario_id,unidade_id,extras,servico:servicos(n
ome,preco),funcionario:funcionarios(nome_completo,telefone),unidade:unidades(nome,endereco,telefone)'
    const selectBase =
'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,extras,servico:servicos(nome,preco)'

    const first = await userClient.from('agendamentos').select(selectFull).eq('id', agendamentoId).maybeSingle()
    const second =
      first.error && (isMissingTableError(first.error.message) || isMissingColumnError(first.error.message) ||
isMissingRelationshipError(first.error.message))
        ? await userClient.from('agendamentos').select(selectBase).eq('id', agendamentoId).maybeSingle()
        : null

    const ag = (second ? second.data : first.data) as unknown
    const agErr = second ? second.error : first.error

    if (agErr || !ag) return jsonResponse(404, { error: 'agendamento_not_found' })
    const agRow = ag as unknown as AgendamentoRow
    if (agRow.usuario_id !== masterId) return jsonResponse(403, { error: 'not_allowed' })
    const statusNormalized = String(agRow.status ?? '')
      .trim()
      .toLowerCase()
    if (statusNormalized !== 'confirmado') {
      return jsonResponse(400, {
        error: 'agendamento_not_confirmed',
        status: agRow.status,
        hint: 'A confirmação automática só envia quando o status do agendamento está como “confirmado”.',
      })
    }

    const { data: agExtraRaw, error: agExtraErr } = await userClient
      .from('agendamentos')
      .select('confirmacao_enviada')
      .eq('id', agendamentoId)
      .maybeSingle()

    const agExtraOk = !agExtraErr || !isMissingColumnError(agExtraErr.message)
    const agExtraSafe = agExtraOk ? agExtraRaw : null

    const agExtra = (agExtraSafe ?? null) as unknown as AgendamentoConfirmacaoRow | null

    if (agExtra?.confirmacao_enviada === true) return jsonResponse(200, { ok: true, alreadySent: true })

    const shouldSend = userExtra?.enviar_confirmacao !== false
    if (!shouldSend) return jsonResponse(200, { ok: true, skipped: 'disabled' })

    const tmplCandidate = (userExtra?.mensagem_confirmacao ?? '').trim()
    const tmpl = tmplCandidate ? (userExtra?.mensagem_confirmacao ?? '') : defaultConfirmacao

    const clienteEndereco = readExtrasEndereco(agRow.extras)
    const unidadeEndereco = (agRow.unidade?.endereco ?? '').trim()
    const endereco = clienteEndereco || unidadeEndereco || (userBaseRow.endereco ?? '')
    const telefoneProfissional = (agRow.funcionario?.telefone ?? '').trim() || (userBaseRow.telefone ?? '')

    const vars = {
      nome: agRow.cliente_nome ?? '',
      data: formatBRDate(agRow.data),
      hora: agRow.hora_inicio ?? '',
      servico: agRow.servico?.nome ?? '',
      preco: agRow.servico?.preco != null ? formatBRL(Number(agRow.servico.preco)) : '',
      cliente_endereco: clienteEndereco,
      endereco,
      nome_negocio: userBaseRow.nome_negocio ?? '',
      telefone_profissional: telefoneProfissional,
      profissional_nome: agRow.funcionario?.nome_completo ?? '',
      unidade_nome: agRow.unidade?.nome ?? '',
      unidade_endereco: unidadeEndereco,
      unidade_telefone: agRow.unidade?.telefone ?? '',
    }
    const message = interpolateTemplate(tmpl, vars).trim()
    const phoneRaw = String(agRow.cliente_telefone ?? '')
    const phoneOk = Boolean(sanitizePhone(phoneRaw))
    const msgOk = Boolean(message)
```

```
    const clienteEmail = readExtrasEmail(agRow.extras)
    const canEmail = Boolean(resendApiKey && clienteEmail)

    const tryEmail = async (reason: string) => {
      if (!canEmail) return null
      if (!msgOk) return null
      const negocio = (userBaseRow.nome_negocio ?? '').trim() || 'SMagenda'
      const subject = `Agendamento confirmado — ${negocio}`
      const res = await resendSendEmail({ apiKey=[REDACTED]
      if (res.ok && agExtraOk) {
        await userClient
          .from('agendamentos')
          .update({ confirmacao_enviada: true, confirmacao_enviada_em: new Date().toISOString() })
          .eq('id', agendamentoId)
          .eq('usuario_id', masterId)
          .eq('confirmacao_enviada', false)
      }
      return { ok: res.ok, status: res.status, reason, to: clienteEmail, body: res.body }
    }

    const canWhatsapp = Boolean(apiUrl && apiKey)
    if (!msgOk) {
      return jsonResponse(400, {
        error: 'missing_message_data',
        phone_ok: phoneOk,
        message_ok: msgOk,
        phone_masked: phoneRaw ? maskRecipient(phoneRaw) : null,
        hint: 'A mensagem de confirmação ficou vazia. Verifique o modelo de mensagem em Configurações > Mensagens
automáticas.',
      })
    }

    if (!phoneOk) {
      const emailFallback = await tryEmail('invalid_phone')
      if (emailFallback) return jsonResponse(emailFallback.ok ? 200 : 502, { ok: emailFallback.ok, fallback: { email:
emailFallback } })
      return jsonResponse(400, {
        error: 'missing_message_data',
        phone_ok: phoneOk,
        message_ok: msgOk,
        phone_masked: phoneRaw ? maskRecipient(phoneRaw) : null,
        hint: 'O agendamento não tem telefone válido do cliente. Preencha o telefone com DDD (ex.: 11999999999).',
      })
    }

    if (!canWhatsapp) {
      const emailFallback = await tryEmail('whatsapp_not_configured')
      if (emailFallback) return jsonResponse(emailFallback.ok ? 200 : 502, { ok: emailFallback.ok, skipped:
'not_configured', fallback: { email: emailFallback } })
      return jsonResponse(200, { ok: true, skipped: 'not_configured' })
    }

    const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)

    const stateRes = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path:
`/instance/connectionState/${instanceName}`, method: 'GET', stopOn404 })
    const state = stateRes.ok ? extractInstanceState(stateRes.body) : null
    const stateNormalized = (state ?? '').trim().toLowerCase()

    let sendRes = await evolutionSendTextWithFallback({ apiUrl, apiKey, instanceName, phoneRaw, text: message, stopOn404
})

    if (!sendRes.ok && sendRes.status === 404 && isEvolutionInstanceNotFound(sendRes.body, instanceName)) {
      const createRes = await createEvolutionInstance({ apiUrl, apiKey, instanceName })
      if (!createRes.ok && createRes.status !== 409) {
        const hint = evolutionHint(createRes.body)
        return jsonResponse(createRes.status, { error: 'evolution_error', details: createRes.body, hint })
      }
      sendRes = await evolutionSendTextWithFallback({ apiUrl, apiKey, instanceName, phoneRaw, text: message, stopOn404
})
    }
```

```
    if (!sendRes.ok) {
      if (stateNormalized && stateNormalized !== 'open' && stateNormalized !== 'connected') {
        const emailFallback = await tryEmail('instance_not_connected')
        if (emailFallback) {
          return jsonResponse(emailFallback.ok ? 200 : 502, {
            ok: emailFallback.ok,
            error: 'instance_not_connected',
            state,
            instanceName,
            fallback: { email: emailFallback },
          })
        }
        return jsonResponse(409, {
          error: 'instance_not_connected',
          state,
          instanceName,
          hint: 'WhatsApp ainda não está conectado nessa instância. Vá em Configurações > WhatsApp e clique em Conectar
(QR Code) e depois tente novamente.',
        })
      }

      const hint = evolutionHint(sendRes.body)
      const emailFallback = await tryEmail('evolution_error')
      if (emailFallback) {
        return jsonResponse(emailFallback.ok ? 200 : 502, {
          ok: emailFallback.ok,
          error: 'evolution_error',
          details: sendRes.body,
          hint,
          attempts: (sendRes as unknown as { attempts?: unknown }).attempts ?? null,
          fallback: { email: emailFallback },
        })
      }
      return jsonResponse(sendRes.status, {
        error: 'evolution_error',
        details: sendRes.body,
        hint,
        attempts: (sendRes as unknown as { attempts?: unknown }).attempts ?? null,
      })
    }

    if (agExtraOk) {
      const { error: updErr } = await userClient
        .from('agendamentos')
        .update({ confirmacao_enviada: true, confirmacao_enviada_em: new Date().toISOString() })
        .eq('id', agendamentoId)
        .eq('usuario_id', masterId)
        .eq('confirmacao_enviada', false)
      if (updErr && !isMissingColumnError(updErr.message)) {
        return jsonResponse(400, { error: 'save_confirmacao_flag_failed', message: updErr.message })
      }
    }

    return jsonResponse(200, { ok: true, state })
  }

  if (payload.action === 'send_cancelamento') {
    const agendamentoId = String(payload.agendamento_id ?? '').trim()
    if (!agendamentoId) return jsonResponse(400, { error: 'invalid_payload' })

    const selectFull =

'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,funcionario_id,unidade_id,extras,servico:servicos(n
ome,preco),funcionario:funcionarios(nome_completo,telefone),unidade:unidades(nome,endereco,telefone)'
    const selectBase =
'id,usuario_id,cliente_nome,cliente_telefone,data,hora_inicio,status,extras,servico:servicos(nome,preco)'

    const first = await userClient.from('agendamentos').select(selectFull).eq('id', agendamentoId).maybeSingle()
    const second =
      first.error && (isMissingTableError(first.error.message) || isMissingColumnError(first.error.message) ||
isMissingRelationshipError(first.error.message))
```

```
      ? await userClient.from('agendamentos').select(selectBase).eq('id', agendamentoId).maybeSingle()
      : null

    const ag = (second ? second.data : first.data) as unknown
    const agErr = second ? second.error : first.error

    if (agErr || !ag) return jsonResponse(404, { error: 'agendamento_not_found' })
    const agRow = ag as unknown as AgendamentoRow
    if (agRow.usuario_id !== masterId) return jsonResponse(403, { error: 'not_allowed' })

    const statusNormalized = String(agRow.status ?? '')
      .trim()
      .toLowerCase()
    if (statusNormalized !== 'cancelado') {
      return jsonResponse(400, {
        error: 'agendamento_not_cancelled',
        status: agRow.status,
        hint: 'O aviso de cancelamento só envia quando o status do agendamento está como “cancelado”.',
      })
    }

    const shouldSend = (userExtra?.enviar_cancelamento ?? true) !== false
    if (!shouldSend) return jsonResponse(200, { ok: true, skipped: 'disabled' })

    const tmplCandidate = (userExtra?.mensagem_cancelamento ?? '').trim()
    const tmpl = tmplCandidate ? (userExtra?.mensagem_cancelamento ?? '') : defaultCancelamento

    const clienteEndereco = readExtrasEndereco(agRow.extras)
    const unidadeEndereco = (agRow.unidade?.endereco ?? '').trim()
    const endereco = clienteEndereco || unidadeEndereco || (userBaseRow.endereco ?? '')
    const telefoneProfissional = (agRow.funcionario?.telefone ?? '').trim() || (userBaseRow.telefone ?? '')

    const vars = {
      nome: agRow.cliente_nome ?? '',
      data: formatBRDate(agRow.data),
      hora: agRow.hora_inicio ?? '',
      servico: agRow.servico?.nome ?? '',
      preco: agRow.servico?.preco != null ? formatBRL(Number(agRow.servico.preco)) : '',
      cliente_endereco: clienteEndereco,
      endereco,
      nome_negocio: userBaseRow.nome_negocio ?? '',
      telefone_profissional: telefoneProfissional,
      profissional_nome: agRow.funcionario?.nome_completo ?? '',
      unidade_nome: agRow.unidade?.nome ?? '',
      unidade_endereco: unidadeEndereco,
      unidade_telefone: agRow.unidade?.telefone ?? '',
    }
    const message = interpolateTemplate(tmpl, vars).trim()
    const phoneRaw = String(agRow.cliente_telefone ?? '')
    const phoneOk = Boolean(sanitizePhone(phoneRaw))
    const msgOk = Boolean(message)

    const clienteEmail = readExtrasEmail(agRow.extras)
    const canEmail = Boolean(resendApiKey && clienteEmail)

    const tryEmail = async (reason: string) => {
      if (!canEmail) return null
      if (!msgOk) return null
      const negocio = (userBaseRow.nome_negocio ?? '').trim() || 'SMagenda'
      const subject = `Agendamento cancelado — ${negocio}`
      const res = await resendSendEmail({ apiKey=[REDACTED]
      return { ok: res.ok, status: res.status, reason, to: clienteEmail, body: res.body }
    }

    const canWhatsapp = Boolean(apiUrl && apiKey)
    if (!msgOk) {
      return jsonResponse(400, {
        error: 'missing_message_data',
        phone_ok: phoneOk,
        message_ok: msgOk,
        phone_masked: phoneRaw ? maskRecipient(phoneRaw) : null,
        hint: 'A mensagem de cancelamento ficou vazia.',
```

```
      })
    }

    if (!phoneOk) {
      const emailFallback = await tryEmail('invalid_phone')
      if (emailFallback) return jsonResponse(emailFallback.ok ? 200 : 502, { ok: emailFallback.ok, fallback: { email: emailFallback } })
      return jsonResponse(400, {
        error: 'missing_message_data',
        phone_ok: phoneOk,
        message_ok: msgOk,
        phone_masked: phoneRaw ? maskRecipient(phoneRaw) : null,
        hint: 'O agendamento não tem telefone válido do cliente. Preencha o telefone com DDD (ex.: 11999999999).',
      })
    }

    if (!canWhatsapp) {
      const emailFallback = await tryEmail('whatsapp_not_configured')
      if (emailFallback) return jsonResponse(emailFallback.ok ? 200 : 502, { ok: emailFallback.ok, skipped: 'not_configured', fallback: { email: emailFallback } })
      return jsonResponse(200, { ok: true, skipped: 'not_configured' })
    }

    const stopOn404 = ({ body }: { body: unknown }) => isEvolutionInstanceNotFound(body, instanceName)

    const stateRes = await evolutionRequestAuto({ baseUrl: apiUrl, apiKey, path: `/instance/connectionState/${instanceName}`, method: 'GET', stopOn404 })
    const state = stateRes.ok ? extractInstanceState(stateRes.body) : null
    const stateNormalized = (state ?? '').trim().toLowerCase()

    let sendRes = await evolutionSendTextWithFallback({ apiUrl, apiKey, instanceName, phoneRaw, text: message, stopOn404 })

    if (!sendRes.ok && sendRes.status === 404 && isEvolutionInstanceNotFound(sendRes.body, instanceName)) {
      const createRes = await createEvolutionInstance({ apiUrl, apiKey, instanceName })
      if (!createRes.ok && createRes.status !== 409) {
        const hint = evolutionHint(createRes.body)
        return jsonResponse(createRes.status, { error: 'evolution_error', details: createRes.body, hint })
      }
      sendRes = await evolutionSendTextWithFallback({ apiUrl, apiKey, instanceName, phoneRaw, text: message, stopOn404 })
    }

    if (!sendRes.ok) {
      if (stateNormalized && stateNormalized !== 'open' && stateNormalized !== 'connected') {
        const emailFallback = await tryEmail('instance_not_connected')
        if (emailFallback) {
          return jsonResponse(emailFallback.ok ? 200 : 502, {
            ok: emailFallback.ok,
            error: 'instance_not_connected',
            state,
            instanceName,
            fallback: { email: emailFallback },
          })
        }
        return jsonResponse(409, {
          error: 'instance_not_connected',
          state,
          instanceName,
          hint: 'WhatsApp ainda não está conectado nessa instância. Vá em Configurações > WhatsApp e clique em Conectar (QR Code) e depois tente novamente.',
        })
      }

      const hint = evolutionHint(sendRes.body)
      const emailFallback = await tryEmail('evolution_error')
      if (emailFallback) {
        return jsonResponse(emailFallback.ok ? 200 : 502, {
          ok: emailFallback.ok,
          error: 'evolution_error',
          details: sendRes.body,
          hint,
```

```
          attempts: (sendRes as unknown as { attempts?: unknown }).attempts ?? null,
          fallback: { email: emailFallback },
        })
      }
      return jsonResponse(sendRes.status, {
        error: 'evolution_error',
        details: sendRes.body,
        hint,
        attempts: (sendRes as unknown as { attempts?: unknown }).attempts ?? null,
      })
    }

    return jsonResponse(200, { ok: true, state })
  }

  return jsonResponse(400, { error: 'unknown_action' })
})
```

smagenda/tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
export default {
  content: ['./index.html', './src/**/*.{ts,tsx}'],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

smagenda/tsconfig.app.json

```
{
  "compilerOptions": {
    "composite": true,
    "incremental": true,
    "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.app.tsbuildinfo",
    "target": "ES2022",
    "useDefineForClassFields": true,
    "lib": ["ES2022", "DOM", "DOM.Iterable"],
    "module": "ESNext",
    "types": ["vite/client", "react", "react-dom"],
    "skipLibCheck": true,

    /* Bundler mode */
    "moduleResolution": "bundler",
    "allowImportingTsExtensions": true,
    "verbatimModuleSyntax": true,
    "moduleDetection": "force",
    "noEmit": true,
    "jsx": "react-jsx",

    /* Linting */
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noFallthroughCasesInSwitch": true
  },
  "include": ["src"]
}
```

smagenda/tsconfig.json

```
{
  "files": [],
  "references": [
    { "path": "./tsconfig.app.json" },
    { "path": "./tsconfig.node.json" }
  ]
}
```

smagenda/tsconfig.node.json

```
{
  "compilerOptions": {
    "composite": true,
    "incremental": true,
    "tsBuildInfoFile": "./node_modules/.tmp/tsconfig.node.tsbuildinfo",
    "target": "ES2022",
    "lib": ["ES2022"],
    "module": "ESNext",
    "types": ["node"],
    "skipLibCheck": true,

    /* Bundler mode */
    "moduleResolution": "bundler",
    "allowImportingTsExtensions": true,
    "verbatimModuleSyntax": true,
    "moduleDetection": "force",
    "noEmit": true,

    /* Linting */
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noFallthroughCasesInSwitch": true
  },
  "include": ["vite.config.ts"]
}
```

smagenda/vercel.json

```
{
  "rewrites": [{ "source": "/(.*)", "destination": "/index.html" }]
}
```

smagenda/vite.config.ts

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vite.dev/config/
export default defineConfig({
  plugins: [react()],
  build: {
    outDir: '../dist',
    emptyOutDir: true,
  },
})
```

Apêndice B) Fontes brutas (rotas, suporte, termos e privacidade)

smagenda/src/App.tsx

```
import { Suspense, lazy } from 'react'
import { Navigate, Route, Routes } from 'react-router-dom'
import { RequireAuth } from './state/auth/RequireAuth'

const PublicBookingPage = lazy(() => import('./views/public/PublicBookingPage').then((m) => ({ default: m.PublicBookingPage })))
const LandingPage = lazy(() => import('./views/public/LandingPage').then((m) => ({ default: m.LandingPage })))
const TermosPage = lazy(() => import('./views/public/TermosPage').then((m) => ({ default: m.TermosPage })))
const PrivacidadePage = lazy(() => import('./views/public/PrivacidadePage').then((m) => ({ default: m.PrivacidadePage })))
const AjudaPage = lazy(() => import('./views/public/AjudaPage').then((m) => ({ default: m.AjudaPage })))

const LoginPage = lazy(() => import('./views/auth/LoginPage').then((m) => ({ default: m.LoginPage })))
const CadastroPage = lazy(() => import('./views/auth/CadastroPage').then((m) => ({ default: m.CadastroPage })))
const OnboardingPage = lazy(() => import('./views/auth/OnboardingPage').then((m) => ({ default: m.OnboardingPage })))
const ForgotPasswordPage = lazy(() => import('./views/auth/ForgotPasswordPage').then((m) => ({ default: m.ForgotPasswordPage })))
const ResetPasswordPage = lazy(() => import('./views/auth/ResetPasswordPage').then((m) => ({ default: m.ResetPasswordPage })))

const DashboardPage = lazy(() => import('./views/app/DashboardPage').then((m) => ({ default: m.DashboardPage })))
const ServicosPage = lazy(() => import('./views/app/ServicosPage').then((m) => ({ default: m.ServicosPage })))
const FuncionariosPage = lazy(() => import('./views/app/FuncionariosPage').then((m) => ({ default: m.FuncionariosPage })))
const ClientesPage = lazy(() => import('./views/app/ClientesPage').then((m) => ({ default: m.ClientesPage })))
const ClienteDetalhesPage = lazy(() => import('./views/app/ClienteDetalhesPage').then((m) => ({ default: m.ClienteDetalhesPage })))
const RelatoriosPage = lazy(() => import('./views/app/RelatoriosPage').then((m) => ({ default: m.RelatoriosPage })))
const WhatsappSettingsPage = lazy(() => import('./views/app/WhatsappSettingsPage').then((m) => ({ default: m.WhatsappSettingsPage })))
const MensagensSettingsPage = lazy(() => import('./views/app/MensagensSettingsPage').then((m) => ({ default: m.MensagensSettingsPage })))
const PaginaPublicaSettingsPage = lazy(() =>
  import('./views/app/PaginaPublicaSettingsPage').then((m) => ({ default: m.PaginaPublicaSettingsPage }))
)
const FuncionarioAgendaPage = lazy(() => import('./views/app/FuncionarioAgendaPage').then((m) => ({ default: m.FuncionarioAgendaPage })))
const PagamentoPage = lazy(() => import('./views/app/PagamentoPage').then((m) => ({ default: m.PagamentoPage })))

const AdminLoginPage = lazy(() => import('./views/admin/AdminLoginPage').then((m) => ({ default: m.AdminLoginPage })))
const AdminBootstrapPage = lazy(() => import('./views/admin/AdminBootstrapPage').then((m) => ({ default: m.AdminBootstrapPage })))
const AdminDashboardPage = lazy(() => import('./views/admin/AdminDashboardPage').then((m) => ({ default: m.AdminDashboardPage })))
const AdminClientesPage = lazy(() => import('./views/admin/AdminClientesPage').then((m) => ({ default: m.AdminClientesPage })))
const AdminClienteDetalhesPage = lazy(() =>
  import('./views/admin/AdminClienteDetalhesPage').then((m) => ({ default: m.AdminClienteDetalhesPage }))
)
const AdminLogsPage = lazy(() => import('./views/admin/AdminLogsPage').then((m) => ({ default: m.AdminLogsPage })))
const AdminConfiguracoesPage = lazy(() => import('./views/admin/AdminConfiguracoesPage').then((m) => ({ default: m.AdminConfiguracoesPage })))
const AdminWhatsappAvisosPage = lazy(() => import('./views/admin/AdminWhatsappAvisosPage').then((m) => ({ default: m.AdminWhatsappAvisosPage })))

function AppFallback() {
  return (
    <div className="min-h-screen bg-slate-50 flex items-center justify-center px-4">
      <div className="text-sm text-slate-600">Carregando…</div>
    </div>
  )
}

function App() {
  return (
    <Suspense fallback={<AppFallback />}>
      <Routes>
```

```
          <Route path="/" element={<LandingPage />} />

          <Route path="/agendar/:slug/:unidadeSlug" element={<PublicBookingPage />} />
          <Route path="/agendar/:slug" element={<PublicBookingPage />} />

          <Route path="/termos" element={<TermosPage />} />
          <Route path="/privacidade" element={<PrivacidadePage />} />
          <Route path="/ajuda" element={<AjudaPage />} />

          <Route path="/login" element={<LoginPage />} />
          <Route path="/esqueci-senha" element={<ForgotPasswordPage />} />
          <Route path="/resetar-senha" element={<ResetPasswordPage />} />
          <Route path="/cadastro" element={<CadastroPage />} />
          <Route
            path="/onboarding"
            element={
              <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
                <OnboardingPage />
              </RequireAuth>
            }
          />

          <Route
            path="/dashboard"
            element={
              <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
                <DashboardPage />
              </RequireAuth>
            }
          />
          <Route
            path="/servicos"
            element={
              <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
                <ServicosPage />
              </RequireAuth>
            }
          />
          <Route
            path="/clientes"
            element={
              <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
                <ClientesPage />
              </RequireAuth>
            }
          />
          <Route
            path="/clientes/:telefone"
            element={
              <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
                <ClienteDetalhesPage />
              </RequireAuth>
            }
          />
          <Route
            path="/relatorios"
            element={
              <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
                <RelatoriosPage />
              </RequireAuth>
            }
          />
          <Route
            path="/pagamento"
            element={
              <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
                <PagamentoPage />
              </RequireAuth>
            }
          />
          <Route
            path="/funcionarios"
```

```
            element={
              <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
                <FuncionariosPage />
              </RequireAuth>
            }
          />
          <Route
            path="/configuracoes/whatsapp"
            element={
              <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
                <WhatsappSettingsPage />
              </RequireAuth>
            }
          />
          <Route
            path="/configuracoes/mensagens"
            element={
              <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
                <MensagensSettingsPage />
              </RequireAuth>
            }
          />
          <Route
            path="/configuracoes/pagina-publica"
            element={
              <RequireAuth requiredKind="usuario" allowFuncionarioAdmin={true}>
                <PaginaPublicaSettingsPage />
              </RequireAuth>
            }
          />
          <Route
            path="/funcionario/agenda"
            element={
              <RequireAuth requiredKind="funcionario">
                <FuncionarioAgendaPage />
              </RequireAuth>
            }
          />

          {import.meta.env.DEV ? <Route path="/admin/bootstrap" element={<AdminBootstrapPage />} /> : null}
          <Route path="/admin/login" element={<AdminLoginPage />} />
          <Route
            path="/admin/dashboard"
            element={
              <RequireAuth requiredKind="super_admin">
                <AdminDashboardPage />
              </RequireAuth>
            }
          />
          <Route
            path="/admin/clientes"
            element={
              <RequireAuth requiredKind="super_admin">
                <AdminClientesPage />
              </RequireAuth>
            }
          />
          <Route
            path="/admin/clientes/:id"
            element={
              <RequireAuth requiredKind="super_admin">
                <AdminClienteDetalhesPage />
              </RequireAuth>
            }
          />
          <Route
            path="/admin/logs"
            element={
              <RequireAuth requiredKind="super_admin">
                <AdminLogsPage />
              </RequireAuth>
            }
```

```
        />
        <Route
          path="/admin/whatsapp"
          element={
            <RequireAuth requiredKind="super_admin">
              <AdminWhatsappAvisosPage />
            </RequireAuth>
          }
        />
        <Route
          path="/admin/configuracoes"
          element={
            <RequireAuth requiredKind="super_admin">
              <AdminConfiguracoesPage />
            </RequireAuth>
          }
        />

        <Route path="*" element={<Navigate to="/" replace />} />
      </Routes>
    </Suspense>
  )
}

export default App
```

smagenda/src/components/layout/AppShell.tsx

```
import { Link, useLocation, useNavigate } from 'react-router-dom'
import { useAuth } from '../../state/auth/useAuth'
import { getOptionalEnv } from '../../lib/env'
import { Button } from '../ui/Button'

export function AppShell({ children }: { children: React.ReactNode }) {
  const { principal, appPrincipal, impersonation, stopImpersonation, signOut, masterUsuario } = useAuth()
  const location = useLocation()
  const navigate = useNavigate()
  const isFuncionario = appPrincipal?.kind === 'funcionario'
  const isFuncionarioAdmin = appPrincipal?.kind === 'funcionario' && appPrincipal.profile.permissao === 'admin'
  const funcionario = appPrincipal?.kind === 'funcionario' ? appPrincipal.profile : null
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : isFuncionarioAdmin ? masterUsuario : null
  const temaProspector = usuario?.tema_prospector_habilitado === true

  const plano = String(usuario?.plano ?? '').trim().toLowerCase()
  const isProPlus = plano === 'pro' || plano === 'team' || plano === 'enterprise'

  const supportNumber = getOptionalEnv('VITE_SUPPORT_WHATSAPP_NUMBER')
  const canUseWhatsappSupport = Boolean(supportNumber && usuario && isProPlus)

  const nav = isFuncionario && !isFuncionarioAdmin
    ? [
        { to: '/funcionario/agenda', label: 'Minha Agenda' },
        { to: '/ajuda', label: 'Ajuda' },
      ]
    : isFuncionarioAdmin
      ? [
          ...(funcionario?.pode_ver_agenda ? [{ to: '/dashboard', label: 'Agenda' }] : []),
          ...(funcionario?.pode_ver_agenda ? [{ to: '/funcionario/agenda', label: 'Minha Agenda' }] : []),
          ...(funcionario?.pode_gerenciar_servicos ? [{ to: '/servicos', label: 'Meus Serviços' }] : []),
          ...(funcionario?.pode_ver_clientes_de_outros ? [{ to: '/clientes', label: 'Clientes' }] : []),
          ...(isProPlus && funcionario?.pode_ver_financeiro ? [{ to: '/relatorios', label: 'Relatórios' }] : []),
          { to: '/ajuda', label: 'Ajuda' },
        ]
      : [
          { to: '/dashboard', label: 'Agenda' },
          { to: '/servicos', label: 'Meus Serviços' },
          { to: '/clientes', label: 'Clientes' },
          ...(isProPlus ? [{ to: '/relatorios', label: 'Relatórios' }] : []),
          { to: '/pagamento', label: 'Pagamento' },
          { to: '/funcionarios', label: 'Funcionários' },
```

```
          { to: '/configuracoes/whatsapp', label: 'WhatsApp' },
          { to: '/configuracoes/mensagens', label: 'Mensagens Automáticas' },
          { to: '/configuracoes/pagina-publica', label: 'Página Pública' },
          { to: '/ajuda', label: 'Ajuda' },
        ]

  return (
    <div className={['min-h-screen bg-slate-50', temaProspector ? 'theme-prospector' : ''].filter(Boolean).join(' ')}>
      <div className="mx-auto max-w-6xl px-4 py-6">
        <div className="mb-6 flex items-center justify-between">
          <div>
            <div className="text-xs font-semibold tracking-wide text-slate-500">SMagenda</div>
            <div className="text-lg font-semibold text-slate-900">
              {appPrincipal?.kind === 'usuario'
                ? appPrincipal.profile.nome_negocio
                : appPrincipal?.kind === 'funcionario'
                  ? `Olá, ${appPrincipal.profile.nome_completo}`
                  : ''}
            </div>
          </div>
          <div className="flex items-center gap-2">
            {canUseWhatsappSupport ? (
              <a
                className="inline-flex items-center justify-center rounded-lg px-4 py-2 text-sm font-medium transition focus:outline-none focus:ring-2 focus:ring-slate-300 bg-emerald-600 text-white hover:bg-emerald-500"
                href={`https://wa.me/${encodeURIComponent(String(supportNumber))}?text=${encodeURIComponent('Olá! Preciso de suporte no SMagenda.')}`}
                target="_blank"
                rel="noreferrer"
              >
                Suporte WhatsApp
              </a>
            ) : null}
            {principal?.kind === 'super_admin' && impersonation ? (
              <Button
                variant="secondary"
                onClick={() => {
                  stopImpersonation()
                  navigate('/admin/clientes')
                }}
              >
                Voltar ao admin
              </Button>
            ) : null}
            <Button variant="secondary" onClick={() => signOut()}>
              Sair
            </Button>
          </div>
        </div>

        <div className="grid grid-cols-1 gap-6 md:grid-cols-[240px_1fr]">
          <nav className="md:sticky md:top-6 md:h-fit">
            <div className="rounded-xl border border-slate-200 bg-white p-2">
              {nav.map((item) => {
                const active = location.pathname === item.to
                return (
                  <Link
                    key={item.to}
                    to={item.to}
                    className={[
                      'block rounded-lg px-3 py-2 text-sm font-medium',
                      active ? 'bg-slate-900 text-white' : 'text-slate-700 hover:bg-slate-100',
                    ].join(' ')}
                  >
                    {item.label}
                  </Link>
                )
              })}
            </div>
          </nav>

          <main>{children}</main>
```

```
        </div>
      </div>
    </div>
  )
}
```

smagenda/src/views/public/AjudaPage.tsx

```
import { Link } from 'react-router-dom'
import { AppShell } from '../../components/layout/AppShell'
import { getOptionalEnv } from '../../lib/env'
import { useAuth } from '../state/auth/useAuth'

function normalizeWhatsApp(value: string) {
  const digits = value.replace(/\D+/g, '')
  if (!digits) return null
  if (digits.startsWith('55')) return digits
  if (digits.length === 10 || digits.length === 11) return `55${digits}`
  return digits
}

export function AjudaPage() {
  const { appPrincipal } = useAuth()

  const inApp = appPrincipal?.kind === 'usuario' || appPrincipal?.kind === 'funcionario'
  const backTo = inApp ? (appPrincipal?.kind === 'funcionario' ? '/funcionario/agenda' : '/dashboard') : '/login'

  const supportEmail = (getOptionalEnv('VITE_SUPORTE_EMAIL') ?? getOptionalEnv('VITE_SUPPORT_EMAIL') ??
'suporte@smagenda.com').trim()
  const supportWhatsAppRaw = (
    getOptionalEnv('VITE_SUPORTE_WHATSAPP') ??
    getOptionalEnv('VITE_SUPPORT_WHATSAPP_NUMBER') ??
    getOptionalEnv('VITE_SUPPORT_WHATSAPP') ??
    '(31) 9 7518-4428'
  ).trim()
  const supportWhatsAppDigits = supportWhatsAppRaw ? normalizeWhatsApp(supportWhatsAppRaw) : null

  const waLink = supportWhatsAppDigits
    ? `https://wa.me/${supportWhatsAppDigits}?text=${encodeURIComponent('Olá! Preciso de ajuda com o SMagenda.')}`
    : null

  const content = (
    <div className={inApp ? '' : 'min-h-screen bg-slate-50 px-4 py-10'}>
      <div className={inApp ? 'mx-auto w-full max-w-3xl space-y-6' : 'mx-auto w-full max-w-3xl space-y-6'}>
        <div className="rounded-2xl border border-slate-200 bg-white p-6 shadow-sm">
          <div className="space-y-1">
            <div className="text-2xl font-semibold text-slate-900">Ajuda e Suporte</div>
            <div className="text-sm text-slate-600">Contato, termos e dúvidas frequentes.</div>
          </div>

          <div className="mt-6 space-y-4">
            <div className="rounded-xl border border-slate-200 bg-white p-4">
              <div className="text-sm font-semibold text-slate-900">Fale com a gente</div>
              <div className="mt-2 space-y-2 text-sm text-slate-700">
                {waLink ? (
                  <a className="inline-flex items-center gap-2 font-medium text-slate-900 hover:underline" href={waLink}
target="_blank" rel="noreferrer">
                    <span>WhatsApp:</span>
                    <span className="font-semibold text-slate-900">{supportWhatsAppRaw}</span>
                  </a>
                ) : (
                  <div className="text-slate-600">WhatsApp do suporte não configurado.</div>
                )}

                <a className="block font-medium text-slate-900 hover:underline" href={`mailto:${supportEmail}`}>
                  Email: {supportEmail}
                </a>
              </div>
            </div>

```

```
            <div className="rounded-xl border border-slate-200 bg-white p-4">
              <div className="text-sm font-semibold text-slate-900">Documentos</div>
              <div className="mt-2 flex flex-wrap gap-3 text-sm">
                <Link to="/termos" className="font-medium text-slate-900 hover:underline">
                  Termos de Uso
                </Link>
                <Link to="/privacidade" className="font-medium text-slate-900 hover:underline">
                  Política de Privacidade
                </Link>
              </div>
            </div>

            <div className="rounded-xl border border-slate-200 bg-white p-4">
              <div className="text-sm font-semibold text-slate-900">Perguntas frequentes</div>
              <div className="mt-3 space-y-3">
                <details className="rounded-xl border border-slate-200 bg-slate-50 p-3">
                  <summary className="cursor-pointer text-sm font-semibold text-slate-900">Como o cliente agenda pelo link público?</summary>
                  <div className="mt-2 text-sm text-slate-700">
                    Você configura os serviços e horários; depois compartilha o link /agendar/SEU-SLUG. O cliente escolhe serviço, dia e horário e confirma.
                  </div>
                </details>

                <details className="rounded-xl border border-slate-200 bg-slate-50 p-3">
                  <summary className="cursor-pointer text-sm font-semibold text-slate-900">Não estou recebendo email de confirmação do Supabase</summary>
                  <div className="mt-2 text-sm text-slate-700">
                    Verifique spam e configure SMTP em Authentication → SMTP Settings. No painel admin do SMagenda existe uma área de validação do Resend.
                  </div>
                </details>

                <details className="rounded-xl border border-slate-200 bg-slate-50 p-3">
                  <summary className="cursor-pointer text-sm font-semibold text-slate-900">Como funcionam serviços de dia inteiro?</summary>
                  <div className="mt-2 text-sm text-slate-700">
                    Marque o serviço como “dia inteiro” e defina a capacidade diária (1 ou 2). No link público, o cliente escolhe a data em um calendário com dias disponíveis.
                  </div>
                </details>

                <details className="rounded-xl border border-slate-200 bg-slate-50 p-3">
                  <summary className="cursor-pointer text-sm font-semibold text-slate-900">Onde vejo e altero os horários de trabalho?</summary>
                  <div className="mt-2 text-sm text-slate-700">
                    No painel do usuário você configura horários base. Para funcionários, cada profissional pode ajustar seus horários na própria agenda.
                  </div>
                </details>
              </div>
            </div>
          </div>
        </div>

        <div className="text-center text-sm text-slate-600">
          <Link to={backTo} className="hover:underline">
            Voltar
          </Link>
        </div>
      </div>
    </div>
  )

  return inApp ? <AppShell>{content}</AppShell> : content
}
```

smagenda/src/views/public/LandingPage.tsx

```
import { useCallback, useMemo, useState } from 'react'
import { Link, Navigate } from 'react-router-dom'
import { getOptionalEnv, getPublicBrandConfig } from '../../lib/env'
import { useAuth } from '../../state/auth/useAuth'

function normalizeWhatsApp(value: string) {
  const digits = value.replace(/\D+/g, '')
  if (!digits) return null
  if (digits.startsWith('55')) return digits
  if (digits.length === 10 || digits.length === 11) return `55${digits}`
  return digits
}

export function LandingPage() {
  const { appPrincipal, loading } = useAuth()
  const brand = useMemo(() => getPublicBrandConfig(), [])
  const [copyState, setCopyState] = useState<'idle' | 'ok' | 'error'>('idle')

  const supportWhatsAppRaw = (
    getOptionalEnv('VITE_SUPORTE_WHATSAPP') ??
    getOptionalEnv('VITE_SUPPORT_WHATSAPP_NUMBER') ??
    getOptionalEnv('VITE_SUPPORT_WHATSAPP') ??
    ''
  ).trim()
  const supportWhatsAppDigits = supportWhatsAppRaw ? normalizeWhatsApp(supportWhatsAppRaw) : null
  const waLink = supportWhatsAppDigits
    ? `https://wa.me/${supportWhatsAppDigits}?text=${encodeURIComponent('Olá! Quero conhecer os planos do SMagenda.')}`
    : null

  const preSaleUntilLabel = '08/02/2026'
  const isPreSale = useMemo(() => {
    const now = new Date()
    const end = new Date(2026, 1, 8, 23, 59, 59, 999)
    return now.getTime() <= end.getTime()
  }, [])

  const plans = useMemo(
    () =>
      [
        {
          key: 'basic',
          title: 'BASIC',
          priceLabel: 'R$ 34,99/mês',
          subtitle: 'Ideal para começar com 1 profissional',
          bullets: ['Agendamentos 60 por mês', '1 profissional incluído', 'Lembretes automáticos via WhatsApp', 'Até 3 
serviços', 'Página pública personalizável'],
          highlight: false,
        },
        {
          key: 'pro',
          title: 'PRO',
          priceLabel: 'R$ 59,99/mês',
          subtitle: 'Até 6 profissionais (4 inclusos + até 2 adicionais)',
          bullets: ['4 profissionais incluídos', 'Serviços ilimitados', 'Logo e fotos de serviços', 'Relatórios', 
'Bloqueios recorrentes'],
          highlight: true,
        },
        {
          key: 'enterprise',
          title: 'EMPRESA',
          priceLabel: 'R$ 98,99/mês',
          subtitle: 'Até 10 profissionais',
          bullets: ['Até 10 profissionais', 'Multi-unidades', 'Agendamentos ilimitados', 'Serviços ilimitados', 'Para 
mais profissionais, fale com o suporte'],
          highlight: false,
        },
      ] as const,
    []
  )
```

```
  const exampleBookingUrl = useMemo(() => {
    const origin = typeof window !== 'undefined' ? window.location.origin : ''
    return origin ? `${origin}/agendar/seu-negocio` : '/agendar/seu-negocio'
  }, [])

  const copyExampleUrl = useCallback(async () => {
    try {
      if (!navigator?.clipboard?.writeText) {
        setCopyState('error')
        return
      }
      await navigator.clipboard.writeText(exampleBookingUrl)
      setCopyState('ok')
      window.setTimeout(() => setCopyState('idle'), 1400)
    } catch {
      setCopyState('error')
      window.setTimeout(() => setCopyState('idle'), 1800)
    }
  }, [exampleBookingUrl])

  if (!loading && appPrincipal) {
    if (appPrincipal.kind === 'funcionario') return <Navigate to="/funcionario/agenda" replace />
    if (appPrincipal.kind === 'super_admin') return <Navigate to="/admin/dashboard" replace />
    return <Navigate to="/dashboard" replace />
  }

  return (
    <div className="relative min-h-screen overflow-hidden bg-slate-950 text-white">
      <div className="pointer-events-none absolute inset-0">
        <div className="absolute inset-0 bg-gradient-to-b from-indigo-500/20 via-slate-950 to-slate-950" />
        <div className="absolute -top-40 left-1/2 h-[560px] w-[860px] -translate-x-1/2 rounded-full bg-gradient-to-r
from-indigo-500/25 via-cyan-400/15 to-fuchsia-500/20 blur-3xl" />
        <svg
          className="absolute inset-0 h-full w-full opacity-[0.14]"
          viewBox="0 0 1200 800"
          fill="none"
          xmlns="http://www.w3.org/2000/svg"
          preserveAspectRatio="none"
        >
          <defs>
            <pattern id="grid" width="42" height="42" patternUnits="userSpaceOnUse">
              <path d="M 42 0 L 0 0 0 42" stroke="rgba(148,163,184,0.35)" strokeWidth="1" />
            </pattern>
          </defs>
          <rect width="1200" height="800" fill="url(#grid)" />
        </svg>
      </div>

      <div className="relative">
        <div className="mx-auto w-full max-w-6xl px-4">
          <div className="flex items-center justify-between py-6">
            <div className="flex items-center gap-2">
              {brand.companyLogoUrl ? (
                <div className="flex items-center rounded-xl bg-white/5 px-2 py-1 ring-1 ring-white/10">
                  <img src={brand.companyLogoUrl} alt={brand.companyName} className="h-6 w-auto" />
                </div>
              ) : (
                <>
                  <div className="flex h-9 w-9 items-center justify-center rounded-xl bg-white/10 ring-1 ring-white/10">
                    <svg viewBox="0 0 24 24" fill="none" className="h-5 w-5">
                      <path
                        d="M7 7h10M7 12h10M7 17h7"
                        stroke="currentColor"
                        strokeWidth="2"
                        strokeLinecap="round"
                        strokeLinejoin="round"
                      />
                    </svg>
                  </div>
                </>
              )}
              <div>
```

```
                <div className="text-sm font-semibold leading-tight">{brand.productName}</div>
                <div className="text-xs text-white/60 leading-tight">por {brand.companyName}</div>
              </div>
            </div>

            <div className="flex items-center gap-3">
              <Link to="/login" className="hidden text-sm font-semibold text-white/80 hover:text-white sm:inline">
                Entrar
              </Link>
              <Link
                to="/cadastro"
                className="inline-flex h-10 items-center rounded-xl bg-white px-4 text-sm font-semibold text-slate-950"
              >
                Criar conta
              </Link>
            </div>
          </div>

          <div className="grid grid-cols-1 gap-10 pb-14 pt-10 md:grid-cols-2 md:items-center">
            <div className="space-y-6">
              <div className="inline-flex items-center gap-2 rounded-full bg-white/10 px-3 py-1 text-xs font-semibold
text-white/80 ring-1 ring-white/10">
                <span className="h-1.5 w-1.5 rounded-full bg-emerald-400" />
                Agendamentos, WhatsApp e equipe em um só lugar
              </div>

              <div className="text-4xl font-semibold tracking-tight sm:text-5xl">
                Agendamentos online com painel completo
              </div>
              <div className="max-w-xl text-base leading-relaxed text-white/80">
                Link público para clientes agendarem em poucos cliques, agenda com profissionais, serviços e mensagens
automáticas no WhatsApp.
              </div>

              <div className="flex flex-col gap-3 sm:flex-row sm:items-center">
                <Link
                  to="/cadastro"
                  className="inline-flex h-11 items-center justify-center rounded-xl bg-white px-5 text-sm font-semibold
text-slate-950"
                >
                  Criar minha conta
                </Link>
                <a
                  href="#planos"
                  className="inline-flex h-11 items-center justify-center rounded-xl bg-white/10 px-5 text-sm font-
semibold text-white ring-1 ring-white/10"
                >
                  Ver planos
                </a>
                {waLink ? (
                  <a
                    href={waLink}
                    target="_blank"
                    rel="noreferrer"
                    className="inline-flex h-11 items-center justify-center rounded-xl bg-emerald-400/15 px-5 text-sm
font-semibold text-emerald-100 ring-1 ring-emerald-300/20"
                  >
                    Falar no WhatsApp
                  </a>
                ) : null}
              </div>

              <div className="text-xs text-white/60">
                Ao criar conta você concorda com{' '}
                <Link to="/termos" className="font-semibold text-white/80 hover:text-white">
                  Termos
                </Link>{' '}
                e{' '}
                <Link to="/privacidade" className="font-semibold text-white/80 hover:text-white">
                  Privacidade
                </Link>
                .
```

```
              </div>
            </div>

            <div className="relative">
              <div className="absolute -inset-6 rounded-[32px] bg-gradient-to-r from-indigo-500/25 via-cyan-400/15 to-fuchsia-500/20 blur-2xl" />
              <div className="relative rounded-[28px] border border-white/10 bg-slate-900/40 p-4 shadow-2xl">
                <div className="flex items-center justify-between rounded-2xl border border-white/10 bg-slate-950/40 px-4 py-3">
                  <div className="flex items-center gap-2">
                    <div className="h-2.5 w-2.5 rounded-full bg-rose-400/80" />
                    <div className="h-2.5 w-2.5 rounded-full bg-amber-300/80" />
                    <div className="h-2.5 w-2.5 rounded-full bg-emerald-400/80" />
                  </div>
                  <div className="text-xs font-semibold text-white/70">Painel {brand.productName}</div>
                  <div className="h-5 w-16 rounded-lg bg-white/5" />
                </div>

                <div className="mt-4 grid grid-cols-1 gap-3">
                  <div className="grid grid-cols-3 gap-3">
                    {[
                      { title: 'Agendamentos', value: 'Organizados', color: 'bg-cyan-400/20 text-cyan-200 ring-cyan-400/30' },
                      { title: 'Clientes', value: 'Centralizados', color: 'bg-indigo-500/20 text-indigo-200 ring-indigo-400/30' },
                      { title: 'Equipe', value: 'Com permissões', color: 'bg-fuchsia-500/20 text-fuchsia-200 ring-fuchsia-400/30' },
                    ].map((k) => (
                      <div key={k.title} className="rounded-2xl border border-white/10 bg-white/5 p-3">
                        <div className="text-xs font-semibold text-white/70">{k.title}</div>
                        <div className="mt-2">
                          <div className={['inline-flex items-center rounded-full px-2 py-1 text-xs font-semibold ring-1', k.color].join(' ')}>
                            {k.value}
                          </div>
                        </div>
                      </div>
                    ))}
                  </div>

                  <div className="rounded-2xl border border-white/10 bg-white/5 p-4">
                    <div className="flex items-center justify-between">
                      <div>
                        <div className="text-sm font-semibold">Link público de agendamento</div>
                        <div className="mt-1 text-xs text-white/60">Compartilhe com clientes para reservar horários</div>
                      </div>
                      <button
                        type="button"
                        onClick={copyExampleUrl}
                        className="inline-flex h-9 items-center rounded-xl bg-white px-3 text-xs font-semibold text-slate-950"
                      >
                        {copyState === 'ok' ? 'Copiado!' : copyState === 'error' ? 'Falhou' : 'Copiar'}
                      </button>
                    </div>
                    <div className="mt-3 rounded-xl border border-white/10 bg-slate-950/30 px-3 py-2 text-xs text-white/70">
                      Exemplo: {exampleBookingUrl}
                    </div>
                  </div>

                  <div className="grid grid-cols-1 gap-3 sm:grid-cols-2">
                    {[
                      { title: 'Confirmação', desc: 'Mensagem automática após agendar' },
                      { title: 'Lembrete', desc: 'Aviso antes do horário marcado' },
                    ].map((b) => (
                      <div key={b.title} className="rounded-2xl border border-white/10 bg-white/5 p-4">
                        <div className="flex items-center gap-2">
                          <div className="flex h-8 w-8 items-center justify-center rounded-xl bg-white/10 ring-1 ring-white/10">
                            <svg viewBox="0 0 24 24" fill="none" className="h-4 w-4 text-white">
```

```
                        <path
                          d="M8 12l2.5 2.5L16 9"
                          stroke="currentColor"
                          strokeWidth="2"
                          strokeLinecap="round"
                          strokeLinejoin="round"
                        />
                      </svg>
                    </div>
                    <div className="text-sm font-semibold">{b.title}</div>
                  </div>
                  <div className="mt-2 text-xs text-white/65">{b.desc}</div>
                </div>
              ))}
            </div>
          </div>
        </div>
      </div>
    </div>

    <div className="pb-14">
      <div className="grid grid-cols-1 gap-4 md:grid-cols-3">
        {[
          {
            title: 'Link público com cara do seu negócio',
            desc: 'Cores, logo e fundo (conforme plano). Cliente agenda em poucos cliques.',
            icon: (
              <svg viewBox="0 0 24 24" fill="none" className="h-5 w-5">
                <path
                  d="M10.5 13.5L13.5 10.5"
                  stroke="currentColor"
                  strokeWidth="2"
                  strokeLinecap="round"
                />
                <path
                  d="M8 16a4 4 0 010-5.657l1.343-1.343A4 4 0 0115 9"
                  stroke="currentColor"
                  strokeWidth="2"
                  strokeLinecap="round"
                />
                <path
                  d="M16 8a4 4 0 010 5.657l-1.343 1.343A4 4 0 019 15"
                  stroke="currentColor"
                  strokeWidth="2"
                  strokeLinecap="round"
                />
              </svg>
            ),
          },
          {
            title: 'Agenda, serviços e profissionais',
            desc: 'Organize horários, regras por serviço e visibilidade por permissões.',
            icon: (
              <svg viewBox="0 0 24 24" fill="none" className="h-5 w-5">
                <path d="M8 7V5m8 2V5" stroke="currentColor" strokeWidth="2" strokeLinecap="round" />
                <path
                  d="M6 9h12M7 19h10a2 2 0 002-2V8a2 2 0 00-2-2H7a2 2 0 00-2 2v9a2 2 0 002 2z"
                  stroke="currentColor"
                  strokeWidth="2"
                  strokeLinecap="round"
                  strokeLinejoin="round"
                />
              </svg>
            ),
          },
          {
            title: 'Automação no WhatsApp',
            desc: 'Templates de confirmação e lembrete, com variáveis do agendamento.',
            icon: (
              <svg viewBox="0 0 24 24" fill="none" className="h-5 w-5">
                <path
                  d="M21 12a8 8 0 01-8 8H7l-4 2 1.2-3.6A8 8 0 1121 12z"
```

```
                        stroke="currentColor"
                        strokeWidth="2"
                        strokeLinecap="round"
                        strokeLinejoin="round"
                      />
                      <path d="M8 12h8" stroke="currentColor" strokeWidth="2" strokeLinecap="round" />
                      <path d="M8 9h5" stroke="currentColor" strokeWidth="2" strokeLinecap="round" />
                    </svg>
                  ),
                },
              ].map((f) => (
                <div key={f.title} className="rounded-3xl border border-white/10 bg-white/5 p-5">
                  <div className="flex items-start gap-3">
                    <div className="flex h-10 w-10 items-center justify-center rounded-2xl bg-white/10 ring-1 ring-
white/10">
                      {f.icon}
                    </div>
                    <div>
                      <div className="text-sm font-semibold">{f.title}</div>
                      <div className="mt-1 text-sm text-white/70">{f.desc}</div>
                    </div>
                  </div>
                </div>
              ))}
            </div>
          </div>

          <div className="pb-14">
            <div className="rounded-[28px] border border-white/10 bg-white/5 p-6">
              <div className="flex flex-col gap-2 sm:flex-row sm:items-end sm:justify-between">
                <div>
                  <div className="text-xs font-semibold tracking-wide text-white/60">Para quem é</div>
                  <div className="mt-2 text-2xl font-semibold">Feito para negócios com agenda e equipe</div>
                  <div className="mt-2 text-sm text-white/70">Funciona bem para atendimentos com horários, profissionais
e regras por serviço.</div>
                </div>
                <Link
                  to="/cadastro"
                  className="inline-flex h-10 items-center justify-center rounded-xl bg-white/10 px-4 text-sm font-
semibold text-white ring-1 ring-white/10"
                >
                  Começar agora
                </Link>
              </div>

              <div className="mt-6 grid grid-cols-1 gap-4 sm:grid-cols-2 lg:grid-cols-3">
                {[
                  { title: 'Barbearias e salões', desc: 'Serviços com duração fixa, buffers e profissionais.' },
                  { title: 'Clínicas e consultórios', desc: 'Histórico de clientes e lembretes automáticos.' },
                  { title: 'Estúdios e aulas', desc: 'Horários recorrentes e organização por equipe.' },
                  { title: 'Lava-jato', desc: 'Serviços por tempo e automações no WhatsApp.' },
                  { title: 'Serviços em domicílio', desc: 'Link público e regras de antecedência.' },
                  { title: 'Negócios multi-unidade', desc: 'Estrutura para planos com unidades (quando habilitado).' },
                ].map((s) => (
                  <div key={s.title} className="rounded-3xl border border-white/10 bg-slate-950/20 p-5">
                    <div className="text-sm font-semibold">{s.title}</div>
                    <div className="mt-2 text-sm text-white/70">{s.desc}</div>
                  </div>
                ))}
              </div>
            </div>
          </div>

          <div className="pb-14">
            <div className="rounded-[28px] border border-white/10 bg-white/5 p-6">
              <div className="grid grid-cols-1 gap-6 md:grid-cols-3 md:items-start">
                <div>
                  <div className="text-xs font-semibold tracking-wide text-white/60">Diferenciais</div>
                  <div className="mt-2 text-2xl font-semibold">Mais controle, menos trabalho manual</div>
                  <div className="mt-2 text-sm text-white/70">
                    Feito para reduzir no-show, organizar a equipe e padronizar a comunicação com o cliente.
                  </div>
```

```
                </div>
                <div className="md:col-span-2">
                  <div className="grid grid-cols-1 gap-4 sm:grid-cols-2">
                    {[
                      {
                        title: 'Regras por serviço',
                        desc: 'Antecedência, janela máxima, buffers e suporte a serviços de dia inteiro.',
                      },
                      {
                        title: 'Permissões por função',
                        desc: 'Gerente, atendente e profissional com acessos alinhados ao seu fluxo.',
                      },
                      {
                        title: 'WhatsApp com variáveis',
                        desc: 'Confirmação e lembrete com dados reais do agendamento, sem copiar e colar.',
                      },
                      {
                        title: 'Identidade no link público',
                        desc: 'Cores e aparência para passar confiança na hora do cliente agendar (conforme plano).',
                      },
                    ].map((d) => (
                      <div key={d.title} className="rounded-3xl border border-white/10 bg-slate-950/20 p-5">
                        <div className="text-sm font-semibold">{d.title}</div>
                        <div className="mt-2 text-sm text-white/70">{d.desc}</div>
                      </div>
                    ))}
                  </div>
                </div>
              </div>
            </div>
          </div>

          <div className="rounded-[28px] border border-white/10 bg-white/5 p-6">
            <div className="grid grid-cols-1 gap-6 md:grid-cols-3">
              {[
                {
                  step: '1',
                  title: 'Cadastre serviços e horários',
                  desc: 'Defina duração, regras e disponibilidade por profissional.',
                },
                {
                  step: '2',
                  title: 'Compartilhe seu link público',
                  desc: 'O cliente escolhe serviço, dia e horário e confirma.',
                },
                {
                  step: '3',
                  title: 'Envie confirmações e lembretes',
                  desc: 'Templates prontos e personalizáveis no WhatsApp.',
                },
              ].map((s) => (
                <div key={s.step} className="rounded-3xl border border-white/10 bg-slate-950/20 p-5">
                  <div className="flex items-center gap-3">
                    <div className="flex h-9 w-9 items-center justify-center rounded-2xl bg-white text-sm font-semibold
text-slate-950">
                      {s.step}
                    </div>
                    <div>
                      <div className="text-sm font-semibold">{s.title}</div>
                      <div className="mt-1 text-sm text-white/70">{s.desc}</div>
                    </div>
                  </div>
                </div>
              ))}
            </div>
          </div>

          <div id="planos" className="mt-10 rounded-[28px] border border-white/10 bg-white/5 p-6">
            <div className="flex flex-col gap-2 sm:flex-row sm:items-end sm:justify-between">
              <div>
                <div className="text-xs font-semibold tracking-wide text-white/60">Planos</div>
                <div className="mt-2 text-2xl font-semibold">Escolha um plano e comece hoje</div>
```

```
                <div className="mt-2 text-sm text-white/70">Agendamento online + automações no WhatsApp + gestão de equipe.</div>
                {isPreSale ? <div className="mt-3 text-[11px] text-white/60">Valores de pré-venda • válido até {preSaleUntilLabel}.</div> : null}
              </div>

              <div className="flex flex-col gap-3 sm:flex-row sm:items-center">
                <Link
                  to="/cadastro"
                  className="inline-flex h-10 items-center justify-center rounded-xl bg-white px-4 text-sm font-semibold text-slate-950"
                >
                  Criar conta e escolher plano
                </Link>
                {waLink ? (
                  <a
                    href={waLink}
                    target="_blank"
                    rel="noreferrer"
                    className="inline-flex h-10 items-center justify-center rounded-xl bg-white/10 px-4 text-sm font-semibold text-white ring-1 ring-white/10"
                  >
                    Tirar dúvidas no WhatsApp
                  </a>
                ) : null}
              </div>
            </div>

            <div className="mt-6 grid grid-cols-1 gap-4 lg:grid-cols-3">
              {plans.map((p) => (
                <div
                  key={p.key}
                  className={
                    p.highlight
                      ? 'relative rounded-3xl border border-emerald-300/20 bg-emerald-400/10 p-5 ring-1 ring-emerald-300/20'
                      : 'rounded-3xl border border-white/10 bg-slate-950/20 p-5'
                  }
                >
                  {p.highlight ? (
                    <div className="absolute -top-3 left-5 inline-flex items-center rounded-full bg-emerald-300/20 px-3 py-1 text-[11px] font-semibold text-emerald-100 ring-1 ring-emerald-300/30">
                      Mais escolhido
                    </div>
                  ) : null}
                  <div className="flex items-start justify-between gap-4">
                    <div>
                      <div className="text-sm font-semibold">{p.title}</div>
                      <div className="mt-1 text-xs text-white/60">{p.subtitle}</div>
                    </div>
                    <div className="text-right">
                      <div className="text-lg font-semibold">{p.priceLabel}</div>
                      <div className="mt-1 text-[11px] text-white/60">por negócio</div>
                    </div>
                  </div>

                  <div className="mt-4 space-y-2 text-sm text-white/75">
                    {p.bullets.map((b) => (
                      <div key={b} className="flex gap-2">
                        <span className="mt-1 h-1.5 w-1.5 shrink-0 rounded-full bg-white/50" />
                        <span>{b}</span>
                      </div>
                    ))}
                  </div>

                  <div className="mt-5">
                    <Link
                      to="/cadastro"
                      className={
                        p.highlight
                          ? 'inline-flex h-10 w-full items-center justify-center rounded-xl bg-white px-4 text-sm font-semibold text-slate-950'
```

```
                      : 'inline-flex h-10 w-full items-center justify-center rounded-xl bg-white/10 px-4 text-sm
font-semibold text-white ring-1 ring-white/10'
                  }
                >
                  Começar com {p.title}
                </Link>
              </div>
            </div>
          ))}
        </div>

        <div className="mt-6 grid grid-cols-1 gap-4 md:grid-cols-3">
          {[
            { title: 'Checkout em produção', desc: 'Pagamento via PIX (30 dias) ou cartão (assinatura) no Stripe.'
},
            { title: 'Cancelamento simples', desc: 'Você controla sua assinatura dentro do painel.' },
            { title: 'Suporte humano', desc: waLink && supportWhatsAppRaw ? `Atendimento via WhatsApp:
${supportWhatsAppRaw}` : 'Atendimento via WhatsApp e email.' },
          ].map((s) => (
            <div key={s.title} className="rounded-3xl border border-white/10 bg-slate-950/20 p-5">
              <div className="text-sm font-semibold">{s.title}</div>
              <div className="mt-2 text-sm text-white/70">{s.desc}</div>
            </div>
          ))}
        </div>
      </div>

      <div className="mt-10 rounded-[28px] border border-white/10 bg-white/5 p-6">
        <div className="flex flex-col gap-2 sm:flex-row sm:items-end sm:justify-between">
          <div>
            <div className="text-xs font-semibold tracking-wide text-white/60">Dúvidas frequentes</div>
            <div className="mt-2 text-2xl font-semibold">Perguntas comuns antes de começar</div>
          </div>
          <Link to="/ajuda" className="text-sm font-semibold text-white/80 hover:text-white">
            Ver ajuda
          </Link>
        </div>

        <div className="mt-6 grid grid-cols-1 gap-4 md:grid-cols-2">
          {[
            {
              q: 'O cliente precisa instalar aplicativo?',
              a: 'Não. Ele agenda pelo link público no navegador (celular ou PC).',
            },
            {
              q: 'Como funciona a cobrança?',
              a: 'No painel, você escolhe PIX (30 dias) ou cartão (assinatura). O checkout abre em produção via
Stripe.',
            },
            {
              q: 'Consigo reduzir faltas (no-show)?',
              a: 'Sim. Use confirmações e lembretes automáticos no WhatsApp para padronizar o atendimento.',
            },
            {
              q: 'Dá para ter equipe com acessos diferentes?',
              a: 'Sim. Permissões por função para organizar rotina e evitar alterações indevidas.',
            },
            {
              q: 'Meu link pode ter minha identidade?',
              a: 'Sim. Aparência da página pública com cores e, conforme plano, logo e fundo.',
            },
            {
              q: 'O que muda entre BASIC, PRO e EMPRESA?',
              a: 'Principalmente limites (agendamentos e profissionais) e recursos como relatórios, multi-unidades e
personalização completa.',
            },
          ].map((f) => (
            <div key={f.q} className="rounded-3xl border border-white/10 bg-slate-950/20 p-5">
              <div className="text-sm font-semibold">{f.q}</div>
              <div className="mt-2 text-sm text-white/70">{f.a}</div>
            </div>
          ))}
```

```
            </div>
          </div>

          <div className="mt-10 rounded-[28px] border border-white/10 bg-white/5 p-6">
            <div className="flex flex-col gap-4 sm:flex-row sm:items-center sm:justify-between">
              <div>
                <div className="text-lg font-semibold">Quer ver o {brand.productName} rodando?</div>
                <div className="mt-1 text-sm text-white/70">Crie sua conta e configure seu link público em minutos.
</div>
              </div>
              <div className="flex gap-3">
                <Link to="/cadastro" className="inline-flex h-11 items-center rounded-xl bg-white px-5 text-sm font-
semibold text-slate-950">
                  Criar conta
                </Link>
                <Link
                  to="/login"
                  className="inline-flex h-11 items-center rounded-xl bg-white/10 px-5 text-sm font-semibold text-white
ring-1 ring-white/10"
                >
                  Entrar
                </Link>
              </div>
            </div>
          </div>

          <div className="py-10">
            <div className="flex flex-wrap items-center justify-center gap-5 text-sm text-white/70">
              <Link to="/ajuda" className="hover:text-white">
                Ajuda
              </Link>
              <Link to="/termos" className="hover:text-white">
                Termos
              </Link>
              <Link to="/privacidade" className="hover:text-white">
                Privacidade
              </Link>
            </div>
            <div className="mt-4 text-center text-xs text-white/40">
              © {new Date().getFullYear()} {brand.companyName} — {brand.productName}
            </div>
          </div>
        </div>
      </div>
    </div>
  )
}
```

smagenda/src/views/public/TermosPage.tsx

```
import { Link } from 'react-router-dom'

const TERMS_VERSION = '2026-01-11'

export function TermosPage() {
  return (
    <div className="min-h-screen bg-slate-50 px-4 py-10">
      <div className="mx-auto w-full max-w-3xl">
        <div className="rounded-2xl border border-slate-200 bg-white p-6 shadow-sm">
          <div className="space-y-1">
            <div className="text-2xl font-semibold text-slate-900">Termos de Uso — SMagenda</div>
            <div className="text-sm text-slate-600">Versão {TERMS_VERSION}</div>
          </div>

          <div className="mt-6 space-y-5 text-sm leading-relaxed text-slate-700">
            <div>
              <div className="font-semibold text-slate-900">1. Aceite</div>
              <div>
                Ao criar uma conta, acessar ou usar o SMagenda (“Plataforma”), você declara que leu e concorda com estes
Termos de
```

```
                Uso e com a Política de Privacidade. Se você não concordar, não utilize a Plataforma.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">2. Quem pode usar</div>
              <div>
                Você deve ter capacidade legal para contratar e fornecer informações verdadeiras. Se você cria a conta
em nome de uma
                empresa, você declara ter poderes para representá-la.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">3. O que a Plataforma faz</div>
              <div>
                O SMagenda é um sistema de gestão de agenda, serviços e clientes, incluindo uma página pública para
agendamentos
                (“Link Público”). A Plataforma não presta os serviços finais ao cliente do seu negócio — você é o
responsável pelo
                atendimento, execução do serviço, preços e políticas do seu estabelecimento.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">4. Conta, acesso e segurança</div>
              <div>
                Você é responsável por manter a confidencialidade das credenciais de acesso e por toda atividade
ocorrida na sua conta.
                Em caso de suspeita de uso indevido, altere sua senha e entre em contato com o suporte.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">5. Conteúdo e dados inseridos</div>
              <div>
                Você mantém a titularidade dos dados inseridos (por exemplo: cadastros, serviços e agendamentos). Você
garante que tem
                base legal para tratar os dados dos seus clientes e profissionais e que cumprirá a legislação aplicável,
inclusive a
                LGPD.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">6. Mensagens e comunicações</div>
              <div>
                A Plataforma pode integrar-se a canais como WhatsApp e e-mail para lembretes e confirmações. Você é o
responsável pelo
                conteúdo das mensagens e por obter consentimentos quando necessário. A disponibilidade de provedores
terceiros pode
                variar.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">7. Planos, pagamento e limitações</div>
              <div>
                Recursos podem variar por plano e podem existir limites de uso (por exemplo: número de funcionários).
Condições de
                cobrança, reajustes e períodos de teste, quando aplicáveis, podem ser informados no momento da
contratação.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">8. Proibições</div>
              <div>
                É proibido: usar a Plataforma para fins ilícitos; tentar explorar vulnerabilidades; interferir no
funcionamento do
                serviço; automatizar acessos sem autorização; ou violar direitos de terceiros.
              </div>
```

```
            </div>

            <div>
              <div className="font-semibold text-slate-900">9. Disponibilidade e suporte</div>
              <div>
                Buscamos alta disponibilidade, mas podem ocorrer indisponibilidades por manutenção, atualizações, falhas
de rede ou
                serviços de terceiros. Poderemos realizar melhorias e mudanças na Plataforma.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">10. Propriedade intelectual</div>
              <div>
                A Plataforma, marcas e softwares associados pertencem ao Single Motion e/ou seus licenciadores. Você não
recebe qualquer
                direito de propriedade sobre a Plataforma.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">11. Rescisão e encerramento</div>
              <div>
                Você pode encerrar sua conta a qualquer momento. Podemos suspender ou encerrar o acesso em caso de
violação destes
                Termos, exigências legais, segurança ou uso abusivo. Podemos manter registros quando exigidos por lei.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">12. Limitação de responsabilidade</div>
              <div>
                Na máxima extensão permitida pela lei, não nos responsabilizamos por perdas indiretas, lucros cessantes
ou danos
                decorrentes do uso do serviço, incluindo falhas de terceiros (por exemplo: internet, provedores de
mensagens, e-mail).
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">13. Alterações</div>
              <div>
                Podemos atualizar estes Termos. Quando houver alteração relevante, poderemos solicitar novo aceite. A
versão vigente
                estará sempre disponível nesta página.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">14. Contato</div>
              <div>
                Para dúvidas, solicite suporte pelos canais informados no painel da Plataforma.
              </div>
            </div>
          </div>

          <div className="mt-8 flex flex-wrap items-center justify-between gap-3 border-t border-slate-100 pt-4 text-
sm">
            <Link to="/privacidade" className="font-medium text-slate-900 hover:underline">
              Ler Política de Privacidade
            </Link>
            <Link to="/cadastro" className="text-slate-600 hover:underline">
              Voltar ao cadastro
            </Link>
          </div>
        </div>
      </div>
    </div>
  )
}
```

smagenda/src/views/public/PrivacidadePage.tsx

```
import { Link } from 'react-router-dom'

const PRIVACY_VERSION = '2026-01-11'

export function PrivacidadePage() {
  return (
    <div className="min-h-screen bg-slate-50 px-4 py-10">
      <div className="mx-auto w-full max-w-3xl">
        <div className="rounded-2xl border border-slate-200 bg-white p-6 shadow-sm">
          <div className="space-y-1">
            <div className="text-2xl font-semibold text-slate-900">Política de Privacidade — SMagenda</div>
            <div className="text-sm text-slate-600">Versão {PRIVACY_VERSION}</div>
          </div>

          <div className="mt-6 space-y-5 text-sm leading-relaxed text-slate-700">
            <div>
              <div className="font-semibold text-slate-900">1. Visão geral</div>
              <div>
                Esta Política explica como o SMagenda trata dados pessoais ao oferecer a Plataforma. Em muitos casos, o
seu negócio é o
                “Controlador” dos dados dos seus clientes, e o SMagenda atua como “Operador”, tratando dados conforme
suas instruções.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">2. Quais dados tratamos</div>
              <div className="space-y-2">
                <div>
                  <span className="font-medium text-slate-900">Dados de conta:</span> nome, e-mail, telefone, nome do
negócio, slug,
                  configurações.
                </div>
                <div>
                  <span className="font-medium text-slate-900">Dados de agendamentos:</span> informações inseridas por
você, como nome e
                  telefone do cliente, datas/horários, serviço, observações e campos extras (ex.: endereço).
                </div>
                <div>
                  <span className="font-medium text-slate-900">Dados técnicos:</span> registros de autenticação e
segurança, e quando
                  disponível, IP e user-agent no registro de aceite.
                </div>
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">3. Finalidades e bases legais</div>
              <div className="space-y-2">
                <div>
                  <span className="font-medium text-slate-900">Prestação do serviço:</span> operar a agenda, criar e
listar agendamentos,
                  autenticação e suporte.
                </div>
                <div>
                  <span className="font-medium text-slate-900">Cumprimento legal e segurança:</span> prevenção a
fraudes, auditoria e
                  manutenção de logs.
                </div>
                <div>
                  <span className="font-medium text-slate-900">Comunicações:</span> envio de confirmações e lembretes
conforme sua
                  configuração e instruções.
                </div>
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">4. Compartilhamento</div>
              <div>
```

```
                Podemos compartilhar dados com provedores necessários para operar o serviço (por exemplo:
infraestrutura, envio de
                e-mails e integrações de mensagens), sempre com finalidade compatível e medidas de segurança. Não
vendemos dados pessoais.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">5. Armazenamento, retenção e segurança</div>
              <div>
                Aplicamos medidas técnicas e organizacionais para proteger dados. Mantemos dados enquanto sua conta
estiver ativa e pelo
                tempo necessário para cumprir obrigações legais, resolver disputas e fazer cumprir acordos.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">6. Direitos do titular (LGPD)</div>
              <div>
                Titulares podem ter direitos como confirmação, acesso, correção e eliminação. Quando o dado pertence aos
seus clientes,
                normalmente o pedido deve ser direcionado ao seu negócio (Controlador). Podemos auxiliar tecnicamente
mediante sua
                solicitação.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">7. Cookies e tecnologias similares</div>
              <div>
                Podemos usar armazenamento local e recursos essenciais para autenticação e experiência do usuário. Caso
usemos cookies
                adicionais, poderemos apresentar aviso/gestão conforme aplicável.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">8. Transferência internacional</div>
              <div>
                Alguns provedores podem processar dados fora do Brasil. Adotamos salvaguardas contratuais e medidas de
segurança quando
                aplicável.
              </div>
            </div>

            <div>
              <div className="font-semibold text-slate-900">9. Alterações desta Política</div>
              <div>
                Podemos atualizar esta Política e, quando necessário, solicitar novo aceite.
              </div>
            </div>
          </div>

          <div className="mt-8 flex flex-wrap items-center justify-between gap-3 border-t border-slate-100 pt-4 text-
sm">
            <Link to="/termos" className="font-medium text-slate-900 hover:underline">
              Ler Termos de Uso
            </Link>
            <Link to="/cadastro" className="text-slate-600 hover:underline">
              Voltar ao cadastro
            </Link>
          </div>
        </div>
      </div>
    </div>
  )
}
```

smagenda/src/views/public/PublicBookingPage.tsx

```
import { useCallback, useEffect, useMemo, useRef, useState } from 'react'
import { useParams } from 'react-router-dom'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { formatBRMoney } from '../../lib/dates'
import { supabase, supabaseEnv } from '../../lib/supabase'

type UsuarioPublico = {
  id: string
  nome_negocio: string
  logo_url: string | null
  tipo_negocio?: string | null
  endereco: string | null
  telefone: string | null
  instagram_url?: string | null
  horario_inicio: string | null
  horario_fim: string | null
  dias_trabalho: number[] | null
  intervalo_inicio: string | null
  intervalo_fim: string | null
  ativo: boolean
  tipo_conta: 'master' | 'individual'
  plano: 'free' | 'basic' | 'pro' | 'team' | 'enterprise'
  public_primary_color: string | null
  public_background_color: string | null
  public_use_background_image: boolean | null
  public_background_image_url: string | null
  unidade_id?: string | null
  unidade_nome?: string | null
  unidade_slug?: string | null
}

function coerceHexColor(value: string | null | undefined, fallback: string) {
  const v = (value ?? '').trim()
  if (!v) return fallback
  if (!/^#(?:[0-9a-fA-F]{3}|[0-9a-fA-F]{6})$/.test(v)) return fallback
  return v
}

type Servico = {
  id: string
  nome: string
  descricao?: string | null
  duracao_minutos: number
  buffer_antes_min?: number
  buffer_depois_min?: number
  antecedencia_minutos?: number
  janela_max_dias?: number
  dia_inteiro?: boolean
  preco: number
  taxa_agendamento: number
  cor: string | null
  foto_url: string | null
}

function startOfMonth(d: Date) {
  return new Date(d.getFullYear(), d.getMonth(), 1)
}

function endOfMonth(d: Date) {
  return new Date(d.getFullYear(), d.getMonth() + 1, 0)
}

function addMonths(d: Date, months: number) {
  return new Date(d.getFullYear(), d.getMonth() + months, 1)
}

function addYears(d: Date, years: number) {
```

```
  return new Date(d.getFullYear() + years, d.getMonth(), d.getDate())
}

function monthRangeIsos(monthIso: string, minDate: Date, maxDate: Date) {
  const base = monthIso ? new Date(`${monthIso}T00:00:00`) : null
  if (!base || !Number.isFinite(base.getTime())) return [] as string[]

  const mStart = startOfMonth(base)
  const mEnd = endOfMonth(base)
  const start = mStart < minDate ? minDate : mStart
  const end = mEnd > maxDate ? maxDate : mEnd

  const out: string[] = []
  const cur = new Date(start)
  cur.setHours(0, 0, 0, 0)
  const last = new Date(end)
  last.setHours(0, 0, 0, 0)

  while (cur <= last) {
    out.push(toIsoDateLocal(cur))
    cur.setDate(cur.getDate() + 1)
  }
  return out
}

function monthLabelPTBR(d: Date) {
  const months = ['Janeiro', 'Fevereiro', 'Março', 'Abril', 'Maio', 'Junho', 'Julho', 'Agosto', 'Setembro', 'Outubro', 'Novembro', 'Dezembro']
  return `${months[d.getMonth()] ?? ''} ${d.getFullYear()}`.trim()
}

function DiaInteiroCalendar(args: {
  primaryColor: string
  selectedIso: string
  setSelectedIso: (iso: string) => void
  minIso: string
  maxIso: string
  monthIso: string
  setMonthIso: (iso: string) => void
  isDayDisabled: (d: Date) => boolean
  diaInteiroDisponibilidade: Record<string, string>
  calendarioLoading: boolean
}) {
  const minIso = args.minIso ?? ''
  const maxIso = args.maxIso ?? ''

  const minDate = useMemo(() => {
    if (!minIso) return null
    const d = new Date(`${minIso}T00:00:00`)
    return Number.isFinite(d.getTime()) ? d : null
  }, [minIso])

  const maxDate = useMemo(() => {
    if (!maxIso) return null
    const d = new Date(`${maxIso}T00:00:00`)
    return Number.isFinite(d.getTime()) ? d : null
  }, [maxIso])

  const fallbackMonthIso = useMemo(() => {
    const fromSelected = args.selectedIso ? new Date(`${args.selectedIso}T00:00:00`) : null
    const base = fromSelected && Number.isFinite(fromSelected.getTime()) ? fromSelected : minDate
    if (base) return toIsoDateLocal(startOfMonth(base))
    const today = new Date()
    today.setHours(0, 0, 0, 0)
    return toIsoDateLocal(startOfMonth(today))
  }, [args.selectedIso, minDate])

  const monthIso = args.monthIso || fallbackMonthIso

  const calendarMonth = useMemo(() => {
    const base = monthIso ? new Date(`${monthIso}T00:00:00`) : null
    if (!base || !Number.isFinite(base.getTime())) return startOfMonth(new Date())
```

```
    return startOfMonth(base)
  }, [monthIso])

  const calendarGridDays = useMemo(() => {
    const monthStart = startOfMonth(calendarMonth)
    const monthEnd = endOfMonth(calendarMonth)
    const startWeekday = monthStart.getDay()
    const gridStart = addDays(monthStart, -startWeekday)
    const gridDays: Date[] = []
    for (let i = 0; i < 42; i += 1) gridDays.push(addDays(gridStart, i))
    return { monthStart, monthEnd, gridDays }
  }, [calendarMonth])

  const goPrevMonth = () => {
    const prev = addMonths(calendarMonth, -1)
    if (minDate && Number.isFinite(minDate.getTime()) && prev < startOfMonth(minDate)) return
    args.setMonthIso(toIsoDateLocal(prev))
  }

  const goNextMonth = () => {
    const next = addMonths(calendarMonth, 1)
    if (maxDate && Number.isFinite(maxDate.getTime()) && startOfMonth(next) > startOfMonth(maxDate)) return
    args.setMonthIso(toIsoDateLocal(next))
  }

  return (
    <div className="mt-3 space-y-3">
      <div className="flex items-center justify-between gap-2">
        <button
          type="button"
          onClick={goPrevMonth}
          className="rounded-xl border border-slate-200 bg-white px-3 py-2 text-sm font-semibold text-slate-700
hover:bg-slate-50"
        >
          ◀
        </button>
        <div className="text-sm font-semibold text-slate-900">{monthLabelPTBR(calendarMonth)}</div>
        <button
          type="button"
          onClick={goNextMonth}
          className="rounded-xl border border-slate-200 bg-white px-3 py-2 text-sm font-semibold text-slate-700
hover:bg-slate-50"
        >
          ▶
        </button>
      </div>

      <div className="grid grid-cols-7 gap-1">
        {['Dom', 'Seg', 'Ter', 'Qua', 'Qui', 'Sex', 'Sáb'].map((w) => (
          <div key={w} className="px-1 py-1 text-center text-[11px] font-semibold text-slate-500">
            {w}
          </div>
        ))}

        {calendarGridDays.gridDays.map((dObj) => {
          const inMonth = dObj.getMonth() === calendarGridDays.monthStart.getMonth()
          if (!inMonth) {
            return <div key={toIsoDateLocal(dObj)} className="h-10 sm:h-12 md:h-14" />
          }
          const iso = toIsoDateLocal(dObj)
          const disabledByRule = args.isDayDisabled(dObj)
          const slot = args.diaInteiroDisponibilidade[iso] ?? ''
          const disabled = disabledByRule || !slot
          const selected = args.selectedIso === iso && !disabled
          return (
            <button
              key={iso}
              type="button"
              disabled={disabled}
              onClick={() => {
                args.setSelectedIso(iso)
              }}
```

```
              className={[
                'h-10 sm:h-12 md:h-14 rounded-xl border text-sm font-semibold',
                disabled ? 'opacity-40 cursor-not-allowed bg-white border-slate-200 text-slate-500' : 'bg-white border-
slate-200 text-slate-900 hover:bg-slate-50',
              ].join(' ')}
              style={selected ? { backgroundColor: args.primaryColor, borderColor: args.primaryColor, color: '#fff' } :
undefined}
            >
              {dObj.getDate()}
            </button>
          )
        })}
      </div>

      <div className="text-xs text-slate-600">{args.calendarioLoading ? 'Carregando disponibilidade…' : 'Selecione um
dia disponível.'}</div>
    </div>
  )
}

function DatePopupCalendar(args: {
  open: boolean
  primaryColor: string
  selectedIso: string
  monthIso: string
  setMonthIso: (iso: string) => void
  onClose: () => void
  onSelect: (iso: string) => void
  minIso: string
  maxIso: string
  isDayDisabled: (d: Date) => boolean
  dayHasSlots: Record<string, boolean>
  availabilityLoading: boolean
}) {
  const minIso = args.minIso ?? ''
  const maxIso = args.maxIso ?? ''

  const minDate = useMemo(() => {
    if (!minIso) return null
    const d = new Date(`${minIso}T00:00:00`)
    return Number.isFinite(d.getTime()) ? d : null
  }, [minIso])

  const maxDate = useMemo(() => {
    if (!maxIso) return null
    const d = new Date(`${maxIso}T00:00:00`)
    return Number.isFinite(d.getTime()) ? d : null
  }, [maxIso])

  const fallbackMonthIso = useMemo(() => {
    const fromSelected = args.selectedIso ? new Date(`${args.selectedIso}T00:00:00`) : null
    const base = fromSelected && Number.isFinite(fromSelected.getTime()) ? fromSelected : minDate
    if (base) return toIsoDateLocal(startOfMonth(base))
    const today = new Date()
    today.setHours(0, 0, 0, 0)
    return toIsoDateLocal(startOfMonth(today))
  }, [args.selectedIso, minDate])

  const monthIso = args.monthIso || fallbackMonthIso

  const calendarMonth = useMemo(() => {
    const base = monthIso ? new Date(`${monthIso}T00:00:00`) : null
    if (!base || !Number.isFinite(base.getTime())) return startOfMonth(new Date())
    return startOfMonth(base)
  }, [monthIso])

  const calendarGridDays = useMemo(() => {
    const monthStart = startOfMonth(calendarMonth)
    const monthEnd = endOfMonth(calendarMonth)
    const startWeekday = monthStart.getDay()
    const gridStart = addDays(monthStart, -startWeekday)
    const gridDays: Date[] = []
```

```
    for (let i = 0; i < 42; i += 1) gridDays.push(addDays(gridStart, i))
    return { monthStart, monthEnd, gridDays }
  }, [calendarMonth])

  const canPrev = useMemo(() => {
    if (!minDate) return true
    const prevMonth = addMonths(calendarMonth, -1)
    return prevMonth >= startOfMonth(minDate)
  }, [calendarMonth, minDate])

  const canNext = useMemo(() => {
    if (!maxDate) return true
    const nextMonth = addMonths(calendarMonth, 1)
    return nextMonth <= startOfMonth(maxDate)
  }, [calendarMonth, maxDate])

  useEffect(() => {
    if (!args.open) return
    const onKeyDown = (e: KeyboardEvent) => {
      if (e.key === 'Escape') args.onClose()
    }
    window.addEventListener('keydown', onKeyDown)
    const prevOverflow = document.body.style.overflow
    document.body.style.overflow = 'hidden'
    return () => {
      window.removeEventListener('keydown', onKeyDown)
      document.body.style.overflow = prevOverflow
    }
  }, [args])

  if (!args.open) return null

  return (
    <div className="fixed inset-0 z-50">
      <button type="button" className="absolute inset-0 bg-slate-900/40" onClick={args.onClose} />
      <div className="relative h-full w-full flex items-end sm:items-center justify-center p-4">
        <div className="w-full max-w-sm rounded-2xl border border-slate-200 bg-white shadow-xl">
          <div className="p-4 space-y-3">
            <div className="flex items-center justify-between gap-3">
              <button
                type="button"
                onClick={() => {
                  if (!canPrev) return
                  args.setMonthIso(toIsoDateLocal(addMonths(calendarMonth, -1)))
                }}
                disabled={!canPrev}
                className="rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm font-semibold text-slate-900
disabled:opacity-40"
              >
                ‹
              </button>
              <div className="text-sm font-semibold text-slate-900">{monthLabelPTBR(calendarMonth)}</div>
              <button
                type="button"
                onClick={() => {
                  if (!canNext) return
                  args.setMonthIso(toIsoDateLocal(addMonths(calendarMonth, 1)))
                }}
                disabled={!canNext}
                className="rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm font-semibold text-slate-900
disabled:opacity-40"
              >
                ›
              </button>
            </div>

            <div className="grid grid-cols-7 gap-1">
              {['Dom', 'Seg', 'Ter', 'Qua', 'Qui', 'Sex', 'Sáb'].map((w) => (
                <div key={w} className="text-center text-[11px] font-semibold text-slate-500 py-1">
                  {w}
                </div>
              ))}
```

```
            </div>

            <div className="grid grid-cols-7 gap-1">
              {calendarGridDays.gridDays.map((dObj) => {
                const iso = toIsoDateLocal(dObj)
                const inMonth = dObj >= calendarGridDays.monthStart && dObj <= calendarGridDays.monthEnd
                const disabled = !inMonth || args.isDayDisabled(dObj) || args.dayHasSlots[iso] === false
                const selected = args.selectedIso === iso && !disabled
                return (
                  <button
                    key={iso}
                    type="button"
                    disabled={disabled}
                    onClick={() => {
                      if (disabled) return
                      args.onSelect(iso)
                      args.onClose()
                    }}
                    className={[
                      'h-10 rounded-xl border text-sm font-semibold',
                      disabled
                        ? 'opacity-40 cursor-not-allowed bg-white border-slate-200 text-slate-400'
                        : 'bg-white border-slate-200 text-slate-900 hover:bg-slate-50',
                      !inMonth && !selected ? 'opacity-60' : '',
                    ].join(' ')}
                    style={selected ? { backgroundColor: args.primaryColor, borderColor: args.primaryColor, color:
'#fff' } : undefined}
                  >
                    {dObj.getDate()}
                  </button>
                )
              })}
            </div>

            <div className="flex justify-end">
              <Button variant="secondary" onClick={args.onClose}>
                Fechar
              </Button>
            </div>

            <div className="text-xs text-slate-600">{args.availabilityLoading ? 'Carregando disponibilidade…' :
'Selecione um dia disponível.'}</div>
          </div>
        </div>
      </div>
    </div>
  )
}

async function callPublicPaymentsFn(body: Record<string, unknown>) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } as unknown }
  }

  const supabaseUrl = String(supabaseEnv.values.VITE_SUPABASE_URL ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
    .replace(/\/+$/g, '')
  const supabaseAnonKey = String(supabaseEnv.values.VITE_SUPABASE_ANON_KEY ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')

  const fnUrl = `${supabaseUrl}/functions/v1/payments`

  let res: Response
  try {
    res = await fetch(fnUrl, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        apikey=[REDACTED]
      },
```

```
      body: JSON.stringify(body),
    })
  } catch (e: unknown) {
    const msg = e instanceof Error ? e.message : 'Falha de rede'
    return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } as unknown }
  }

  const text = await res.text().catch(() => '')
  let parsed: unknown = null
  try {
    parsed = text ? (JSON.parse(text) as unknown) : null
  } catch {
    parsed = text
  }

  return { ok: res.ok as boolean, status: res.status, body: parsed as unknown }
}

type Funcionario = {
  id: string
  nome_completo: string
  horario_inicio: string | null
  horario_fim: string | null
  dias_trabalho: number[] | null
  intervalo_inicio: string | null
  intervalo_fim: string | null
}

function normalizePhone(value: string) {
  return value.replace(/[^0-9+]/g, '').trim()
}

function normalizeEmail(value: string) {
  const v = String(value ?? '').trim().toLowerCase()
  if (!v) return ''
  const ok = /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(v)
  return ok ? v : ''
}

function resolvePublicLabels(tipoNegocio: string | null | undefined) {
  const key = (tipoNegocio ?? '').trim().toLowerCase()
  if (key === 'lava_jatos') return { servico: 'Lavagem', servicos: 'Lavagens', profissional: 'Lavador', profissionais: 'Lavadores' }
  if (key === 'barbearia') return { servico: 'Serviço', servicos: 'Serviços', profissional: 'Barbeiro', profissionais: 'Barbeiros' }
  if (key === 'salao') return { servico: 'Serviço', servicos: 'Serviços', profissional: 'Profissional', profissionais: 'Profissionais' }
  if (key === 'estetica') return { servico: 'Procedimento', servicos: 'Procedimentos', profissional: 'Especialista', profissionais: 'Especialistas' }
  if (key === 'odontologia') return { servico: 'Procedimento', servicos: 'Procedimentos', profissional: 'Dentista', profissionais: 'Dentistas' }
  if (key === 'manicure') return { servico: 'Serviço', servicos: 'Serviços', profissional: 'Manicure', profissionais: 'Manicures' }
  if (key === 'pilates') return { servico: 'Aula', servicos: 'Aulas', profissional: 'Instrutor', profissionais: 'Instrutores' }
  if (key === 'faxina') return { servico: 'Diária', servicos: 'Diárias', profissional: 'Profissional', profissionais: 'Profissionais' }
  return { servico: 'Serviço', servicos: 'Serviços', profissional: 'Profissional', profissionais: 'Profissionais' }
}

function addDays(d: Date, days: number) {
  const x = new Date(d)
  x.setDate(x.getDate() + days)
  return x
}

function toWhatsappNumber(value: string) {
  const digits = String(value ?? '').replace(/[^0-9]/g, '')
  if (!digits) return null
  if (digits.startsWith('55')) return digits
  if (digits.length === 10 || digits.length === 11) return `55${digits}`
  return digits
```

```
}

function toGoogleMapsLink(address: string) {
  return `https://www.google.com/maps/search/?api=1&query=${encodeURIComponent(address)}`
}

function parseHHMMToMinutes(value: string | null | undefined) {
  const raw = String(value ?? '').trim()
  if (!raw) return null
  const parts = raw.split(':')
  const hh = Number(parts[0])
  const mm = Number(parts[1] ?? 0)
  if (!Number.isFinite(hh) || !Number.isFinite(mm)) return null
  if (hh < 0 || hh > 23 || mm < 0 || mm > 59) return null
  return hh * 60 + mm
}

function normalizeInstagramUrl(value: string) {
  const raw = String(value ?? '').trim()
  if (!raw) return null
  if (raw.startsWith('http://') || raw.startsWith('https://')) return raw
  if (raw.startsWith('@')) return `https://instagram.com/${raw.slice(1)}`
  if (raw.includes('instagram.com/')) return `https://${raw.replace(/^https?:\/\//, '')}`
  return `https://instagram.com/${raw}`
}

function rpcErrText(err: unknown) {
  const e = err && typeof err === 'object' ? (err as Record<string, unknown>) : null
  const msg = typeof e?.message === 'string' ? e.message.trim() : 'Erro'
  const code = typeof e?.code === 'string' ? e.code.trim() : ''
  const details = typeof e?.details === 'string' ? e.details.trim() : ''
  const hint = typeof e?.hint === 'string' ? e.hint.trim() : ''
  const parts = [msg]
  if (code && !msg.includes(code)) parts.push(code)
  if (details && details !== msg) parts.push(details)
  if (hint) parts.push(hint)
  return parts.filter(Boolean).join(' — ')
}

function shouldRetryWithoutUnidade(err: unknown) {
  const lower = rpcErrText(err).toLowerCase()
  const mentionsUnidadeArg = lower.includes('p_unidade_id')
  const looksLikeSignatureMismatch = isSignatureMismatchText(lower)
  return mentionsUnidadeArg && looksLikeSignatureMismatch
}

function shouldRetryWithoutFuncionario(err: unknown) {
  const lower = rpcErrText(err).toLowerCase()
  const mentionsFuncionarioArg = lower.includes('p_funcionario_id')
  const looksLikeSignatureMismatch = isSignatureMismatchText(lower)
  return mentionsFuncionarioArg && looksLikeSignatureMismatch
}

function shouldRetryWithoutExtras(err: unknown) {
  const lower = rpcErrText(err).toLowerCase()
  const mentionsExtrasArg = lower.includes('p_extras')
  const looksLikeSignatureMismatch = isSignatureMismatchText(lower)
  return mentionsExtrasArg && looksLikeSignatureMismatch
}

function isMissingRpcFnText(lower: string, fnName: string) {
  const needle = fnName.toLowerCase()
  if (!lower.includes(needle)) return false
  return lower.includes('could not find the function') || lower.includes('does not exist') || (lower.includes('schema
cache') && lower.includes('could not find'))
}

function isSignatureMismatchText(lower: string) {
  return (
    lower.includes('pgrst202') ||
    lower.includes('pgrst203') ||
    lower.includes('could not find the function') ||
```

```
    lower.includes('does not exist') ||
    lower.includes('could not choose the best candidate function between') ||
    (lower.includes('schema cache') && lower.includes('could not find'))
  )
}

async function callPublicRpc<T>(fnName: string, args: Record<string, unknown>) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, errorText: `Supabase não configurado. Faltando: ${supabaseEnv.missing.join(', ')}` }
  }

  const base = String(supabaseEnv.values.VITE_SUPABASE_URL ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
    .replace(/\/+$/g, '')
  const key = String(supabaseEnv.values.VITE_SUPABASE_ANON_KEY ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
  const url = `${base}/rest/v1/rpc/${fnName}`

  let res: Response
  try {
    res = await fetch(url, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        apikey=[REDACTED]
        Authorization: `Bearer ${key}`,
      },
      body: JSON.stringify(args),
    })
  } catch (e: unknown) {
    return { ok: false as const, errorText: e instanceof Error ? e.message : 'Falha de rede' }
  }

  const raw = await res.text().catch(() => '')
  const parsed: unknown = raw
    ? (() => {
        try {
          return JSON.parse(raw) as unknown
        } catch {
          return raw
        }
      })()
    : null

  if (!res.ok) {
    const obj = parsed && typeof parsed === 'object' ? (parsed as Record<string, unknown>) : null
    const msg =
      (typeof obj?.message === 'string' && obj.message.trim()) ||
      (typeof obj?.error === 'string' && obj.error.trim()) ||
      (typeof obj?.hint === 'string' && obj.hint.trim()) ||
      `HTTP ${res.status}`
    const details = typeof obj?.details === 'string' ? obj.details.trim() : ''
    const code = typeof obj?.code === 'string' ? obj.code.trim() : ''
    const parts = [msg]
    if (code && !msg.includes(code)) parts.push(code)
    if (details && details !== msg) parts.push(details)
    return { ok: false as const, errorText: parts.filter(Boolean).join(' — ') }
  }

  return { ok: true as const, data: parsed as T }
}

async function callPublicRpcWithSignatureFallback<T>(
  fnName: string,
  args: Record<string, unknown>,
  unidadeId: string | null
): Promise<{ ok: true; data: T } | { ok: false; errorText: string }> {
  const first = await callPublicRpc<T>(fnName, args)
  if (first.ok) return first
```

```
  const firstLower = first.errorText.toLowerCase()
  if (!isSignatureMismatchText(firstLower)) return first

  const hasUnidadeKey = Object.prototype.hasOwnProperty.call(args, 'p_unidade_id')
  const hasFuncionarioKey = Object.prototype.hasOwnProperty.call(args, 'p_funcionario_id')
  const hasExtrasKey = Object.prototype.hasOwnProperty.call(args, 'p_extras')

  const stableKey = (o: Record<string, unknown>) => {
    const keys = Object.keys(o).sort()
    const ordered: Record<string, unknown> = {}
    for (const k of keys) ordered[k] = o[k]
    return JSON.stringify(ordered)
  }

  const withoutKey = (o: Record<string, unknown>, key: string) => {
    const next: Record<string, unknown> = { ...o }
    delete (next as Record<string, unknown>)[key]
    return next
  }

  const seen = new Set<string>([stableKey(args)])
  const variants: Record<string, unknown>[] = []

  if (hasUnidadeKey) {
    if (shouldRetryWithoutUnidade({ message: first.errorText })) variants.push(withoutKey(args, 'p_unidade_id'))
  } else {
    variants.push({ ...args, p_unidade_id: unidadeId ?? null })
  }

  if (hasFuncionarioKey) {
    if (shouldRetryWithoutFuncionario({ message: first.errorText })) variants.push(withoutKey(args, 'p_funcionario_id'))
  } else {
    variants.push({ ...args, p_funcionario_id: null })
  }

  if (hasExtrasKey) {
    if (shouldRetryWithoutExtras({ message: first.errorText })) variants.push(withoutKey(args, 'p_extras'))
  }

  if (hasUnidadeKey && hasFuncionarioKey) {
    if (shouldRetryWithoutUnidade({ message: first.errorText }) || shouldRetryWithoutFuncionario({ message:
first.errorText })) {
      variants.push(withoutKey(withoutKey(args, 'p_unidade_id'), 'p_funcionario_id'))
    }
  }

  for (const v of variants) {
    const key = stableKey(v)
    if (seen.has(key)) continue
    seen.add(key)
    const res = await callPublicRpc<T>(fnName, v)
    if (res.ok) return res
  }

  return first
}

function toIsoDateLocal(d: Date) {
  const x = new Date(d)
  x.setHours(0, 0, 0, 0)
  const yyyy = x.getFullYear()
  const mm = String(x.getMonth() + 1).padStart(2, '0')
  const dd = String(x.getDate()).padStart(2, '0')
  return `${yyyy}-${mm}-${dd}`
}

function formatIsoDateBR(iso: string) {
  const [yyyy, mm, dd] = iso.split('-')
  if (!yyyy || !mm || !dd) return iso
  return `${dd}/${mm}/${yyyy}`
}
```

```
export function PublicBookingPage() {
  const { slug, unidadeSlug } = useParams()

  const enableTaxaAgendamento = (import.meta.env.VITE_ENABLE_TAXA_AGENDAMENTO as string | undefined) === '1'

  const debugEnabled = useMemo(() => {
    try {
      return new URLSearchParams(window.location.search).get('debug') === '1'
    } catch {
      return false
    }
  }, [])

  const [usuario, setUsuario] = useState<UsuarioPublico | null>(null)
  const [servicos, setServicos] = useState<Servico[]>([])
  const [funcionarios, setFuncionarios] = useState<Funcionario[]>([])
  const [loading, setLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)

  const [servicoId, setServicoId] = useState('')
  const [funcionarioId, setFuncionarioId] = useState('')
  const [data, setData] = useState('')
  const [hora, setHora] = useState('')
  const [clienteNome, setClienteNome] = useState('')
  const [clienteTelefone, setClienteTelefone] = useState('')
  const [clienteEmail, setClienteEmail] = useState('')
  const [clientePlaca, setClientePlaca] = useState('')
  const [clienteEndereco, setClienteEndereco] = useState('')
  const [submitting, setSubmitting] = useState(false)
  const [successId, setSuccessId] = useState<string | null>(null)
  const [logoFailed, setLogoFailed] = useState(false)

  const [syncingPayment, setSyncingPayment] = useState(() => {
    try {
      const qp = new URLSearchParams(window.location.search)
      const paid = qp.get('paid') === '1'
      const sessionId = (qp.get('session_id') ?? '').trim()
      return Boolean(enableTaxaAgendamento && paid && sessionId)
    } catch {
      return false
    }
  })

  const [modalOpen, setModalOpen] = useState(false)

  const [datePopupOpen, setDatePopupOpen] = useState(false)
  const [datePopupMonthIso, setDatePopupMonthIso] = useState('')
  const [diaInteiroMonthIso, setDiaInteiroMonthIso] = useState('')

  const [step, setStep] = useState<'servico' | 'profissional' | 'quando'>('servico')

  const servicoRef = useRef<HTMLDivElement | null>(null)
  const profissionalRef = useRef<HTMLDivElement | null>(null)
  const whenRef = useRef<HTMLDivElement | null>(null)

  const canChooseProfessional = useMemo(() => {
    const p = String(usuario?.plano ?? '').trim().toLowerCase()
    return p === 'pro' || p === 'team' || p === 'enterprise'
  }, [usuario?.plano])

  const labels = useMemo(() => resolvePublicLabels(usuario?.tipo_negocio ?? null), [usuario?.tipo_negocio])

  const needsPlaca = useMemo(() => String(usuario?.tipo_negocio ?? '').trim().toLowerCase() === 'lava_jatos',
[usuario?.tipo_negocio])

  const needsEndereco = useMemo(() => String(usuario?.tipo_negocio ?? '').trim().toLowerCase() === 'faxina',
[usuario?.tipo_negocio])

  const hasStaff = useMemo(() => canChooseProfessional && funcionarios.length > 0, [canChooseProfessional,
funcionarios.length])

  const primaryColor = useMemo(() => coerceHexColor(usuario?.public_primary_color ?? null, '#0f172a'),
```

```
[usuario?.public_primary_color])
  const backgroundColor = useMemo(
    () => coerceHexColor(usuario?.public_background_color ?? null, '#f8fafc'),
    [usuario?.public_background_color]
  )
  const backgroundImageUrl = (usuario?.public_background_image_url ?? '').trim()
  const shouldUseBgImage = Boolean(usuario?.public_use_background_image) && Boolean(backgroundImageUrl)
  const hasBgImage = Boolean(shouldUseBgImage)

  useEffect(() => {
    const run = async () => {
      if (!slug) {
        setError('Link inválido')
        setLoading(false)
        return
      }
      setLoading(true)
      setError(null)
      setSuccessId(null)

      const wantsUnidade = Boolean((unidadeSlug ?? '').trim())
      const userArgs: Record<string, unknown> = { p_slug: slug }
      if (wantsUnidade) userArgs.p_unidade_slug = unidadeSlug
      const { data: userData, error: userErr } = await supabase.rpc('public_get_usuario_publico',
userArgs).maybeSingle()
      if (userErr) {
        const msg = userErr.message
        const lower = msg.toLowerCase()
        const missingFn = isMissingRpcFnText(lower, 'public_get_usuario_publico')
        setError(
          missingFn
            ? wantsUnidade
              ? 'Configuração do Supabase incompleta: crie o SQL do link público e o SQL de Multi-unidades (EMPRESA).'
              : 'Configuração do Supabase incompleta: crie o SQL do link público (listar + agendar).'
            : msg
        )
        setLoading(false)
        return
      }
      if (!userData) {
        setError('Página não encontrada')
        setLoading(false)
        return
      }
      const usuarioPublico = userData as unknown as UsuarioPublico
      if ((unidadeSlug ?? '').trim() && !usuarioPublico.unidade_id) {
        setError('Unidade não encontrada')
        setLoading(false)
        return
      }
      setUsuario(usuarioPublico)

      const unidadeId = usuarioPublico.unidade_id ?? null

      const { data: servicesData, error: servicesErr } = await supabase.rpc('public_get_servicos_publicos', {
p_usuario_id: usuarioPublico.id })
      if (servicesErr) {
        const msg = servicesErr.message
        const lower = msg.toLowerCase()
        const missingFn = isMissingRpcFnText(lower, 'public_get_servicos_publicos')
        setError(missingFn ? 'Configuração do Supabase incompleta: crie o SQL do link público (listar + agendar).' :
msg)
        setLoading(false)
        return
      }
      const serviceList = (servicesData ?? []) as unknown as Servico[]
      setServicos(serviceList)

      const staffArgs: Record<string, unknown> = { p_usuario_master_id: usuarioPublico.id }
      if (unidadeId) staffArgs.p_unidade_id = unidadeId
      const { data: staffData, error: staffErr } = await supabase.rpc('public_get_funcionarios_publicos', staffArgs)
      if (staffErr) {
```

```
        const msg = staffErr.message
        const lower = msg.toLowerCase()
        const missingFn = isMissingRpcFnText(lower, 'public_get_funcionarios_publicos')
        setError(missingFn ? 'Configuração do Supabase incompleta: crie o SQL do link público (listar + agendar).' :
msg)
        setLoading(false)
        return
      }
      const staffList = (staffData ?? []) as unknown as Funcionario[]
      setFuncionarios(staffList)
      setLoading(false)
    }

    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar')
      setLoading(false)
    })
  }, [slug, unidadeSlug])

  const selectedServico = useMemo(() => servicos.find((s) => s.id === servicoId) ?? null, [servicos, servicoId])
  const selectedFuncionario = useMemo(() => funcionarios.find((f) => f.id === funcionarioId) ?? null, [funcionarios,
funcionarioId])

  const isDiaInteiro = selectedServico?.dia_inteiro === true

  const usuarioId = usuario?.id ?? null

  const minLeadMinutes = useMemo(() => {
    const v = selectedServico?.antecedencia_minutos
    return typeof v === 'number' && Number.isFinite(v) && v >= 0 ? Math.floor(v) : 120
  }, [selectedServico?.antecedencia_minutos])

  const [availableSlots, setAvailableSlots] = useState<string[]>([])
  const [slotsLoading, setSlotsLoading] = useState(false)

  const [debugSlotsText, setDebugSlotsText] = useState<string | null>(null)

  const today = useMemo(() => {
    const d = new Date()
    d.setHours(0, 0, 0, 0)
    return d
  }, [])

  const maxDate = useMemo(() => {
    const d = addYears(today, 1)
    d.setHours(0, 0, 0, 0)
    return d
  }, [today])

  const maxDays = useMemo(() => {
    const diff = maxDate.getTime() - today.getTime()
    if (!Number.isFinite(diff) || diff <= 0) return 0
    return Math.max(0, Math.round(diff / (24 * 60 * 60 * 1000)))
  }, [maxDate, today])

  const allowedWeekdays = useMemo(() => {
    const base = usuario?.dias_trabalho ?? null
    if (!base || base.length === 0) return null
    return new Set(base)
  }, [usuario?.dias_trabalho])

  const isDayDisabledByRule = useCallback(
    (d: Date) => {
      const x = new Date(d)
      x.setHours(0, 0, 0, 0)
      if (x < today) return true
      if (x > maxDate) return true
      if (allowedWeekdays && !allowedWeekdays.has(x.getDay())) return true
      return false
    },
    [allowedWeekdays, maxDate, today]
  )
```

```
  const [diaInteiroDisponibilidade, setDiaInteiroDisponibilidade] = useState<Record<string, string>>({})
  const [calendarDayHasSlots, setCalendarDayHasSlots] = useState<Record<string, boolean>>({})
  const [calendarioLoading, setCalendarioLoading] = useState(false)

  const minIso = useMemo(() => toIsoDateLocal(today), [today])
  const maxIso = useMemo(() => toIsoDateLocal(maxDate), [maxDate])

  const activeCalendarMonthIso = useMemo(() => {
    if (!usuarioId || !selectedServico) return ''
    if (hasStaff && !funcionarioId) return ''
    if (step !== 'quando') return ''
    if (isDiaInteiro) {
      if (diaInteiroMonthIso) return diaInteiroMonthIso
      const base = data ? new Date(`${data}T00:00:00`) : null
      const safe = base && Number.isFinite(base.getTime()) ? base : today
      return toIsoDateLocal(startOfMonth(safe))
    }
    if (datePopupOpen) return datePopupMonthIso || toIsoDateLocal(startOfMonth(today))
    return ''
  }, [data, datePopupMonthIso, datePopupOpen, diaInteiroMonthIso, funcionarioId, hasStaff, isDiaInteiro,
selectedServico, step, today, usuarioId])

  const calendarFetchIsos = useMemo(() => {
    if (!activeCalendarMonthIso) return [] as string[]
    return monthRangeIsos(activeCalendarMonthIso, today, maxDate)
  }, [activeCalendarMonthIso, maxDate, today])

  useEffect(() => {
    let alive = true
    const run = async () => {
      if (!usuarioId || !selectedServico) {
        setCalendarioLoading(false)
        return
      }
      if (hasStaff && !funcionarioId) {
        setCalendarioLoading(false)
        return
      }
      if (calendarFetchIsos.length === 0) {
        setCalendarioLoading(false)
        return
      }
      setDiaInteiroDisponibilidade({})
      setCalendarDayHasSlots({})
      setCalendarioLoading(true)
      const entries = await Promise.all(
        calendarFetchIsos.map(async (iso) => {
          const dObj = new Date(`${iso}T00:00:00`)
          if (!Number.isFinite(dObj.getTime()) || isDayDisabledByRule(dObj)) return null

          const slotsArgs: Record<string, unknown> = {
            p_usuario_id: usuarioId,
            p_data: iso,
            p_servico_id: selectedServico.id,
            p_funcionario_id: hasStaff ? funcionarioId : null,
          }
          if (usuario?.unidade_id) slotsArgs.p_unidade_id = usuario.unidade_id

          const slotsRes = await callPublicRpcWithSignatureFallback<unknown>('public_get_slots_publicos', slotsArgs,
usuario?.unidade_id ?? null)
          if (!slotsRes.ok) return null

          const raw = (slotsRes.data ?? []) as unknown
          const rows = Array.isArray(raw) ? raw : []
          const list = rows
            .map((r) => {
              if (typeof r === 'string') return r.trim()
              if (!r || typeof r !== 'object') return ''
              const obj = r as Record<string, unknown>
              if (typeof obj.hora_inicio === 'string') return obj.hora_inicio.trim()
              if (typeof obj.hora === 'string') return obj.hora.trim()
```

```
            return ''
          })
          .filter(Boolean)
        const slot = list[0] ?? ''
        return { iso, has: Boolean(slot), slot }
      })
    )

    if (!alive) return
    const nextHasSlots: Record<string, boolean> = {}
    for (const iso of calendarFetchIsos) nextHasSlots[iso] = false
    const next: Record<string, string> = {}
    for (const e of entries) {
      if (!e) continue
      nextHasSlots[e.iso] = e.has
      if (e.has && e.slot) next[e.iso] = e.slot
    }
    setCalendarDayHasSlots(nextHasSlots)
    setDiaInteiroDisponibilidade(isDiaInteiro ? next : {})
    setCalendarioLoading(false)
  }
  run().catch(() => {
    if (!alive) return
    setCalendarioLoading(false)
  })
  return () => {
    alive = false
  }
}, [calendarFetchIsos, funcionarioId, hasStaff, isDayDisabledByRule, isDiaInteiro, selectedServico,
usuario?.unidade_id, usuarioId])

useEffect(() => {
  if (!usuario) return
  if (!data) return
  const current = new Date(`${data}T00:00:00`)
  if (!Number.isFinite(current.getTime())) return
  if (!isDayDisabledByRule(current)) return
  setTimeout(() => {
    setHora('')
    setData('')
  }, 0)
}, [data, isDayDisabledByRule, today, usuario])

useEffect(() => {
  let alive = true
  const run = async () => {
    if (!usuarioId || !selectedServico || !data) {
      setAvailableSlots([])
      setSlotsLoading(false)
      if (debugEnabled) setDebugSlotsText(JSON.stringify({ ok: false, reason: 'missing_payload', usuarioId, servicoId:
selectedServico?.id ?? null, data }, null, 2))
      return
    }
    if (hasStaff && !funcionarioId) {
      setAvailableSlots([])
      setSlotsLoading(false)
      if (debugEnabled) setDebugSlotsText(JSON.stringify({ ok: false, reason: 'missing_funcionario', usuarioId, data,
servicoId: selectedServico.id }, null, 2))
      return
    }

    setSlotsLoading(true)
    setError(null)

    const slotsArgs: Record<string, unknown> = {
      p_usuario_id: usuarioId,
      p_data: data,
      p_servico_id: selectedServico.id,
      p_funcionario_id: hasStaff ? funcionarioId : null,
    }
    if (usuario?.unidade_id) slotsArgs.p_unidade_id = usuario.unidade_id
```

```
      const slotsRes = await callPublicRpcWithSignatureFallback<unknown>('public_get_slots_publicos', slotsArgs,
usuario?.unidade_id ?? null)
      if (!alive) return
      if (!slotsRes.ok) {
        const msg = slotsRes.errorText
        const lower = msg.toLowerCase()
        const missingSlotsFn = lower.includes('public_get_slots_publicos') && (lower.includes('function') ||
lower.includes('rpc'))
        const missingOcupacoesFn = lower.includes('public_get_ocupacoes') && isSignatureMismatchText(lower)
        if (missingSlotsFn) {
          setError(
            (unidadeSlug ?? '').trim()
              ? 'Configuração do Supabase incompleta: atualize o SQL do link público e o SQL de Multi-unidades
(EMPRESA).'
              : 'Configuração do Supabase incompleta: atualize o SQL do link público (listar + agendar).'
          )
        } else if (missingOcupacoesFn) {
          setError('Configuração do Supabase incompleta: crie o SQL de horários públicos (ocupações + bloqueios).')
        } else if (lower.includes('data_passada')) {
          setError('Selecione uma data futura.')
        } else if (lower.includes('data_muito_futura')) {
          setError('Selecione uma data dentro de 1 ano.')
        } else if (lower.includes('fora_do_dia_de_trabalho')) {
          setError('Esse dia não está disponível para agendamento.')
        } else if (lower.includes('horarios_nao_configurados')) {
          setError('Horários de trabalho não configurados.')
        } else {
          setError(msg)
        }
        setSlotsLoading(false)
        if (debugEnabled) {
          const current = new Date(`${data}T00:00:00`)
          const schedStartMin = parseHHMMToMinutes(usuario?.horario_inicio ?? null)
          const schedEndMin = parseHHMMToMinutes(usuario?.horario_fim ?? null)
          const ivStartMin = parseHHMMToMinutes(usuario?.intervalo_inicio ?? null)
          const ivEndMin = parseHHMMToMinutes(usuario?.intervalo_fim ?? null)
          const schedSpanMin = schedStartMin !== null && schedEndMin !== null ? schedEndMin - schedStartMin : null
          const duracaoMin = selectedServico?.duracao_minutos ?? null
          const snapshot = {
            ok: false,
            error: msg,
            env: supabaseEnv.ok ? { url: supabaseEnv.values.VITE_SUPABASE_URL, anonKeyPrefix:
`${supabaseEnv.values.VITE_SUPABASE_ANON_KEY.slice(0, 12)}…` } : { missing: supabaseEnv.missing },
            route: { slug: slug ?? null, unidadeSlug: unidadeSlug ?? null },
            payload: slotsArgs,
            selected: {
              servico: selectedServico ? { id: selectedServico.id, duracao_minutos: selectedServico.duracao_minutos,
nome: selectedServico.nome } : null,
              funcionario: hasStaff ? (selectedFuncionario ? { id: selectedFuncionario.id, nome:
selectedFuncionario.nome_completo } : { id: funcionarioId || null }) : null,
            },
            rules: {
              maxDays,
              minLeadMinutes,
              allowedWeekdays: Array.isArray(usuario?.dias_trabalho) ? usuario?.dias_trabalho : null,
              isDayDisabled: Number.isFinite(current.getTime()) ? isDayDisabledByRule(current) : null,
              weekday: Number.isFinite(current.getTime()) ? current.getDay() : null,
            },
            base: {
              usuario_id: usuario?.id ?? null,
              unidade_id: usuario?.unidade_id ?? null,
              horario_inicio: usuario?.horario_inicio ?? null,
              horario_fim: usuario?.horario_fim ?? null,
              intervalo_inicio: usuario?.intervalo_inicio ?? null,
              intervalo_fim: usuario?.intervalo_fim ?? null,
              timezone: (usuario as unknown as { timezone?: string | null } | null)?.timezone ?? null,
            },
            derived: {
              schedStartMin,
              schedEndMin,
              ivStartMin,
              ivEndMin,
```

```
              schedSpanMin,
              duracaoMin,
              duracaoMaiorQueExpediente: schedSpanMin !== null && duracaoMin !== null ? duracaoMin > schedSpanMin :
null,
            },
          }
          setDebugSlotsText(JSON.stringify(snapshot, null, 2))
        }
        return
      }

      const raw = (slotsRes.data ?? []) as unknown
      const rows = Array.isArray(raw) ? raw : []

      const list = rows
        .map((r) => {
          if (typeof r === 'string') return r.trim()
          if (!r || typeof r !== 'object') return ''
          const obj = r as Record<string, unknown>
          if (typeof obj.hora_inicio === 'string') return obj.hora_inicio.trim()
          if (typeof obj.hora === 'string') return obj.hora.trim()
          return ''
        })
        .filter(Boolean)

      const uniqueSorted = Array.from(new Set(list)).sort((a, b) => a.localeCompare(b))
      setAvailableSlots(uniqueSorted)
      setSlotsLoading(false)

      if (debugEnabled) {
        const current = new Date(`${data}T00:00:00`)
        const schedStartMin = parseHHMMToMinutes(usuario?.horario_inicio ?? null)
        const schedEndMin = parseHHMMToMinutes(usuario?.horario_fim ?? null)
        const ivStartMin = parseHHMMToMinutes(usuario?.intervalo_inicio ?? null)
        const ivEndMin = parseHHMMToMinutes(usuario?.intervalo_fim ?? null)
        const schedSpanMin = schedStartMin !== null && schedEndMin !== null ? schedEndMin - schedStartMin : null
        const duracaoMin = selectedServico?.duracao_minutos ?? null
        const snapshot = {
          ok: true,
          env: supabaseEnv.ok ? { url: supabaseEnv.values.VITE_SUPABASE_URL, anonKeyPrefix:
`${supabaseEnv.values.VITE_SUPABASE_ANON_KEY.slice(0, 12)}…` } : { missing: supabaseEnv.missing },
          route: { slug: slug ?? null, unidadeSlug: unidadeSlug ?? null },
          payload: slotsArgs,
          selected: {
            servico: selectedServico ? { id: selectedServico.id, duracao_minutos: selectedServico.duracao_minutos, nome:
selectedServico.nome } : null,
            funcionario: hasStaff ? (selectedFuncionario ? { id: selectedFuncionario.id, nome:
selectedFuncionario.nome_completo } : { id: funcionarioId || null }) : null,
          },
          rules: {
            maxDays,
            minLeadMinutes,
            allowedWeekdays: Array.isArray(usuario?.dias_trabalho) ? usuario?.dias_trabalho : null,
            isDayDisabled: Number.isFinite(current.getTime()) ? isDayDisabledByRule(current) : null,
            weekday: Number.isFinite(current.getTime()) ? current.getDay() : null,
          },
          base: {
            usuario_id: usuario?.id ?? null,
            unidade_id: usuario?.unidade_id ?? null,
            horario_inicio: usuario?.horario_inicio ?? null,
            horario_fim: usuario?.horario_fim ?? null,
            intervalo_inicio: usuario?.intervalo_inicio ?? null,
            intervalo_fim: usuario?.intervalo_fim ?? null,
            timezone: (usuario as unknown as { timezone?: string | null } | null)?.timezone ?? null,
          },
          derived: {
            schedStartMin,
            schedEndMin,
            ivStartMin,
            ivEndMin,
            schedSpanMin,
            duracaoMin,
```

```
            duracaoMaiorQueExpediente: schedSpanMin !== null && duracaoMin !== null ? duracaoMin > schedSpanMin : null,
          },
          result: {
            slots: uniqueSorted.length,
            slots_preview: uniqueSorted.slice(0, 40),
          },
        }
        setDebugSlotsText(JSON.stringify(snapshot, null, 2))
      }
    }
    run().catch((e: unknown) => {
      if (!alive) return
      setError(e instanceof Error ? e.message : 'Erro ao calcular horários')
      setSlotsLoading(false)
      if (debugEnabled) setDebugSlotsText(JSON.stringify({ ok: false, error: e instanceof Error ? e.message : 'Erro ao
calcular horários' }, null, 2))
    })

    return () => {
      alive = false
    }
  }, [data, debugEnabled, funcionarioId, hasStaff, isDayDisabledByRule, maxDays, minLeadMinutes, selectedFuncionario,
selectedServico, slug, unidadeSlug, usuario, usuarioId])

  const effectiveHora = useMemo(() => {
    if (!isDiaInteiro) return hora
    if (!data) return ''
    const slot = diaInteiroDisponibilidade[data] ?? ''
    return slot || ''
  }, [data, diaInteiroDisponibilidade, hora, isDiaInteiro])

  const submit = async () => {
    if (!usuarioId || !selectedServico || !data || !effectiveHora || !clienteNome.trim() || !clienteTelefone.trim())
return
    if (needsPlaca && !clientePlaca.trim()) return
    if (needsEndereco && !clienteEndereco.trim()) return
    setSubmitting(true)
    setError(null)
    setSuccessId(null)

    const telefone = normalizePhone(clienteTelefone)
    if (telefone.length < 8) {
      setError('Telefone inválido.')
      setSubmitting(false)
      return
    }

    const email = normalizeEmail(clienteEmail)
    if (clienteEmail.trim() && !email) {
      setError('Email inválido.')
      setSubmitting(false)
      return
    }

    const placa = clientePlaca.trim().replace(/[^a-zA-Z0-9-]/g, '').toUpperCase()
    const clienteNomeFinal = needsPlaca && placa ? `${clienteNome.trim()} (Placa: ${placa})` : clienteNome.trim()

    const extras: Record<string, unknown> = {}
    if (needsEndereco) extras.endereco = clienteEndereco.trim()
    if (needsPlaca && placa) extras.placa = placa
    if (email) extras.email = email
    const extrasFinal = Object.keys(extras).length > 0 ? extras : null

    const taxa = typeof selectedServico.taxa_agendamento === 'number' &&
Number.isFinite(selectedServico.taxa_agendamento) ? selectedServico.taxa_agendamento : 0
    const normalizedTaxa = Math.max(0, taxa)
    if (enableTaxaAgendamento && normalizedTaxa > 0) {
      const payload: Record<string, unknown> = {
        action: 'create_booking_fee_checkout',
        usuario_id: usuarioId,
        servico_id: selectedServico.id,
        data,
```

```
        hora_inicio: effectiveHora,
        cliente_nome: clienteNomeFinal,
        cliente_telefone: telefone,
        cliente_endereco: needsEndereco ? clienteEndereco.trim() : null,
        slug: slug ?? null,
        unidade_slug: unidadeSlug ?? null,
      }
      if (extrasFinal) payload.extras = extrasFinal
      if (hasStaff) payload.funcionario_id = funcionarioId
      if (usuario?.unidade_id) payload.unidade_id = usuario.unidade_id

      const res = await callPublicPaymentsFn(payload)
      if (!res.ok) {
        const obj = res.body && typeof res.body === 'object' ? (res.body as Record<string, unknown>) : null
        const msg =
          (typeof obj?.message === 'string' && obj.message.trim()) ||
          (typeof obj?.error === 'string' && obj.error.trim()) ||
          (typeof res.body === 'string' && res.body.trim()) ||
          `Erro ao iniciar pagamento (HTTP ${res.status}).`
        setError(msg)
        setSubmitting(false)
        return
      }

      const body = res.body && typeof res.body === 'object' ? (res.body as Record<string, unknown>) : null
      const checkoutUrl = typeof body?.checkout_url === 'string' ? body.checkout_url.trim() : ''
      const kind = typeof body?.kind === 'string' ? body.kind.trim().toLowerCase() : ''
      const agendamentoId = typeof body?.agendamento_id === 'string' ? body.agendamento_id.trim() : ''

      if (kind === 'credit' && agendamentoId) {
        setSuccessId(agendamentoId)
        setSubmitting(false)
        setModalOpen(false)
        return
      }

      if (!checkoutUrl) {
        setError('Falha ao iniciar pagamento: checkout_url ausente.')
        setSubmitting(false)
        return
      }

      window.location.href = checkoutUrl
      return
    }

    const createArgs: Record<string, unknown> = {
      p_usuario_id: usuarioId,
      p_data: data,
      p_hora_inicio: effectiveHora,
      p_servico_id: selectedServico.id,
      p_cliente_nome: clienteNomeFinal,
      p_cliente_telefone: telefone,
      p_funcionario_id: hasStaff ? funcionarioId : null,
    }
    if (extrasFinal) createArgs.p_extras = extrasFinal
    if (usuario?.unidade_id) createArgs.p_unidade_id = usuario.unidade_id

    const createRes = await callPublicRpcWithSignatureFallback<unknown>('public_create_agendamento_publico', createArgs, usuario?.unidade_id ?? null)
    if (!createRes.ok) {
      const msg = createRes.errorText
      const lower = msg.toLowerCase()
      const missingFn = isMissingRpcFnText(lower, 'public_create_agendamento_publico')
      if (missingFn) {
        setError(
          (unidadeSlug ?? '').trim()
            ? 'Configuração do Supabase incompleta: crie o SQL do link público e o SQL de Multi-unidades (EMPRESA).'
            : 'Configuração do Supabase incompleta: crie o SQL do link público (listar + agendar).'
        )
      } else if (lower.includes('data_passada')) {
        setError('Selecione uma data futura.')
```

```
      } else if (lower.includes('data_muito_futura')) {
        setError('Selecione uma data dentro de 1 ano.')
      } else if (lower.includes('antecedencia_minima')) {
        setError(`Selecione um horário com no mínimo ${Math.round(minLeadMinutes / 60)}h de antecedência.`)
      } else if (lower.includes('ocupado')) {
        setError('Esse horário acabou de ser ocupado. Selecione outro horário.')
      } else if (lower.includes('limite_mensal_atingido')) {
        setError('Este estabelecimento atingiu o limite de agendamentos do mês. Tente novamente no próximo mês.')
      } else if (needsEndereco && lower.includes('p_extras')) {
        setError('Atualização necessária no Supabase: aplique o SQL do link público para salvar o endereço.')
      } else {
        setError(msg)
      }
      setSubmitting(false)
      return
    }
    setSuccessId(createRes.data ? String(createRes.data) : 'ok')
    setSubmitting(false)
    setModalOpen(false)
  }

  useEffect(() => {
    const qp = (() => {
      try {
        return new URLSearchParams(window.location.search)
      } catch {
        return null
      }
    })()

    const paid = qp?.get('paid') === '1'
    const sessionId = (qp?.get('session_id') ?? '').trim()
    if (!enableTaxaAgendamento || !paid || !sessionId) return
    if (!syncingPayment || submitting) return

    callPublicPaymentsFn({ action: 'sync_booking_fee_session', session_id: sessionId })
      .then((res) => {
        if (!res.ok) {
          const obj = res.body && typeof res.body === 'object' ? (res.body as Record<string, unknown>) : null
          const msg =
            (typeof obj?.message === 'string' && obj.message.trim()) ||
            (typeof obj?.error === 'string' && obj.error.trim()) ||
            (typeof res.body === 'string' && res.body.trim()) ||
            `Erro ao confirmar pagamento (HTTP ${res.status}).`
          setError(msg)
          setSyncingPayment(false)
          return
        }
        const body = res.body && typeof res.body === 'object' ? (res.body as Record<string, unknown>) : null
        const agendamentoId = typeof body?.agendamento_id === 'string' ? body.agendamento_id.trim() : ''
        if (agendamentoId) setSuccessId(agendamentoId)
        setSyncingPayment(false)
      })
      .catch((e: unknown) => {
        setError(e instanceof Error ? e.message : 'Erro ao confirmar pagamento')
        setSyncingPayment(false)
      })
  }, [enableTaxaAgendamento, slug, submitting, syncingPayment, unidadeSlug])

  const canSubmit = Boolean(
    usuarioId &&
      selectedServico &&
      data &&
      effectiveHora &&
      clienteNome.trim() &&
      clienteTelefone.trim() &&
      (!hasStaff || funcionarioId) &&
      (!needsPlaca || clientePlaca.trim()) &&
      (!needsEndereco || clienteEndereco.trim())
  )

  const closeModal = () => {
```

```
    setModalOpen(false)
    setError(null)
  }


  const summary = useMemo(() => {
    const servicoNome = selectedServico?.nome ?? null
    const servicoPreco = typeof selectedServico?.preco === 'number' ? selectedServico.preco : null
    const servicoTaxa = typeof selectedServico?.taxa_agendamento === 'number' ? selectedServico.taxa_agendamento : null
    const profissionalNome = hasStaff ? (selectedFuncionario?.nome_completo ?? null) : null
    const canContinue = Boolean(usuarioId && selectedServico && data && effectiveHora && (!hasStaff || funcionarioId))
    const actionLabel = canContinue ? 'Continuar' : 'Escolher horário'
    return { servicoNome, servicoPreco, servicoTaxa, profissionalNome, canContinue, actionLabel }
  }, [data, effectiveHora, funcionarioId, hasStaff, selectedFuncionario?.nome_completo, selectedServico, usuarioId])

  const openFromFooter = () => {
    if (!usuario) return
    if (!summary.canContinue) {
      if (!selectedServico) {
        setStep('servico')
        setTimeout(() => {
          servicoRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
        }, 0)
        return
      }
      if (hasStaff && !funcionarioId) {
        setStep('profissional')
        setTimeout(() => {
          profissionalRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
        }, 0)
        return
      }

      setStep('quando')
      setTimeout(() => {
        whenRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
      }, 0)
      return
    }
    setModalOpen(true)
  }

  const whatsappHref = useMemo(() => {
    const digits = toWhatsappNumber(usuario?.telefone ?? '')
    return digits ? `https://wa.me/${digits}` : null
  }, [usuario?.telefone])

  const instagramHref = useMemo(() => {
    return normalizeInstagramUrl(String(usuario?.instagram_url ?? ''))
  }, [usuario?.instagram_url])

  const visibleSteps = useMemo(() => {
    return hasStaff ? (['servico', 'profissional', 'quando'] as const) : (['servico', 'quando'] as const)
  }, [hasStaff])

  const canGoStep = (target: 'servico' | 'profissional' | 'quando') => {
    if (target === 'servico') return true
    if (target === 'profissional') return Boolean(selectedServico && hasStaff)
    if (target === 'quando') return Boolean(selectedServico && (!hasStaff || funcionarioId))
    return false
  }

  const goStep = (target: 'servico' | 'profissional' | 'quando') => {
    if (!canGoStep(target)) return
    setStep(target)
    setTimeout(() => {
      if (target === 'servico') servicoRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
      if (target === 'profissional') profissionalRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
      if (target === 'quando') whenRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
    }, 0)
  }
```

```
  const slotGroups = useMemo(() => {
    const manha: string[] = []
    const tarde: string[] = []
    const noite: string[] = []
    for (const t of availableSlots) {
      const hh = Number(String(t).split(':')[0])
      if (!Number.isFinite(hh)) continue
      if (hh < 12) manha.push(t)
      else if (hh < 18) tarde.push(t)
      else noite.push(t)
    }
    return { manha, tarde, noite }
  }, [availableSlots])

  return (
    <div
      className="min-h-screen"
      style={{
        backgroundColor: shouldUseBgImage ? 'transparent' : backgroundColor,
        backgroundImage: shouldUseBgImage ? `url(${backgroundImageUrl})` : undefined,
        backgroundSize: shouldUseBgImage ? 'cover' : undefined,
        backgroundPosition: shouldUseBgImage ? 'center' : undefined,
      }}
    >
      <div className="min-h-screen" style={hasBgImage ? { backgroundColor: 'transparent' } : undefined}>
        <div className="mx-auto max-w-xl px-4 py-8 space-y-6">
          <div>
            <div className="text-xs font-semibold tracking-wide text-slate-500">SMagenda</div>
            <div className="text-2xl font-semibold text-slate-900">Agendar</div>
          </div>

          {loading ? <div className="text-sm text-slate-600">Carregando…</div> : null}
          {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div>
: null}
          {successId ? (
            <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">Agendamento
criado.</div>
          ) : null}

          {usuario ? (
            <Card>
              <div className="overflow-hidden rounded-xl">
                <div
                  className="h-24"
                  style={
                    usuario.public_use_background_image && usuario.public_background_image_url
                      ? { backgroundImage: `url(${usuario.public_background_image_url})`, backgroundSize: 'cover',
backgroundPosition: 'center' }
                      : {
                          backgroundImage: `linear-gradient(135deg, ${primaryColor} 0%, #0b1220 100%)`,
                        }
                  }
                />
                <div className="px-5 pb-5">
                  <div className="-mt-10 flex items-end gap-3">
                    {usuario.logo_url && !logoFailed ? (
                      <img
                        src={usuario.logo_url}
                        alt={usuario.nome_negocio}
                        className="h-16 w-16 rounded-full border-2 border-white object-cover shadow-sm"
                        onError={() => setLogoFailed(true)}
                      />
                    ) : (
                      <div className="h-16 w-16 rounded-full border-2 border-white bg-white/70 shadow-sm" />
                    )}
                    <div className="min-w-0">
                      <div className="text-lg font-semibold text-slate-900 truncate">{usuario.nome_negocio}</div>
                      {usuario.unidade_nome ? <div className="text-xs text-slate-600 truncate">{usuario.unidade_nome}
</div> : null}
                    </div>
                  </div>
```

```
          <div className="mt-4 flex flex-wrap gap-2">
            <Badge tone="green">Agendamento online</Badge>
            {usuario.endereco ? <Badge>Endereço</Badge> : null}
            {usuario.telefone && whatsappHref ? (
              <a href={whatsappHref} target="_blank" rel="noreferrer noopener" className="inline-flex">
                <Badge>WhatsApp</Badge>
              </a>
            ) : usuario.telefone ? (
              <Badge>WhatsApp</Badge>
            ) : null}
            {instagramHref ? (
              <a href={instagramHref} target="_blank" rel="noreferrer noopener" className="inline-flex">
                <Badge>Instagram</Badge>
              </a>
            ) : null}
          </div>

          <div className="mt-4 space-y-2">
            {usuario.endereco ? (
              <a
                className="block text-sm text-slate-700 underline underline-offset-2"
                href={toGoogleMapsLink(usuario.endereco)}
                target="_blank"
                rel="noreferrer noopener"
              >
                {usuario.endereco}
              </a>
            ) : null}
          </div>
        </div>
      </div>
    </Card>
  ) : null}

  {usuario ? (
    <Card>
      <div className="p-4 space-y-4">
        <div className="flex gap-2">
          {visibleSteps.map((s) => {
            const active = step === s
            const enabled = canGoStep(s)
            const label =
              s === 'servico'
                ? `1. ${labels.servico}`
                : s === 'profissional'
                  ? `2. ${labels.profissional}`
                  : `${hasStaff ? '3' : '2'}. Quando`
            return (
              <button
                key={s}
                type="button"
                disabled={!enabled}
                onClick={() => goStep(s)}
                className={[
                  'flex-1 rounded-lg border px-3 py-2 text-sm font-medium',
                  enabled ? 'cursor-pointer' : 'opacity-40 cursor-not-allowed',
                  active ? 'text-white' : 'bg-white border-slate-200 text-slate-700',
                ].join(' ')}
                style={active ? { backgroundColor: primaryColor, borderColor: primaryColor } : undefined}
              >
                {label}
              </button>
            )
          })}
        </div>
      </div>
    </Card>
  ) : null}

  {usuario ? (
    <div ref={servicoRef} className="space-y-3">
      <div className="text-sm font-semibold text-slate-900">1) {labels.servico}</div>
```

```
              {servicos.length === 0 ? (
                <div className="text-sm text-slate-600">Sem {labels.servicos.toLowerCase()} disponíveis.</div>
              ) : (
                <div className="grid grid-cols-1 gap-3">
                  {servicos.map((s) => {
                    const selected = servicoId === s.id
                    const hasPreco = typeof s.preco === 'number'
                    return (
                      <button
                        key={s.id}
                        type="button"
                        onClick={() => {
                          setError(null)
                          setSuccessId(null)
                          setHora('')
                          setServicoId(s.id)
                          const next = hasStaff ? 'profissional' : 'quando'
                          setStep(next)
                          setTimeout(() => {
                            if (next === 'profissional') profissionalRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
                            if (next === 'quando') whenRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
                          }, 0)
                        }}
                        className={[
                          'rounded-2xl border p-4 text-left transition',
                          selected ? 'text-white' : 'bg-white border-slate-200 text-slate-900 hover:bg-slate-50',
                        ].join(' ')}
                        style={selected ? { backgroundColor: primaryColor, borderColor: primaryColor } : undefined}
                      >
                        <div className="flex items-start justify-between gap-3">
                          <div className="flex items-start gap-3 min-w-0">
                            {s.foto_url ? (
                              <img
                                src={s.foto_url}
                                alt={s.nome}
                                className="h-14 w-14 rounded-xl object-cover border border-slate-200"
                                loading="lazy"
                              />
                            ) : (
                              <div className={['h-14 w-14 rounded-xl border border-slate-200', selected ? 'bg-white/20' : 'bg-slate-50'].join(' ')} />
                            )}
                            <div className="min-w-0">
                              <div className="text-sm font-semibold truncate">{s.nome}</div>
                              {s.descricao ? <div className={selected ? 'mt-1 text-xs text-white/90' : 'mt-1 text-xs text-slate-600'}>{s.descricao}</div> : null}
                              <div className="mt-2 flex flex-wrap gap-2">
                                <span
                                  className={[
                                    'inline-flex items-center rounded-full px-2 py-0.5 text-xs font-medium',
                                    selected ? 'bg-white/20 text-white' : 'bg-slate-100 text-slate-700',
                                  ].join(' ')}
                                >
                                  {s.duracao_minutos} min
                                </span>
                                {hasPreco ? (
                                  <span
                                    className={[
                                      'inline-flex items-center rounded-full px-2 py-0.5 text-xs font-medium',
                                      selected ? 'bg-white/20 text-white' : 'bg-slate-100 text-slate-700',
                                    ].join(' ')}
                                  >
                                    {formatBRMoney(s.preco)}
                                  </span>
                                ) : null}
                              </div>
                            </div>
                          </div>
                          <div className={['shrink-0 rounded-full px-3 py-1 text-xs font-semibold', selected ? 'bg-white/20 text-white' : 'bg-slate-900 text-white'].join(' ')}>
```

```
                            {selected ? 'Selecionado' : 'Selecionar'}
                          </div>
                        </div>
                      </button>
                    )
                  })}
                </div>
              )}
            </div>
          ) : null}

          {usuario && hasStaff && step !== 'servico' ? (
            <div ref={profissionalRef} className="space-y-3">
              <div className="text-sm font-semibold text-slate-900">2) {labels.profissional}</div>
              {funcionarios.length === 0 ? (
                <div className="text-sm text-slate-600">Sem {labels.profissionais.toLowerCase()} disponíveis.</div>
              ) : (
                <div className="grid grid-cols-1 gap-3">
                  {funcionarios.map((f) => {
                    const selected = funcionarioId === f.id
                    return (
                      <button
                        key={f.id}
                        type="button"
                        onClick={() => {
                          setError(null)
                          setSuccessId(null)
                          setHora('')
                          setFuncionarioId(f.id)
                          setStep('quando')
                          setTimeout(() => {
                            whenRef.current?.scrollIntoView({ behavior: 'smooth', block: 'start' })
                          }, 0)
                        }}
                        className={[
                          'rounded-2xl border p-4 text-left transition',
                          selected ? 'text-white' : 'bg-white border-slate-200 text-slate-900 hover:bg-slate-50',
                        ].join(' ')}
                        style={selected ? { backgroundColor: primaryColor, borderColor: primaryColor } : undefined}
                      >
                        <div className="flex items-center justify-between gap-3">
                          <div className="text-sm font-semibold truncate">{f.nome_completo}</div>
                          <div className={['shrink-0 rounded-full px-3 py-1 text-xs font-semibold', selected ? 'bg-
white/20 text-white' : 'bg-slate-900 text-white'].join(' ')}>
                            {selected ? 'Selecionado' : 'Selecionar'}
                          </div>
                        </div>
                      </button>
                    )
                  })}
                </div>
              )}
            </div>
          ) : null}

          {usuario && step === 'quando' ? (
            <div ref={whenRef} className="space-y-3">
              <div className="text-sm font-semibold text-slate-900">{hasStaff ? '3' : '2'}) Quando</div>
              <Card>
                <div className="p-4 space-y-4">
                  <div>
                    <div className="text-xs font-semibold tracking-wide text-slate-500">Data</div>
                    {isDiaInteiro ? (
                      <DiaInteiroCalendar
                        key={`${servicoId}:${funcionarioId ?? ''}:${usuarioId ?? ''}`}
                        primaryColor={primaryColor}
                        selectedIso={data}
                        setSelectedIso={(iso) => {
                          setError(null)
                          setSuccessId(null)
                          setHora('')
                          setData(iso)
```

```
                          setStep('quando')
                        }}
                        minIso={minIso}
                        maxIso={maxIso}
                        monthIso={diaInteiroMonthIso}
                        setMonthIso={setDiaInteiroMonthIso}
                        isDayDisabled={isDayDisabledByRule}
                        diaInteiroDisponibilidade={diaInteiroDisponibilidade}
                        calendarioLoading={calendarioLoading}
                      />
                    ) : (
                      <button
                        type="button"
                        onClick={() => {
                          const baseIso = data || minIso
                          const base = baseIso ? new Date(`${baseIso}T00:00:00`) : null
                          const safeBase = base && Number.isFinite(base.getTime()) ? base : new Date()
                          setDatePopupMonthIso(toIsoDateLocal(startOfMonth(safeBase)))
                          setDatePopupOpen(true)
                        }}
                        aria-haspopup="dialog"
                        aria-expanded={datePopupOpen}
                        className="mt-2 w-full rounded-xl border border-slate-200 bg-white px-3 py-3 text-left hover:bg-slate-50"
                      >
                        <div className="text-xs font-semibold text-slate-500">Data selecionada</div>
                        <div className="mt-0.5 text-sm font-semibold text-slate-900">{formatIsoDateBR(data)}</div>
                      </button>
                    )}

                    {!isDiaInteiro ? (
                      <DatePopupCalendar
                        open={datePopupOpen}
                        primaryColor={primaryColor}
                        selectedIso={data}
                        monthIso={datePopupMonthIso}
                        setMonthIso={setDatePopupMonthIso}
                        onClose={() => setDatePopupOpen(false)}
                        onSelect={(iso) => {
                          setError(null)
                          setSuccessId(null)
                          setHora('')
                          setData(iso)
                          setDatePopupMonthIso(toIsoDateLocal(startOfMonth(new Date(`${iso}T00:00:00`))))
                          setDatePopupOpen(false)
                          setStep('quando')
                        }}
                        minIso={minIso}
                        maxIso={maxIso}
                        isDayDisabled={isDayDisabledByRule}
                        dayHasSlots={calendarDayHasSlots}
                        availabilityLoading={calendarioLoading}
                      />
                    ) : null}

                    <div className="mt-2 text-xs text-slate-600">
                      {calendarioLoading ? 'Carregando disponibilidade do calendário… ' : ''}Escolha qualquer data até 1 ano a partir de hoje (mude o mês no calendário). Antecedência mínima: {Math.round(minLeadMinutes / 60)}h.
                    </div>
                  </div>

                  {!isDiaInteiro ? (
                    <div>
                      <div className="text-xs font-semibold tracking-wide text-slate-500">Horário</div>
                      {!selectedServico ? (
                        <div className="mt-2 text-sm text-slate-600">Selecione um {labels.servico.toLowerCase()}.</div>
                      ) : hasStaff && !selectedFuncionario ? (
                        <div className="mt-2 text-sm text-slate-600">Selecione um {labels.profissional.toLowerCase()}.</div>
                      ) : slotsLoading ? (
                        <div className="mt-2 text-sm text-slate-600">Buscando horários…</div>
                      ) : availableSlots.length === 0 ? (
```

```
                        <div className="mt-2 text-sm text-slate-600">Sem horários disponíveis.</div>
                      ) : (
                        <div className="mt-3 space-y-4">
                          {slotGroups.manha.length > 0 ? (
                            <div className="space-y-2">
                              <div className="text-xs font-semibold text-slate-700">Manhã</div>
                              <div className="grid grid-cols-3 gap-2">
                                {slotGroups.manha.map((t) => {
                                  const selected = hora === t
                                  return (
                                    <button
                                      key={t}
                                      type="button"
                                      onClick={() => setHora(t)}
                                      className={['rounded-xl px-3 py-2 text-sm font-semibold border', selected ? '' :
'bg-white text-slate-700 border-slate-200'].join(' ')}
                                      style={selected ? { backgroundColor: primaryColor, borderColor: primaryColor,
color: '#fff' } : { backgroundColor: '#fff' }}
                                    >
                                      {t}
                                    </button>
                                  )
                                })}
                              </div>
                            </div>
                          ) : null}

                          {slotGroups.tarde.length > 0 ? (
                            <div className="space-y-2">
                              <div className="text-xs font-semibold text-slate-700">Tarde</div>
                              <div className="grid grid-cols-3 gap-2">
                                {slotGroups.tarde.map((t) => {
                                  const selected = hora === t
                                  return (
                                    <button
                                      key={t}
                                      type="button"
                                      onClick={() => setHora(t)}
                                      className={['rounded-xl px-3 py-2 text-sm font-semibold border', selected ? '' :
'bg-white text-slate-700 border-slate-200'].join(' ')}
                                      style={selected ? { backgroundColor: primaryColor, borderColor: primaryColor,
color: '#fff' } : { backgroundColor: '#fff' }}
                                    >
                                      {t}
                                    </button>
                                  )
                                })}
                              </div>
                            </div>
                          ) : null}

                          {slotGroups.noite.length > 0 ? (
                            <div className="space-y-2">
                              <div className="text-xs font-semibold text-slate-700">Noite</div>
                              <div className="grid grid-cols-3 gap-2">
                                {slotGroups.noite.map((t) => {
                                  const selected = hora === t
                                  return (
                                    <button
                                      key={t}
                                      type="button"
                                      onClick={() => setHora(t)}
                                      className={['rounded-xl px-3 py-2 text-sm font-semibold border', selected ? '' :
'bg-white text-slate-700 border-slate-200'].join(' ')}
                                      style={selected ? { backgroundColor: primaryColor, borderColor: primaryColor,
color: '#fff' } : { backgroundColor: '#fff' }}
                                    >
                                      {t}
                                    </button>
                                  )
                                })}
                              </div>
```

```
                            </div>
                          ) : null}
                        </div>
                      )}

                      {debugEnabled && debugSlotsText ? (
                        <div className="mt-4">
                          <Card>
                            <div className="p-4 space-y-2">
                              <div className="text-xs font-semibold tracking-wide text-slate-500">Debug (slots)</div>
                              <pre className="text-[11px] leading-4 text-slate-700 whitespace-pre-wrap break-words font-mono">{debugSlotsText}</pre>
                            </div>
                          </Card>
                        </div>
                      ) : null}
                    </div>
                  ) : (
                    <div>
                      <div className="text-xs font-semibold tracking-wide text-slate-500">Diária</div>
                      {!selectedServico ? (
                        <div className="mt-2 text-sm text-slate-600">Selecione um {labels.servico.toLowerCase()}.</div>
                      ) : hasStaff && !selectedFuncionario ? (
                        <div className="mt-2 text-sm text-slate-600">Selecione um {labels.profissional.toLowerCase()}.</div>
                      ) : slotsLoading ? (
                        <div className="mt-2 text-sm text-slate-600">Buscando disponibilidade…</div>
                      ) : hora ? (
                        <div className="mt-2 text-sm text-slate-700">Dia selecionado com disponibilidade.</div>
                      ) : (
                        <div className="mt-2 text-sm text-slate-600">Selecione um dia disponível no calendário.</div>
                      )}
                    </div>
                  )}
                </div>
              </Card>
            </div>
          ) : null}
        </div>
      </div>

      {usuario && !modalOpen && !successId ? (
        <div className="fixed bottom-0 left-0 right-0 z-40">
          <div className="mx-auto max-w-xl px-4 pb-4">
            <div className="rounded-2xl border border-slate-200 bg-white/95 backdrop-blur p-4 shadow-sm">
              <div className="flex items-center justify-between gap-3">
                <div className="min-w-0">
                  <div className="text-xs font-semibold tracking-wide text-slate-500">Resumo do agendamento</div>
                  <div className="mt-1 text-sm font-semibold text-slate-900 truncate">
                    {formatIsoDateBR(data)} {hora ? `• ${hora}` : ''}
                  </div>
                  <div className="mt-1 text-xs text-slate-700 truncate">
                    {labels.servico}: {summary.servicoNome ?? '—'}
                    {summary.servicoPreco !== null ? ` • ${formatBRMoney(summary.servicoPreco)}` : ''}
                  </div>
                  {hasStaff ? (
                    <div className="mt-1 text-xs text-slate-700 truncate">
                      {labels.profissional}: {summary.profissionalNome ?? '—'}
                    </div>
                  ) : null}
                </div>
                <Button
                  onClick={openFromFooter}
                  style={{ backgroundColor: primaryColor, borderColor: primaryColor }}
                  className="px-6 py-3 text-base shadow-md"
                >
                  {summary.actionLabel}
                </Button>
              </div>
            </div>
          </div>
        </div>
```

```
      ) : null}

      {usuario && !modalOpen && !successId ? <div className="h-28" /> : null}

      {modalOpen && usuario ? (
        <div className="fixed inset-0 z-50 flex items-center justify-center p-4">
          <button type="button" className="absolute inset-0 bg-slate-900/40" onClick={closeModal} aria-label="Fechar" />
          <div className="relative w-full max-w-lg">
            <Card>
              <div className="p-6 space-y-4">
                <div className="flex items-start justify-between gap-3">
                  <div>
                    <div className="text-sm font-semibold text-slate-900">Agendar em {formatIsoDateBR(data)}</div>
                    <div className="text-xs text-slate-600">{usuario.nome_negocio}</div>
                  </div>
                  <Button variant="secondary" onClick={closeModal}>
                    Fechar
                  </Button>
                </div>

                {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}

                <div className="space-y-4">
                  <div className="text-sm font-semibold text-slate-900">Confirmar agendamento</div>

                  <div className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-sm text-slate-700">
                    <div className="flex items-center justify-between gap-2">
                      <div>Dia</div>
                      <div className="font-semibold">{formatIsoDateBR(data)}</div>
                    </div>
                    <div className="flex items-center justify-between gap-2">
                      <div>Hora</div>
                      <div className="font-semibold">{hora || '—'}</div>
                    </div>
                    {needsPlaca && clientePlaca.trim() ? (
                      <div className="flex items-center justify-between gap-2">
                        <div>Placa</div>
                        <div className="font-semibold">{clientePlaca.trim().replace(/[^a-zA-Z0-9-]/g, '').toUpperCase()}</div>
                      </div>
                    ) : null}
                    {needsEndereco && clienteEndereco.trim() ? (
                      <div className="flex items-center justify-between gap-2">
                        <div>Endereço</div>
                        <div className="font-semibold text-right">{clienteEndereco.trim()}</div>
                      </div>
                    ) : null}
                    <div className="flex items-center justify-between gap-2">
                      <div>{labels.servico}</div>
                      <div className="font-semibold">{selectedServico?.nome ?? '—'}</div>
                    </div>
                    {hasStaff ? (
                      <div className="flex items-center justify-between gap-2">
                        <div>{labels.profissional}</div>
                        <div className="font-semibold">{selectedFuncionario?.nome_completo ?? '—'}</div>
                      </div>
                    ) : null}
                  </div>

                  <Input label="Nome" value={clienteNome} onChange={(e) => setClienteNome(e.target.value)} />
                  <Input label="WhatsApp" value={clienteTelefone} onChange={(e) => setClienteTelefone(e.target.value)} />
                  <Input label="Email (opcional)" value={clienteEmail} onChange={(e) => setClienteEmail(e.target.value)} />
                  {needsPlaca ? <Input label="Placa do carro" value={clientePlaca} onChange={(e) => setClientePlaca(e.target.value)} /> : null}
                  {needsEndereco ? <Input label="Endereço" value={clienteEndereco} onChange={(e) => setClienteEndereco(e.target.value)} /> : null}

                  <div className="flex justify-between">
                    <Button variant="secondary" onClick={closeModal}>
```

```
                      Voltar
                    </Button>
                    <Button
                      onClick={submit}
                      disabled={!canSubmit || submitting}
                      style={{ backgroundColor: primaryColor, borderColor: primaryColor }}
                      className="px-7 py-3 text-base shadow-lg"
                    >
                      {submitting ? 'Confirmando…' : 'Confirmar'}
                    </Button>
                  </div>
                </div>
              </div>
            </Card>
          </div>
        </div>
      ) : null}

      {successId ? (
        <div className="fixed inset-0 z-50 flex items-center justify-center p-4">
          <div className="absolute inset-0 bg-slate-900/40" />
          <div className="relative w-full max-w-lg">
            <Card>
              <div className="p-6 space-y-4 text-center">
                <div className="text-lg font-semibold text-slate-900">Agendamento enviado!</div>
                <div className="text-sm text-slate-700">Seu agendamento foi enviado. Você receberá a confirmação em
breve.</div>
                {data || hora ? (
                  <div className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-sm text-slate-700">
                    <div className="flex items-center justify-between gap-2">
                      <div>Dia</div>
                      <div className="font-semibold">{data ? formatIsoDateBR(data) : '—'}</div>
                    </div>
                    <div className="flex items-center justify-between gap-2">
                      <div>Hora</div>
                      <div className="font-semibold">{hora || '—'}</div>
                    </div>
                  </div>
                ) : null}
                <Button
                  onClick={() => {
                    setSuccessId(null)
                    setModalOpen(false)
                    setError(null)
                  }}
                  style={{ backgroundColor: primaryColor, borderColor: primaryColor }}
                  className="py-3 text-base shadow-md"
                  fullWidth
                >
                  Fechar
                </Button>
              </div>
            </Card>
          </div>
        </div>
      ) : null}
    </div>
  )
}
```

smagenda/src/views/app/PagamentoPage.tsx

```
import { useEffect, useMemo, useState } from 'react'
import { useLocation, useNavigate } from 'react-router-dom'
import { AppShell } from '../../components/layout/AppShell'
import { Badge } from '../../components/ui/Badge'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
```

```
import { checkJwtProject, supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

type FnResult = { ok: true; status: number; body: unknown } | { ok: false; status: number; body: unknown }

function safeJson(value: unknown) {
  try {
    return JSON.stringify(value)
  } catch {
    return null
  }
}

async function callPaymentsFn(body: Record<string, unknown>): Promise<FnResult> {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } }
  }

  const supabaseUrl = String(supabaseEnv.values.VITE_SUPABASE_URL ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
    .replace(/\/+$/g, '')
  const supabaseAnonKey = String(supabaseEnv.values.VITE_SUPABASE_ANON_KEY ?? '')
    .trim()
    .replace(/^['"`\s]+|['"`\s]+$/g, '')
  const fnUrl = `${supabaseUrl}/functions/v1/payments`

  const tryRefresh = async () => {
    const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
    if (refreshErr) return null
    return refreshed.session ?? null
  }

  const { data: sessionData } = await supabase.auth.getSession()
  let session = sessionData.session
  const now = Math.floor(Date.now() / 1000)
  const expiresAt = session?.expires_at ?? null

  if (session && (!expiresAt || expiresAt <= now + 60)) {
    const refreshed = await tryRefresh()
    if (refreshed) session = refreshed
  }

  if (session) {
    const { error: userErr } = await supabase.auth.getUser()
    const userErrMsg = typeof userErr?.message === 'string' ? userErr.message : ''
    if (userErr && /invalid\s+jwt/i.test(userErrMsg)) {
      const refreshed = await tryRefresh()
      if (refreshed) session = refreshed
    }
  }

  const token = session?.access_token ?? null
  if (!token) {
    return { ok: false as const, status: 401, body: { error: 'session_expired' } }
  }

  const tokenProject = checkJwtProject(token, supabaseUrl)
  if (!tokenProject.ok) {
    await supabase.auth.signOut().catch(() => undefined)
    return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: tokenProject.iss, expected:
tokenProject.expectedPrefix } }
  }

  const callFetch = async (jwt: string) => {
    let res: Response
    try {
      res = await fetch(fnUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          apikey=[REDACTED]
```

```
          Authorization: `Bearer ${jwt}`,
        },
        body: JSON.stringify(body),
      })
    } catch (e: unknown) {
      const msg = e instanceof Error ? e.message : 'Falha de rede'
      return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
    }

    const text = await res.text()
    let parsed: unknown = null
    try {
      parsed = text ? JSON.parse(text) : null
    } catch {
      parsed = text
    }

    if (!res.ok && res.status === 404) {
      const raw = typeof parsed === 'string' ? parsed : text
      if (typeof raw === 'string' && raw.includes('Requested function was not found')) {
        return {
          ok: false as const,
          status: 404,
          body: {
            error: 'function_not_deployed',
            message: 'A função payments não está publicada no Supabase. Faça deploy da Edge Function `payments` no seu
projeto.',
          },
        }
      }
    }

    if (
      !res.ok &&
      res.status === 401 &&
      parsed &&
      typeof parsed === 'object' &&
      (parsed as Record<string, unknown>).message === 'Invalid JWT' &&
      (parsed as Record<string, unknown>).code === 401
    ) {
      return { ok: false as const, status: 401, body: { error: 'supabase_gateway_invalid_jwt' } }
    }

    if (!res.ok) console.error('payments error', { status: res.status, body: parsed, body_json: safeJson(parsed) })

    if (!res.ok) return { ok: false as const, status: res.status, body: parsed }
    return { ok: true as const, status: res.status, body: parsed }
  }

  const first = await callFetch(token)
  if (!first.ok && first.status === 401) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) {
      await supabase.auth.signOut().catch(() => undefined)
      return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    }
    const nextProject = checkJwtProject(nextToken, supabaseUrl)
    if (!nextProject.ok) {
      await supabase.auth.signOut().catch(() => undefined)
      return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: nextProject.iss, expected:
nextProject.expectedPrefix } }
    }
    return callFetch(nextToken)
  }

  return first
}

async function createCheckoutPagamento(
  usuarioId: string,
  item: string,
```

```
  metodo: 'card' | 'pix',
  funcionariosTotal?: number | null
): Promise<FnResult> {
  const payload: Record<string, unknown> = { action: 'create_checkout', usuario_id: usuarioId, plano: item, metodo }
  if (typeof funcionariosTotal === 'number' && Number.isFinite(funcionariosTotal)) payload.funcionarios_total =
funcionariosTotal
  return callPaymentsFn(payload)
}

async function createBillingPortalPagamento(usuarioId: string): Promise<FnResult> {
  const payload: Record<string, unknown> = { action: 'create_billing_portal', usuario_id: usuarioId }
  return callPaymentsFn(payload)
}

async function syncCheckoutSessionPagamento(sessionId: string, usuarioId: string | null): Promise<FnResult> {
  const payload: Record<string, unknown> = { action: 'sync_checkout_session', session_id: sessionId }
  if (usuarioId) payload.usuario_id = usuarioId
  return callPaymentsFn(payload)
}

type PlanKey = 'free' | 'basic' | 'pro' | 'team' | 'enterprise'

type PlanCard = {
  key: PlanKey
  title: string
  priceLabel: string
  subtitle: string
  bullets: string[]
}

type ServiceCard = {
  key: 'setup_completo' | 'consultoria_hora'
  title: string
  priceLabel: string
  bullets: string[]
}

export function PagamentoPage() {
  const { appPrincipal, refresh } = useAuth()
  const location = useLocation()
  const navigate = useNavigate()
  const usuario = appPrincipal?.kind === 'usuario' ? appPrincipal.profile : null
  const usuarioId = usuario?.id ?? null

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Escolha um plano',
          body: 'Selecione o plano ideal para seu negócio e veja o que está incluído.',
          target: 'plans' as const,
        },
        {
          title: 'Iniciar pagamento',
          body: 'Escolha PIX (30 dias) ou cartão (assinatura) para abrir o checkout em produção.',
          target: 'checkout' as const,
        },
        {
          title: 'Serviços avulsos',
          body: 'Use esta área para contratar setup completo ou consultoria, quando precisar.',
          target: 'services' as const,
        },
      ] as const,
    []
  )

  const [checkoutNotice, setCheckoutNotice] = useState<null | { kind: 'success' | 'cancel'; item: string | null }>(null)
  const [userSelectedPlan, setUserSelectedPlan] = useState<PlanKey | null>(null)
  const [selectedService, setSelectedService] = useState<ServiceCard['key'] | null>(null)
  const [creatingCheckout, setCreatingCheckout] = useState(false)
  const [openingPortal, setOpeningPortal] = useState(false)
  const [error, setError] = useState<string | null>(null)
```

```
  const [funcionariosTotal, setFuncionariosTotal] = useState<number | null>(null)

  const formatCheckoutError = (status: number, body: unknown) => {
    if (typeof body === 'string' && body.trim()) return body
    if (body && typeof body === 'object') {
      const obj = body as Record<string, unknown>
      const err = typeof obj.error === 'string' ? obj.error : null
      const message = typeof obj.message === 'string' && obj.message.trim() ? obj.message.trim() : null
      if (message) {
        const stripeStatus = typeof obj.stripe_status === 'number' && Number.isFinite(obj.stripe_status) ? obj.stripe_status : null
        if (err === 'stripe_error') {
          const mode = typeof obj.stripe_key_mode === 'string' && obj.stripe_key_mode.trim() ? obj.stripe_key_mode.trim() : 'unknown'
          if (stripeStatus) return `Stripe (${mode}, HTTP ${stripeStatus}): ${message}`
          return `Stripe (${mode}): ${message}`
        }
        return message
      }
      if (err === 'missing_supabase_env') return 'Configuração do Supabase ausente no ambiente.'
      if (err === 'invalid_action') {
        return 'A função payments do Supabase está desatualizada (não reconhece a ação solicitada). Faça deploy da Edge Function payments e tente novamente.'
      }
      if (err === 'network_error') return typeof obj.message === 'string' && obj.message.trim() ? obj.message : 'Falha de rede ao iniciar pagamento.'
      if (err === 'session_expired' || err === 'invalid_jwt') return 'Sessão expirada no Supabase. Saia e entre novamente.'
      if (err === 'jwt_project_mismatch') return 'Sessão do Supabase pertence a outro projeto. Saia e entre novamente.'
      if (err === 'supabase_gateway_invalid_jwt') return 'A Edge Function está exigindo JWT no gateway. Faça deploy com verify_jwt=false.'
      if (err === 'function_not_deployed') return 'A função payments não está publicada no Supabase.'
      const asJson = safeJson(body)
      if (err && asJson) return `Erro ao iniciar pagamento: ${err} (HTTP ${status}): ${asJson}`
      if (err) return `Erro ao iniciar pagamento: ${err} (HTTP ${status}).`
      if (asJson) return `Erro ao iniciar pagamento (HTTP ${status}): ${asJson}`
    }
    return `Erro ao iniciar pagamento (HTTP ${status}).`
  }

  const canShowFree = Boolean(usuario && usuario.plano === 'free' && usuario.status_pagamento === 'trial' && usuario.free_trial_consumido !== true)

  const plans = useMemo<PlanCard[]>(
    () =>
      [
        ...(canShowFree
          ? [
              {
                key: 'free' as const,
                title: 'FREE',
                priceLabel: 'R$ 0/mês',
                subtitle: 'Para testar',
                bullets: ['Até 30 agendamentos por mês', '1 profissional', 'Lembretes manuais (link do WhatsApp)', 'Suporte por email'],
              },
            ]
          : []),
        {
          key: 'basic',
          title: 'BASIC',
          priceLabel: 'R$ 34,99/mês',
          subtitle: '',
          bullets: ['Agendamentos 60 por mês', '1 profissional incluído', 'Lembretes automáticos via WhatsApp', 'Até 3 serviços', 'Página pública personalizável', 'Suporte por email'],
        },
        {
          key: 'pro',
          title: 'PRO',
          priceLabel: 'R$ 59,99/mês',
          subtitle: 'Até 6 profissionais (4 inclusos + até 2 adicionais)',
          bullets: ['4 profissionais incluídos', 'Serviços ilimitados', 'Logo e fotos de serviços', 'Relatórios',
```

```
'Bloqueios recorrentes', 'Suporte via WhatsApp'],
        },
        {
          key: 'enterprise',
          title: 'EMPRESA',
          priceLabel: 'R$ 98,99/mês',
          subtitle: 'Até 10 profissionais',
          bullets: ['Até 10 profissionais', 'Multi-unidades', 'Agendamentos ilimitados', 'Serviços ilimitados', 'Logo e
fotos de serviços', 'Para mais profissionais, fale com o suporte'],
        },
      ] satisfies PlanCard[],
    [canShowFree]
  )

  const services = useMemo<ServiceCard[]>(
    () =>
      [
        { key: 'setup_completo', title: 'Setup Completo', priceLabel: 'R$ 150 (uma vez)', bullets: ['Configuramos tudo
para você', 'Cadastramos serviços, fotos, horários', 'Testamos envios do WhatsApp apos conexão', 'Treinamos o cliente em
15 minutos'] },
        { key: 'consultoria_hora', title: 'Consultoria por Hora', priceLabel: 'R$ 80/hora', bullets: ['Ajuda com
configurações avançadas', 'Sugestões de otimização', 'Dúvidas gerais'] },
      ] satisfies ServiceCard[],
    []
  )

  const currentPlan = useMemo<PlanKey>(() => {
    const current = (usuario?.plano ?? '').trim().toLowerCase() as PlanKey
    if (current === 'free') return canShowFree ? 'free' : 'basic'
    if (current === 'enterprise') return 'enterprise'
    if (current === 'team') return 'pro'
    if (current === 'basic' || current === 'pro') return current
    return 'basic'
  }, [canShowFree, usuario?.plano])

  const selectedPlan = userSelectedPlan ?? currentPlan

  const defaultFuncionariosTotal = useMemo(() => {
    const raw = usuario?.limite_funcionarios
    const n = typeof raw === 'number' && Number.isFinite(raw) && raw > 0 ? Math.floor(raw) : 1
    return Math.max(1, Math.min(200, n))
  }, [usuario?.limite_funcionarios])

  const includedPro = 4
  const maxPro = 6
  const defaultProFuncionariosTotal = useMemo(() => {
    return Math.min(maxPro, Math.max(includedPro, defaultFuncionariosTotal))
  }, [defaultFuncionariosTotal, includedPro, maxPro])

  const effectiveFuncionariosTotal = funcionariosTotal ?? (selectedPlan === 'pro' ? defaultProFuncionariosTotal :
defaultFuncionariosTotal)

  useEffect(() => {
    const run = async () => {
      const search = location.search ?? ''
      if (!search) return
      const params = new URLSearchParams(search)
      const checkout = (params.get('checkout') ?? '').trim().toLowerCase()
      if (checkout !== 'success' && checkout !== 'cancel') return

      const item = (params.get('item') ?? params.get('plano') ?? '').trim().toLowerCase() || null
      setCheckoutNotice({ kind: checkout, item })

      const sessionId = (params.get('session_id') ?? '').trim()
      const usuarioIdFromParams = (params.get('usuario_id') ?? '').trim() || null

      const first = await refresh()
      const refreshedUsuarioId = first?.kind === 'usuario' ? first.profile.id : null
      const effectiveUsuarioId = refreshedUsuarioId ?? usuarioIdFromParams

      if (checkout === 'success') {
        if (sessionId) {
```

```
          const sync = await syncCheckoutSessionPagamento(sessionId, effectiveUsuarioId)
          if (!sync.ok) {
            setError(formatCheckoutError(sync.status, sync.body))
          } else {
            await refresh()
          }
        }

        let tries = 0
        while (tries < 10) {
          await new Promise((r) => setTimeout(r, 2000))
          const next = await refresh()
          if (next?.kind === 'usuario' && next.profile.status_pagamento === 'ativo') break
          tries += 1
        }
      }

      navigate('/pagamento', { replace: true })
    }
    run().catch(() => undefined)
  }, [location.search, navigate, refresh])

  const startPlanCheckout = async (metodo: 'card' | 'pix') => {
    if (!usuarioId) return
    const plan = selectedPlan
    if (!plan || plan === 'free') {
      setError('Selecione um plano válido.')
      return
    }

    const requested = plan === 'pro' ? Math.floor(effectiveFuncionariosTotal || includedPro) : 1
    if (plan === 'pro' && requested > maxPro) {
      setUserSelectedPlan('enterprise')
      setFuncionariosTotal(null)
      setError('Para mais de 6 profissionais, selecione o plano EMPRESA.')
      return
    }

    const total = plan === 'pro' ? Math.max(includedPro, Math.min(maxPro, requested)) : 1

    setCreatingCheckout(true)
    setError(null)
    const res = await createCheckoutPagamento(usuarioId, plan, metodo, total)
    if (!res.ok) {
      setError(formatCheckoutError(res.status, res.body))
      setCreatingCheckout(false)
      return
    }
    const body = res.body as Record<string, unknown>
    const url = typeof body.url === 'string' ? body.url : null
    if (!url) {
      setError('A função não retornou o link de checkout.')
      setCreatingCheckout(false)
      return
    }
    window.location.href = url
  }

  const openBillingPortal = async () => {
    if (!usuarioId) return
    setOpeningPortal(true)
    setError(null)
    const res = await createBillingPortalPagamento(usuarioId)
    if (!res.ok) {
      setError(formatCheckoutError(res.status, res.body))
      setOpeningPortal(false)
      return
    }
    const body = res.body as Record<string, unknown>
    const url = typeof body.url === 'string' ? body.url : null
    if (!url) {
      setError('A função não retornou o link do portal.')
```

```
      setOpeningPortal(false)
      return
    }
    window.location.href = url
  }

  const startServiceCheckout = async (metodo: 'card' | 'pix') => {
    if (!usuarioId || !selectedService) return
    setCreatingCheckout(true)
    setError(null)
    const res = await createCheckoutPagamento(usuarioId, selectedService, metodo)
    if (!res.ok) {
      setError(formatCheckoutError(res.status, res.body))
      setCreatingCheckout(false)
      return
    }
    const body = res.body as Record<string, unknown>
    const url = typeof body.url === 'string' ? body.url : null
    if (!url) {
      setError('A função não retornou o link de checkout.')
      setCreatingCheckout(false)
      return
    }
    window.location.href = url
  }

  const formatStatusPagamento = (value: string) => {
    const v = String(value ?? '').trim().toLowerCase()
    if (!v) return '—'
    if (v === 'ativo') return 'Ativo'
    if (v === 'trial') return 'Trial'
    if (v === 'inadimplente') return 'Inadimplente'
    if (v === 'suspenso') return 'Suspenso'
    if (v === 'cancelado') return 'Cancelado'
    return value
  }

  if (!usuarioId || !usuario) {
    return (
      <AppShell>
        <div className="text-slate-700">{appPrincipal ? 'Acesso restrito.' : 'Carregando…'}</div>
      </AppShell>
    )
  }

  const statusTone = usuario.status_pagamento === 'inadimplente' ? 'red' : usuario.status_pagamento === 'ativo' ?
'green' : 'slate'

  const planoLabel = (planoRaw: string) => {
    const p = String(planoRaw ?? '').trim().toLowerCase()
    if (p === 'enterprise') return 'EMPRESA'
    if (p === 'team') return 'PRO'
    if (p === 'pro') return 'PRO'
    if (p === 'basic') return 'BASIC'
    if (p === 'free') return 'FREE'
    return planoRaw
  }

  return (
    <PageTutorial usuarioId={usuarioId} page="pagamento">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="space-y-6">
            <div className="flex items-center justify-between gap-3">
              <div>
                <div className="text-sm font-semibold text-slate-500">Configurações</div>
                <div className="text-xl font-semibold text-slate-900">Pagamento e planos</div>
              </div>
              <Button variant="secondary" onClick={resetTutorial}>
                Rever tutorial
              </Button>
            </div>
```

```
        {checkoutNotice ? (
          <div
            className={[
              'rounded-xl border p-4 text-sm',
              checkoutNotice.kind === 'success' ? 'border-emerald-200 bg-emerald-50 text-emerald-800' : 'border-amber-200 bg-amber-50 text-amber-900',
            ].join(' ')}
          >
            <div className="flex flex-wrap items-center justify-between gap-3">
              <div className="space-y-1">
                <div className="font-semibold">{checkoutNotice.kind === 'success' ? 'Pagamento concluído' : 'Pagamento cancelado'}</div>
                <div>
                  {checkoutNotice.item ? `Item: ${checkoutNotice.item.toUpperCase()}. ` : ''}Status: {formatStatusPagamento(usuario.status_pagamento)}.
                </div>
              </div>
              <div className="flex items-center gap-2">
                <Button variant="secondary" onClick={() => void refresh()}>
                  Atualizar
                </Button>
                <Button variant="secondary" onClick={() => setCheckoutNotice(null)}>
                  Fechar
                </Button>
              </div>
            </div>
          </div>
        ) : null}

        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'plans'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
              <Card>
                <div className="p-6 space-y-4">
                  <div className="flex flex-wrap items-start justify-between gap-3">
                    <div>
                      <div className="text-sm font-semibold text-slate-900">Pagamento</div>
                      <div className="text-sm text-slate-600">Plano atual: {planoLabel(usuario.plano)}</div>
                    </div>
                    <div className="flex items-center gap-2">
                      <Badge tone={usuario.ativo ? 'green' : 'red'}>{usuario.ativo ? 'Conta: Ativa' : 'Conta: Inativa'}</Badge>
                      <Badge tone={statusTone}>Pagamento: {formatStatusPagamento(usuario.status_pagamento)}</Badge>
                      <Button variant="secondary" onClick={() => void refresh()}>
                        Atualizar status
                      </Button>
                    </div>
                  </div>

                  <div className="grid grid-cols-1 gap-3 sm:grid-cols-2 lg:grid-cols-3">
                    {plans.map((p) => {
                      const selected = selectedPlan === p.key
                      const clickable = p.key !== 'free'
                      const best = p.key === 'pro'
                      return (
                        <button
                          key={p.key}
                          type="button"
                          onClick={() => {
                            if (!clickable) return
                            setUserSelectedPlan(p.key)
                            setFuncionariosTotal(null)
                          }}
                          className={[
```

```
                          'text-left rounded-xl border bg-white p-4 transition',
                          best
                            ? selected
                              ? 'border-slate-900 ring-2 ring-slate-900/10 bg-amber-50'
                              : 'border-amber-300 hover:bg-amber-50'
                            : selected
                              ? 'border-slate-900 ring-2 ring-slate-900/10'
                              : 'border-slate-200 hover:bg-slate-50',
                          clickable ? '' : 'cursor-default',
                        ].join(' ')}
                      >
                        <div className="flex items-start justify-between gap-3">
                          <div>
                            <div className="text-sm font-semibold text-slate-900">{p.title}</div>
                            <div className="text-xs text-slate-600">
                              {p.priceLabel}
                              {p.subtitle ? ` • ${p.subtitle}` : ''}
                            </div>
                          </div>
                          <div className="flex items-center gap-2">
                            {best ? <Badge tone="yellow">Melhor opção</Badge> : null}
                            {selected ? <Badge tone="slate">Selecionado</Badge> : null}
                          </div>
                        </div>
                        <div className="mt-3 space-y-1">
                          {p.bullets.map((b) => (
                            <div key={b} className="text-xs text-slate-700">
                              - {b}
                            </div>
                          ))}
                        </div>
                        <div className="mt-3 text-[11px] text-slate-500">Valores de pré-venda • válido até 08/02/2026.
</div>
                      </button>
                    )
                  })}
                </div>

            {selectedPlan === 'pro' ? (
              <div className="grid grid-cols-1 gap-3 sm:grid-cols-2">
                <Input
                  label="Profissionais"
                  type="number"
                  min={includedPro}
                  max={maxPro}
                  value={String(effectiveFuncionariosTotal)}
                  onChange={(e) => {
                    const raw = e.target.value
                    const n = raw.trim() === '' ? includedPro : Number(raw)
                    if (!Number.isFinite(n)) return
                    const i = Math.floor(n)
                    if (i > maxPro) {
                      setUserSelectedPlan('enterprise')
                      setFuncionariosTotal(null)
                      setError('Para mais de 6 profissionais, selecione o plano EMPRESA.')
                      return
                    }
                    const clamped = Math.max(includedPro, Math.min(maxPro, i))
                    setFuncionariosTotal(clamped)
                  }}
                />
                <div className="rounded-xl border border-slate-200 bg-slate-50 p-4 text-xs text-slate-700 flex items-
center">
                  O checkout calcula 4 profissionais inclusos + adicional por profissional acima de 4 (máximo 6 no PRO).
                </div>
              </div>
            ) : null}

                <div
                  className={
                    tutorialOpen && tutorialSteps[tutorialStep]?.target === 'checkout'
                      ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl p-2 -m-2'
```

```
                        : ''
                    }
                  >
                    <div className="flex flex-wrap justify-end gap-2">
                      <Button variant="secondary" onClick={() => void startPlanCheckout('pix')} disabled={creatingCheckout || selectedPlan === 'free'}>
                        {creatingCheckout ? 'Abrindo…' : 'PIX (30 dias)'}
                      </Button>
                      <Button onClick={() => void startPlanCheckout('card')} disabled={creatingCheckout || selectedPlan === 'free'}>
                        {creatingCheckout ? 'Abrindo…' : 'Cartão (assinatura)'}
                      </Button>
                    </div>
                  </div>

                  {usuario.stripe_customer_id ? (
                    <div className="mt-4 rounded-xl border border-slate-200 bg-slate-50 p-4">
                      <div className="flex flex-col gap-3 sm:flex-row sm:items-center sm:justify-between">
                        <div>
                          <div className="text-sm font-semibold text-slate-900">Assinatura</div>
                          <div className="text-xs text-slate-600">Cancelar ou trocar forma de pagamento pelo Stripe.</div>
                        </div>
                        <Button variant="secondary" onClick={() => void openBillingPortal()} disabled={openingPortal}>
                          {openingPortal ? 'Abrindo…' : 'Gerenciar / Cancelar'}
                        </Button>
                      </div>
                    </div>
                  ) : usuario.status_pagamento === 'ativo' ? (
                    <div className="mt-4 rounded-xl border border-slate-200 bg-slate-50 p-4 text-xs text-slate-700">
                      Se você pagou por PIX (30 dias), não existe assinatura para cancelar. Ao vencer, basta não renovar.
                    </div>
                  ) : null}
                </div>
              </Card>
            </div>

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'services'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
              <Card>
                <div className="p-6 space-y-4">
                  <div>
                    <div className="text-sm font-semibold text-slate-900">Serviços</div>
                    <div className="text-sm text-slate-600">Contrate serviços avulsos para configuração e suporte.</div>
                  </div>

            <div className="grid grid-cols-1 gap-3 sm:grid-cols-2">
              {services.map((s) => {
                const selected = selectedService === s.key
                return (
                  <button
                    key={s.key}
                    type="button"
                    onClick={() => setSelectedService(s.key)}
                    className={[
                      'text-left rounded-xl border bg-white p-4 transition',
                      selected ? 'border-slate-900 ring-2 ring-slate-900/10' : 'border-slate-200 hover:bg-slate-50',
                    ].join(' ')}
                  >
                    <div className="flex items-start justify-between gap-3">
                      <div>
                        <div className="text-sm font-semibold text-slate-900">{s.title}</div>
                        <div className="text-xs text-slate-600">{s.priceLabel}</div>
                      </div>
                      {selected ? <Badge tone="slate">Selecionado</Badge> : null}
                    </div>
```

```
                    <div className="mt-3 space-y-1">
                      {s.bullets.map((b) => (
                        <div key={b} className="text-xs text-slate-700">
                          - {b}
                        </div>
                      ))}
                    </div>
                  </button>
                )
              })}
            </div>

                  <div className="flex flex-wrap justify-end gap-2">
                    <Button variant="secondary" onClick={() => void startServiceCheckout('pix')} disabled=
{creatingCheckout || !selectedService}>
                      {creatingCheckout ? 'Abrindo…' : 'Pagar com PIX'}
                    </Button>
                    <Button onClick={() => void startServiceCheckout('card')} disabled={creatingCheckout ||
!selectedService}>
                      {creatingCheckout ? 'Abrindo…' : 'Pagar com cartão'}
                    </Button>
                  </div>
                </div>
              </Card>
            </div>

            <TutorialOverlay
              open={tutorialOpen}
              steps={tutorialSteps}
              step={tutorialStep}
              onStepChange={setTutorialStep}
              onClose={closeTutorial}
              titleFallback="Pagamento"
            />
          </div>
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/WhatsappSettingsPage.tsx

```
import { useEffect, useMemo, useRef, useState } from 'react'
import { AppShell } from '../../components/layout/AppShell'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { Input } from '../../components/ui/Input'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { checkJwtProject, supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

function getOptionalString(payload: unknown, key: string) {
  if (!payload || typeof payload !== 'object') return null
  const value = (payload as Record<string, unknown>)[key]
  return typeof value === 'string' ? value : null
}

function formatDetails(payload: unknown) {
  if (typeof payload === 'string') {
    const s = payload.trim()
    const lower = s.toLowerCase()
    const isHtml = lower.includes('<!doctype html') || lower.includes('<html') || lower.includes('<head')
    const isCloudflare =
      lower.includes('cloudflare tunnel error') ||
      (lower.includes('cloudflare') && (lower.includes('cf-ray') || lower.includes('tunnel') ||
lower.includes('cloudflareapps')))
    if (isCloudflare) return 'Cloudflare Tunnel indisponível. Verifique Cloudflare Tunnel ativo e URL da Evolution API.'
    if (isHtml) return 'Resposta HTML inesperada ao chamar o WhatsApp. Verifique proxy/túnel e a URL configurada.'
    if (s.length > 900) return `${s.slice(0, 900)}…`
```

```
    return s
  }
  if (payload && typeof payload === 'object') {
    const msg = getOptionalString(payload, 'message') ?? getOptionalString(payload, 'details') ??
getOptionalString(payload, 'error')
    if (msg) return formatDetails(msg)
  }
  try {
    const s = JSON.stringify(payload)
    if (s.length > 900) return `${s.slice(0, 900)}…`
    return s
  } catch {
    return String(payload)
  }
}

function isWorkerLimitResponse(args: { status: number; body: unknown }) {
  if (args.status === 546) return true
  if (!args.body || typeof args.body !== 'object') return false
  return (args.body as Record<string, unknown>).code === 'WORKER_LIMIT'
}

async function callWhatsappFunction(body: unknown, opts?: { timeoutMs?: number }) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } }
  }

  const supabaseUrl = supabaseEnv.values.VITE_SUPABASE_URL
  const supabaseAnonKey = supabaseEnv.values.VITE_SUPABASE_ANON_KEY

  const tryRefresh = async () => {
    const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
    if (refreshErr) return null
    return refreshed.session ?? null
  }

  const { data: sessionData } = await supabase.auth.getSession()
  let session = sessionData.session
  const now = Math.floor(Date.now() / 1000)
  const expiresAt = session?.expires_at ?? null

  if (session && expiresAt && expiresAt <= now + 60) {
    const refreshed = await tryRefresh()
    if (refreshed) session = refreshed
  }

  const accessToken=[REDACTED]
  if (!accessToken) {
    return { ok: false as const, status: 401, body: { error: 'session_expired' } }
  }

  const tokenProject = checkJwtProject(accessToken, supabaseUrl)
  if (!tokenProject.ok) {
    await supabase.auth.signOut().catch(() => undefined)
    return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: tokenProject.iss, expected:
tokenProject.expectedPrefix } }
  }

  const callFetch = async (token: string) => {
    const fnUrl = `${supabaseUrl}/functions/v1/whatsapp`
    const controller = new AbortController()
    const timeoutId = setTimeout(() => controller.abort(), opts?.timeoutMs ?? 45000)
    try {
      const headers: Record<string, string> = {
        'Content-Type': 'application/json',
        apikey=[REDACTED]
        Authorization: `Bearer ${token}`,
        'x-user-jwt': token,
      }
      const res = await fetch(fnUrl, {
        method: 'POST',
        headers,
```

```
      body: JSON.stringify(body ?? {}),
      signal: controller.signal,
    })

    const fnVersion = res.headers.get('x-smagenda-fn')

    let text = ''
    try {
      text = await res.text()
    } catch (e: unknown) {
      const msg = e instanceof Error ? e.message : 'Falha de rede'
      return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
    }

    let parsed: unknown = null
    try {
      parsed = text ? JSON.parse(text) : null
    } catch {
      parsed = text
    }

    if (!res.ok && res.status === 401 && !fnVersion) {
      if (
        parsed &&
        typeof parsed === 'object' &&
        (parsed as Record<string, unknown>).message === 'Invalid JWT' &&
        (parsed as Record<string, unknown>).code === 401
      ) {
        return { ok: false as const, status: 401, body: { error: 'supabase_gateway_invalid_jwt' } }
      }
    }

    if (!res.ok) return { ok: false as const, status: res.status, body: parsed }
    return { ok: true as const, status: res.status, body: parsed }
  } catch (e: unknown) {
    const errAny = e as { name?: unknown; message?: unknown }
    const name = typeof errAny.name === 'string' ? errAny.name : ''
    const msg = typeof errAny.message === 'string' ? errAny.message : 'Falha de rede'
    const lower = msg.toLowerCase()
    const isAbort = name === 'AbortError' || lower.includes('aborted') || lower.includes('abort')
    if (isAbort) return { ok: false as const, status: 0, body: { error: 'timeout' } }
    return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
  } finally {
    clearTimeout(timeoutId)
  }
}

const isInvalidJwtPayload = (payload: unknown) => {
  if (typeof payload === 'string') return payload.includes('Invalid JWT')
  if (!payload || typeof payload !== 'object') return false
  const obj = payload as Record<string, unknown>
  return obj.message === 'Invalid JWT' || obj.error === 'invalid_jwt'
}

const first = await callFetch(accessToken)

const action = (() => {
  if (!body || typeof body !== 'object') return null
  const raw = (body as Record<string, unknown>).action
  return typeof raw === 'string' ? raw : null
})()

if (!first.ok && first.status === 0 && action === 'status') {
  await new Promise((r) => setTimeout(r, 300))
  return callFetch(accessToken)
}

if (
  !first.ok &&
  first.status === 401 &&
  typeof first.body === 'object' &&
  first.body !== null &&
```

```
    (first.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
  ) {
    return first
  }

  if (!first.ok && first.status === 401 && isInvalidJwtPayload(first.body)) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }

    const second = await callFetch(nextToken)
    if (!second.ok && second.status === 401 && isInvalidJwtPayload(second.body)) {
      return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    }
    return second
  }

  return first
}

export function WhatsappSettingsPage() {
  const { appPrincipal } = useAuth()
  const usuarioId = appPrincipal?.kind === 'usuario' ? appPrincipal.profile.id : null

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Status e pré-requisitos',
          body: 'Verifique se o recurso está habilitado e se a configuração do WhatsApp está OK. Se estiver pendente, o
Super Admin precisa concluir a configuração.',
          target: 'status' as const,
        },
        {
          title: 'Conectar com QR Code',
          body: 'Clique em "Conectar (QR Code)" e escaneie o QR Code no WhatsApp para vincular a instância.',
          target: 'qr' as const,
        },
        {
          title: 'Testar envio',
          body: 'Envie uma mensagem de teste para validar que está tudo funcionando em produção.',
          target: 'test' as const,
        },
      ] as const,
    []
  )

  const [configurado, setConfigurado] = useState(false)
  const [whatsappHabilitado, setWhatsappHabilitado] = useState<boolean | null>(null)
  const [loading, setLoading] = useState(true)
  const [connecting, setConnecting] = useState(false)
  const [disconnecting, setDisconnecting] = useState(false)
  const [checkingStatus, setCheckingStatus] = useState(false)

  const [sendingTest, setSendingTest] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [saved, setSaved] = useState(false)

  const [instanceState, setInstanceState] = useState<string | null>(null)
  const [qrBase64, setQrBase64] = useState<string | null>(null)
  const [testNumber, setTestNumber] = useState('')
  const [testText, setTestText] = useState('Olá! Teste de envio do SMagenda.')

  const aliveRef = useRef(true)
  const allowStatusRef = useRef(false)

  useEffect(() => {
    aliveRef.current = true
    return () => {
      aliveRef.current = false
    }
  }, [])
```

```
  const habilitado = useMemo(() => (whatsappHabilitado === null ? true : Boolean(whatsappHabilitado)), [whatsappHabilitado])

  const callWhatsapp = async (body: unknown) => {
    const action = (() => {
      if (!body || typeof body !== 'object') return null
      const raw = (body as Record<string, unknown>).action
      return typeof raw === 'string' ? raw : null
    })()

    if (action === 'status' && !allowStatusRef.current) {
      return { ok: false as const, status: 0, body: { error: 'status_not_allowed' } }
    }

    return callWhatsappFunction(body)
  }

  useEffect(() => {
    const run = async () => {
      if (!usuarioId) return
      setLoading(true)
      setError(null)
      setInstanceState(null)
      const isMissingColumnError = (message: string) => message.toLowerCase().includes('does not exist') && message.toLowerCase().includes('column')

      let enabled: boolean | null = null

      const first = await supabase.from('usuarios').select('whatsapp_habilitado').eq('id', usuarioId).maybeSingle()
      if (first.error) {
        if (isMissingColumnError(first.error.message)) {
          enabled = null
          setWhatsappHabilitado(null)
        } else {
          setError(first.error.message)
          setConfigurado(false)
          setLoading(false)
          return
        }
      } else {
        const row = (first.data ?? null) as unknown as { whatsapp_habilitado?: boolean | null } | null
        enabled = typeof row?.whatsapp_habilitado === 'boolean' ? row.whatsapp_habilitado : null
        setWhatsappHabilitado(enabled)
      }
      const cfg = await callWhatsappFunction({ action: 'config_status' })
      if (!cfg.ok) {
        const hint = cfg.body && typeof cfg.body === 'object' ? getOptionalString(cfg.body, 'hint') : null
        const details = typeof cfg.body === 'string' ? cfg.body : JSON.stringify(cfg.body)
        setError(hint ? `Falha ao carregar configuração do WhatsApp: ${details} | Dica: ${hint}` : `Falha ao carregar configuração do WhatsApp: ${details}`)
        setConfigurado(false)
        setLoading(false)
        return
      }

      const configured = cfg.body && typeof cfg.body === 'object' ? Boolean((cfg.body as Record<string, unknown>).configured) : false
      setConfigurado(configured)

      const enabledSafe = enabled === null ? true : Boolean(enabled)
      if (enabledSafe && configured) {
        const quick = await callWhatsappFunction({ action: 'status' }, { timeoutMs: 6000 })
        if (!aliveRef.current) return
        if (quick.ok && quick.body && typeof quick.body === 'object') {
          const s = getOptionalString(quick.body, 'state')
          setInstanceState(s)
          if (s === 'open') {
            setQrBase64(null)
          }
        }
      }
```

```
      setLoading(false)
    }
    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar configurações')
      setConfigurado(false)
      setLoading(false)
      setCheckingStatus(false)
    })
  }, [usuarioId])

  if (!usuarioId) {
    return (
      <AppShell>
        <div className="text-slate-700">{appPrincipal ? 'Acesso restrito.' : 'Carregando…'}</div>
      </AppShell>
    )
  }

  const connect = async () => {
    setError(null)
    setQrBase64(null)
    if (!habilitado) {
      setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
      return
    }
    if (!configurado) {
      setError('WhatsApp ainda não foi configurado no painel do Super Admin.')
      return
    }
    setConnecting(true)
    allowStatusRef.current = true
    const res = await callWhatsapp({ action: 'connect' })
    if (!res.ok) {
      if (isWorkerLimitResponse({ status: res.status, body: res.body })) {
        setError('O Supabase está sem recursos para executar a Edge Function agora (WORKER_LIMIT). Aguarde 1–2 minutos e
tente novamente.')
        setConnecting(false)
        return
      }
      if (res.status === 0 && typeof res.body === 'object' && res.body !== null && (res.body as Record<string,
unknown>).error === 'timeout') {
        setError('Tempo esgotado ao gerar QR Code. A Edge Function pode estar temporariamente sem recursos ou
indisponível. Aguarde 1–2 minutos e tente novamente.')
        setConnecting(false)
        return
      }
      if (
        typeof res.body === 'object' &&
        res.body !== null &&
        (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
      ) {
        setError('JWT inválido para chamar a Edge Function. Saia e entre novamente. Se persistir, reimplante a função
com verify_jwt=false (--no-verify-jwt).')
        setConnecting(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'jwt_project_mismatch') {
        setError('Sessão do Supabase pertence a outro projeto. Saia e entre novamente no sistema.')
        setConnecting(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'invalid_jwt') {
        setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
        setConnecting(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'whatsapp_disabled') {
        setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
        setConnecting(false)
```

```
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error === 'whatsapp_not_configured') {
        setError('WhatsApp ainda não foi configurado no painel do Super Admin.')
        setConnecting(false)
        return
      }
      const hint = getOptionalString(res.body, 'hint')
      const details = formatDetails(res.body)
      setError(hint ? `Falha ao gerar QR Code (HTTP ${res.status}): ${details}\n\nDica: ${hint}` : `Falha ao gerar QR Code (HTTP ${res.status}): ${details}`)
      setConnecting(false)
      return
    }

    const initialState = getOptionalString(res.body, 'state')
    const initialQr = getOptionalString(res.body, 'qrBase64')
    let currentState: string | null = initialState
    if (initialState) setInstanceState(initialState)
    if (initialQr && initialQr.trim()) {
      setQrBase64(initialQr)
    }

    for (let i = 0; i < 30; i += 1) {
      if (currentState === 'open') {
        setQrBase64(null)
        setConnecting(false)
        return
      }
      await new Promise((r) => setTimeout(r, 4000))
      if (!aliveRef.current) return
      const next = await callWhatsapp({ action: 'status' })
      if (!next.ok) {
        if (isWorkerLimitResponse({ status: next.status, body: next.body })) {
          setError('O Supabase está sem recursos para executar a Edge Function agora (WORKER_LIMIT). Aguarde 1–2 minutos e tente novamente.')
          setConnecting(false)
          return
        }
        if (next.status === 0 && typeof next.body === 'object' && next.body !== null && (next.body as Record<string, unknown>).error === 'timeout') {
          setError('Tempo esgotado ao verificar status do WhatsApp. A Edge Function pode estar temporariamente sem recursos ou indisponível. Aguarde 1–2 minutos e tente novamente.')
          setConnecting(false)
          return
        }
        const hint = getOptionalString(next.body, 'hint')
        const details = formatDetails(next.body)
        setError(hint ? `Falha ao verificar status (HTTP ${next.status}): ${details}\n\nDica: ${hint}` : `Falha ao verificar status (HTTP ${next.status}): ${details}`)
        setConnecting(false)
        return
      }
      const nextState = getOptionalString(next.body, 'state')
      if (nextState) {
        currentState = nextState
        setInstanceState(nextState)
      }
      if (currentState === 'open') {
        setQrBase64(null)
        setConnecting(false)
        return
      }
    }

    const hint = getOptionalString(res.body, 'hint')
    if (hint) {
      setError(hint)
    } else {
      setError('A instância ainda não conectou. Se o QR Code expirou, gere um novo e tente novamente.')
    }
```

```
    setConnecting(false)
  }

  const disconnect = async () => {
    setError(null)
    if (!habilitado) return
    if (!configurado) return
    setDisconnecting(true)
    const res = await callWhatsappFunction({ action: 'disconnect' })
    if (!res.ok) {
      if (
        typeof res.body === 'object' &&
        res.body !== null &&
        (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
      ) {
        setError('A Edge Function "whatsapp" está exigindo JWT no Supabase. Refaça o deploy com verify_jwt=false e tente
novamente.')
        setDisconnecting(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'invalid_jwt') {
        setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
        setDisconnecting(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'whatsapp_disabled') {
        setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
        setDisconnecting(false)
        return
      }
      const details = typeof res.body === 'string' ? res.body : JSON.stringify(res.body)
      setError(`Falha ao desconectar (HTTP ${res.status}): ${details}`)
      setDisconnecting(false)
      return
    }
    setInstanceState(null)
    setQrBase64(null)
    setDisconnecting(false)
  }

  const checkStatus = async () => {
    setError(null)
    if (!habilitado) {
      setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
      return
    }
    if (!configurado) {
      setError('WhatsApp ainda não foi configurado no painel do Super Admin.')
      return
    }
    setCheckingStatus(true)
    allowStatusRef.current = true
    const res = await callWhatsapp({ action: 'status' })
    if (!res.ok) {
      if (isWorkerLimitResponse({ status: res.status, body: res.body })) {
        setError('O Supabase está sem recursos para executar a Edge Function agora (WORKER_LIMIT). Aguarde 1–2 minutos e
tente novamente.')
        setCheckingStatus(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'timeout') {
        setError('Tempo esgotado ao verificar status do WhatsApp. A Edge Function pode estar temporariamente sem
recursos ou indisponível. Aguarde 1–2 minutos e tente novamente.')
        setCheckingStatus(false)
        return
      }
      if (
        typeof res.body === 'object' &&
        res.body !== null &&
```

```
        (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
      ) {
        setError('JWT inválido para chamar a Edge Function. Saia e entre novamente. Se persistir, reimplante a função
com verify_jwt=false (--no-verify-jwt).')
        setCheckingStatus(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'jwt_project_mismatch') {
        setError('Sessão do Supabase pertence a outro projeto. Saia e entre novamente no sistema.')
        setCheckingStatus(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'invalid_jwt') {
        setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
        setCheckingStatus(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'whatsapp_disabled') {
        setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
        setCheckingStatus(false)
        return
      }
      const hint = getOptionalString(res.body, 'hint')
      const details = formatDetails(res.body)
      setError(hint ? `Falha ao atualizar status (HTTP ${res.status}): ${details}\n\nDica: ${hint}` : `Falha ao
atualizar status (HTTP ${res.status}): ${details}`)
      setCheckingStatus(false)
      return
    }
    const nextState = getOptionalString(res.body, 'state')
    setInstanceState(nextState)
    if (nextState === 'open') {
      setQrBase64(null)
    }
    setCheckingStatus(false)
  }

  const sendTest = async () => {
    setError(null)
    if (!habilitado) {
      setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
      return
    }
    if (!configurado) {
      setError('WhatsApp ainda não foi configurado no painel do Super Admin.')
      return
    }
    if (!testNumber.trim() || !testText.trim()) {
      setError('Informe número e mensagem.')
      return
    }
    setSendingTest(true)
    const res = await callWhatsappFunction({ action: 'send_test', number: testNumber, text: testText })
    if (!res.ok) {
      if (
        typeof res.body === 'object' &&
        res.body !== null &&
        (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
      ) {
        setError('A Edge Function "whatsapp" está exigindo JWT no Supabase. Refaça o deploy com verify_jwt=false e tente
novamente.')
        setSendingTest(false)
        return
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'invalid_jwt') {
        setError('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
        setSendingTest(false)
        return
```

```
      }
      if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error ===
'whatsapp_disabled') {
        setError('WhatsApp desabilitado para sua conta. Solicite habilitação ao suporte.')
        setSendingTest(false)
        return
      }
      const hint = getOptionalString(res.body, 'hint')
      const details = formatDetails(res.body)
      const attemptsText = (() => {
        const raw = (res.body as Record<string, unknown> | null)?.attempts
        if (!raw) return null
        try {
          return JSON.stringify(raw)
        } catch {
          return null
        }
      })()
      setError(
        hint
          ? `Falha ao enviar teste (HTTP ${res.status}): ${details}\n\nDica: ${hint}${attemptsText ? `\n\nTentativas:
${attemptsText}` : ''}`
          : `Falha ao enviar teste (HTTP ${res.status}): ${details}${attemptsText ? `\n\nTentativas: ${attemptsText}` :
''}`
      )
      setSendingTest(false)
      return
    }
    setSendingTest(false)
    setSaved(true)
    setTimeout(() => setSaved(false), 2000)
  }

  const isConnected = instanceState?.toLowerCase() === 'open'
  const connectionLabel = (() => {
    if (!habilitado || !configurado) return '—'
    if (checkingStatus) return 'verificando…'
    if (isConnected) return 'Conectado'
    if (!instanceState) return 'Desconectado'
    const s = instanceState.toLowerCase()
    if (s === 'close' || s === 'closed') return 'Desconectado'
    if (s === 'connecting') return 'Conectando…'
    return instanceState
  })()

  return (
    <PageTutorial usuarioId={usuarioId} page="whatsapp">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="space-y-6">
            <div className="flex items-center justify-between">
              <div>
                <div className="text-sm font-semibold text-slate-500">Configurações</div>
                <div className="text-xl font-semibold text-slate-900">WhatsApp</div>
              </div>
              <Button variant="secondary" onClick={resetTutorial}>
                Rever tutorial
              </Button>
            </div>

        {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> :
null}
        {saved ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-
800">Enviado.</div> : null}

        <div
          className={
            tutorialOpen && tutorialSteps[tutorialStep]?.target === 'status'
              ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
              : ''
          }
        >
```

```
          <Card>
            <div className="p-6 space-y-4">
              <div className="text-sm font-semibold text-slate-900">Status</div>
              {loading ? (
                <div className="text-sm text-slate-600">Carregando…</div>
              ) : (
                <div className="space-y-2">
                  <div className="text-sm text-slate-700">Recurso: {habilitado ? '🟢 Habilitado' : '🔴 Desabilitado'}</div>
                  <div className="text-sm text-slate-700">Configuração: {configurado ? '🟢 OK' : '🟡 Pendente'}</div>
                  <div className="text-sm text-slate-700">Conexão: {connectionLabel}</div>
                  <div className="flex flex-wrap gap-2">
                    <Button variant="secondary" onClick={checkStatus} disabled={!habilitado || !configurado || checkingStatus || connecting || disconnecting}>
                      {isConnected ? 'Atualizar status' : 'Verificar status'}
                    </Button>
                    {isConnected ? (
                      <Button onClick={checkStatus} disabled={!habilitado || !configurado || checkingStatus || connecting || disconnecting}>
                        Testar conexão
                      </Button>
                    ) : (
                      <Button onClick={() => void connect()} disabled={!habilitado || !configurado || checkingStatus || connecting || disconnecting}>
                        Conectar (QR Code)
                      </Button>
                    )}
                    <Button variant="danger" onClick={disconnect} disabled={!habilitado || !configurado || disconnecting || connecting}>
                      Desconectar
                    </Button>
                  </div>
                </div>
              )}
            </div>
          </Card>
        </div>

        {qrBase64 ? (
          <div
            className={
              tutorialOpen && tutorialSteps[tutorialStep]?.target === 'qr'
                ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                : ''
            }
          >
            <Card>
              <div className="p-6 space-y-4">
                <div className="text-sm font-semibold text-slate-900">QR Code</div>
                <div className="text-sm text-slate-600">Escaneie no WhatsApp para conectar a instância.</div>
                <div className="flex justify-center">
                  <img
                    className="h-64 w-64 rounded-xl border border-slate-200 bg-white"
                    src={qrBase64.trim().toLowerCase().startsWith('data:') ? qrBase64 : `data:image/png;base64,${qrBase64}`}
                    alt="QR Code"
                  />
                </div>
              </div>
            </Card>
          </div>
        ) : null}

        <div
          className={
            tutorialOpen && tutorialSteps[tutorialStep]?.target === 'test'
              ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
              : ''
          }
        >
          <Card>
            <div className="p-6 space-y-4">
```

```
              <div className="text-sm font-semibold text-slate-900">Testar envio</div>
              <Input label="Número (com DDD)" value={testNumber} onChange={(e) => setTestNumber(e.target.value)}
placeholder="11 99999-9999" />
              <div className="space-y-2">
                <div className="text-sm font-medium text-slate-700">Mensagem</div>
                <textarea
                  className="w-full min-h-[120px] rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-
slate-900 outline-none focus:ring-2 focus:ring-slate-300"
                  value={testText}
                  onChange={(e) => setTestText(e.target.value)}
                  disabled={sendingTest}
                />
              </div>
              <div className="flex justify-end">
                <Button onClick={sendTest} disabled={!habilitado || !configurado || sendingTest || connecting ||
disconnecting}>
                  Enviar teste
                </Button>
              </div>
            </div>
          </Card>
        </div>

            <TutorialOverlay open={tutorialOpen} steps={tutorialSteps} step={tutorialStep} onStepChange=
{setTutorialStep} onClose={closeTutorial} />
          </div>
        </AppShell>
      )}
    </PageTutorial>
  )
}
```

smagenda/src/views/app/MensagensSettingsPage.tsx

```
import { useNavigate } from 'react-router-dom'
import { useEffect, useMemo, useRef, useState } from 'react'
import { AppShell } from '../../components/layout/AppShell'
import { Button } from '../../components/ui/Button'
import { Card } from '../../components/ui/Card'
import { PageTutorial, TutorialOverlay } from '../../components/ui/TutorialOverlay'
import { checkJwtProject, supabase, supabaseEnv } from '../../lib/supabase'
import { useAuth } from '../../state/auth/useAuth'

const defaultConfirmacao = `Olá {nome}!\n\nSeu agendamento foi confirmado:\n📅 {data} às {hora}\n✂️ {servico}\n💰
{preco}\n\nLocal: {endereco}\n\nNos vemos em breve!\n{nome_negocio}`
const defaultLembrete = `Oi {nome}!\n\nLembrete: você tem agendamento em {data} às {hora}.\n\nSe não puder comparecer,
me avise!\n{telefone_profissional}`
const defaultCancelamento = `Olá {nome}!\n\nSeu agendamento foi cancelado:\n📅 {data} às {hora}\n✂️ {servico}\n\nSe
quiser remarcar, é só me chamar.\n{nome_negocio}`

type TemplatePreset = {
  key: string
  title: string
  confirmacao: string
  lembrete: string
  cancelamento: string
}

const templatePresets: TemplatePreset[] = [
  {
    key: 'padrao_servicos',
    title: 'Padrão (serviços)',
    confirmacao: defaultConfirmacao,
    lembrete: defaultLembrete,
    cancelamento: defaultCancelamento,
  },
  {
    key: 'lava_jatos',
    title: 'Lava-jato',
    confirmacao:
```

```
      `Olá {nome}!\n\n✅ Sua lavagem está confirmada:\n📅 {data} às {hora}\n🚗 {servico}\n💰 {preco}\n\nLocal: {unidade_nome}\n{unidade_endereco}\n\nDúvidas/alterações: {telefone_profissional}\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete da sua lavagem:\n📅 {data} às {hora}\n🚗 {servico}\n\nLocal: {unidade_nome}\n{unidade_endereco}\n\nSe precisar remarcar: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Sua lavagem foi cancelada:\n📅 {data} às {hora}\n🚗 {servico}\n\nSe quiser remarcar, fale com a gente: {telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'barbearia',
    title: 'Barbearia',
    confirmacao:
      `Olá {nome}!\n\n✅ Seu horário está confirmado:\n📅 {data} às {hora}\n✂️ {servico}\n💰 {preco}\n\nBarbeiro: {profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nAté já!\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete do seu horário:\n📅 {data} às {hora}\n✂️ {servico}\n\nBarbeiro: {profissional_nome}\n\nSe precisar remarcar: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Seu horário foi cancelado:\n📅 {data} às {hora}\n✂️ {servico}\n\nSe quiser remarcar, chama aqui: {telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'salao',
    title: 'Salão de beleza',
    confirmacao:
      `Olá {nome}!\n\n✅ Seu horário está confirmado:\n📅 {data} às {hora}\n💇‍♀️ {servico}\n\nProfissional: {profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nAté já!\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete do seu horário:\n📅 {data} às {hora}\n💇‍♀️ {servico}\nProfissional: {profissional_nome}\n\nSe precisar remarcar, fale com a gente: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Seu horário foi cancelado:\n📅 {data} às {hora}\n💇‍♀️ {servico}\n\nSe quiser remarcar, fale com a gente: {telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'estetica',
    title: 'Estética',
    confirmacao:
      `Olá {nome}!\n\n✅ Seu atendimento está confirmado:\n📅 {data} às {hora}\n✨ {servico}\n\nProfissional: {profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nAté breve!\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n✨ Lembrete do seu atendimento:\n📅 {data} às {hora}\n✨ {servico}\n\nQualquer ajuste: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Seu atendimento foi cancelado:\n📅 {data} às {hora}\n✨ {servico}\n\nSe quiser remarcar, é só me chamar: {telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'odontologia',
    title: 'Odontologia',
    confirmacao:
      `Olá {nome}!\n\n✅ Sua consulta está confirmada:\n📅 {data} às {hora}\n🦷 {servico}\n\nDentista: {profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nQualquer dúvida: {telefone_profissional}\n{nome_negocio}`,
    lembrete:
      `Olá {nome}!\n\n🦷 Lembrete da sua consulta:\n📅 {data} às {hora}\nProcedimento: {servico}\n\nSe precisar remarcar: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Sua consulta foi cancelada:\n📅 {data} às {hora}\n🦷 {servico}\n\nSe quiser remarcar: {telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'manicure',
    title: 'Manicure',
    confirmacao:
      `Olá {nome}!\n\n✅ Seu horário está confirmado:\n📅 {data} às {hora}\n💅 {servico}\n💰 {preco}\n\nProfissional: {profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nAté já!\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete do seu horário:\n📅 {data} às {hora}\n💅 {servico}\n\nSe precisar remarcar: {telefone_profissional}`,
    cancelamento:
```

```
      `Olá {nome}!\n\n❌ Seu horário foi cancelado:\n📅 {data} às {hora}\n💅 {servico}\n\nSe quiser remarcar:
{telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'pilates',
    title: 'Pilates / Estúdio',
    confirmacao:
      `Olá {nome}!\n\n✅ Sua aula está confirmada:\n📅 {data} às {hora}\n🏋️ {servico}\n\nInstrutor:
{profissional_nome}\nLocal: {unidade_nome}\n{unidade_endereco}\n\nAté lá!\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete da sua aula:\n📅 {data} às {hora}\n🏋️ {servico}\n\nSe precisar remarcar:
{telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Sua aula foi cancelada:\n📅 {data} às {hora}\n🏋️ {servico}\n\nSe quiser remarcar:
{telefone_profissional}\n{nome_negocio}`,
  },
  {
    key: 'faxina',
    title: 'Faxina / Diarista',
    confirmacao:
      `Olá {nome}!\n\n✅ Sua diária está confirmada:\n📅 {data} às {hora}\n🧹 {servico}\n💰 {preco}\n\nEndereço do
cliente:\n{cliente_endereco}\n\nProfissional: {profissional_nome}\n\nQualquer ajuste:
{telefone_profissional}\n{nome_negocio}`,
    lembrete:
      `Oi {nome}!\n\n⏰ Lembrete da sua diária:\n📅 {data} às {hora}\n🧹 {servico}\n\nEndereço do
cliente:\n{cliente_endereco}\n\nSe precisar remarcar: {telefone_profissional}`,
    cancelamento:
      `Olá {nome}!\n\n❌ Sua diária foi cancelada:\n📅 {data} às {hora}\n🧹 {servico}\n\nSe quiser remarcar, é só me
chamar: {telefone_profissional}\n{nome_negocio}`,
  },
]

const templateVars: Array<{ key: string; label: string }> = [
  { key: 'nome', label: 'Cliente' },
  { key: 'data', label: 'Data' },
  { key: 'hora', label: 'Hora' },
  { key: 'servico', label: 'Serviço' },
  { key: 'preco', label: 'Preço' },
  { key: 'cliente_endereco', label: 'Endereço cliente' },
  { key: 'profissional_nome', label: 'Profissional' },
  { key: 'telefone_profissional', label: 'Telefone' },
  { key: 'unidade_nome', label: 'Unidade' },
  { key: 'unidade_endereco', label: 'Endereço unidade' },
  { key: 'unidade_telefone', label: 'Telefone unidade' },
  { key: 'endereco', label: 'Endereço (fallback)' },
  { key: 'nome_negocio', label: 'Nome do negócio' },
]

type AgendamentoConfirmadoRow = {
  id: string
  data: string
  hora_inicio: string | null
  cliente_nome: string | null
  cliente_telefone: string | null
  status?: string | null
  confirmacao_enviada?: boolean | null
}

async function sendConfirmacaoWhatsapp(agendamentoId: string) {
  if (!supabaseEnv.ok) {
    return { ok: false as const, status: 0, body: { error: 'missing_supabase_env' } }
  }

  const supabaseUrl = supabaseEnv.values.VITE_SUPABASE_URL
  const supabaseAnonKey = supabaseEnv.values.VITE_SUPABASE_ANON_KEY

  const tryRefresh = async () => {
    const { data: refreshed, error: refreshErr } = await supabase.auth.refreshSession()
    if (refreshErr) return null
    return refreshed.session ?? null
  }
```

```
  const { data: sessionData } = await supabase.auth.getSession()
  let session = sessionData.session
  const now = Math.floor(Date.now() / 1000)
  const expiresAt = session?.expires_at ?? null

  if (session && expiresAt && expiresAt <= now + 60) {
    const refreshed = await tryRefresh()
    if (refreshed) session = refreshed
  }

  const token = session?.access_token ?? null
  if (!token) {
    return { ok: false as const, status: 401, body: { error: 'session_expired' } }
  }

  const tokenProject = checkJwtProject(token, supabaseUrl)
  if (!tokenProject.ok) {
    await supabase.auth.signOut().catch(() => undefined)
    return { ok: false as const, status: 401, body: { error: 'jwt_project_mismatch', iss: tokenProject.iss, expected:
tokenProject.expectedPrefix } }
  }

  const callFetch = async (jwt: string) => {
    const fnUrl = `${supabaseUrl}/functions/v1/whatsapp`
    let res: Response
    try {
      res = await fetch(fnUrl, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          apikey=[REDACTED]
          Authorization: `Bearer ${jwt}`,
          'x-user-jwt': jwt,
        },
        body: JSON.stringify({ action: 'send_confirmacao', agendamento_id: agendamentoId }),
      })
    } catch (e: unknown) {
      const msg = e instanceof Error ? e.message : 'Falha de rede'
      return { ok: false as const, status: 0, body: { error: 'network_error', message: msg } }
    }

    const fnVersion = res.headers.get('x-smagenda-fn')

    const text = await res.text()
    let parsed: unknown = null
    try {
      parsed = text ? JSON.parse(text) : null
    } catch {
      parsed = text
    }

    if (!res.ok && res.status === 401 && !fnVersion) {
      if (
        parsed &&
        typeof parsed === 'object' &&
        (parsed as Record<string, unknown>).message === 'Invalid JWT' &&
        (parsed as Record<string, unknown>).code === 401
      ) {
        return { ok: false as const, status: 401, body: { error: 'supabase_gateway_invalid_jwt' } }
      }
    }

    if (!res.ok) return { ok: false as const, status: res.status, body: parsed }
    return { ok: true as const, status: res.status, body: parsed }
  }

  const isInvalidJwtPayload = (payload: unknown) => {
    if (typeof payload === 'string') return payload.includes('Invalid JWT')
    if (!payload || typeof payload !== 'object') return false
    const obj = payload as Record<string, unknown>
    return obj.message === 'Invalid JWT' || obj.error === 'invalid_jwt'
  }
```

```
  const first = await callFetch(token)
  if (
    !first.ok &&
    first.status === 401 &&
    typeof first.body === 'object' &&
    first.body !== null &&
    (first.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
  ) {
    return first
  }

  if (!first.ok && first.status === 401 && isInvalidJwtPayload(first.body)) {
    const refreshed = await tryRefresh()
    const nextToken = refreshed?.access_token ?? null
    if (!nextToken) return { ok: false as const, status: 401, body: { error: 'invalid_jwt' } }
    return callFetch(nextToken)
  }

  return first
}

export function MensagensSettingsPage() {
  const { appPrincipal } = useAuth()
  const navigate = useNavigate()
  const usuarioId = appPrincipal?.kind === 'usuario' ? appPrincipal.profile.id : null
  const canEditTemplates = useMemo(() => appPrincipal?.kind === 'usuario' && Boolean(usuarioId), [appPrincipal?.kind,
usuarioId])
  const canTestConfirmacao = Boolean(usuarioId)

  const tutorialSteps = useMemo(
    () =>
      [
        {
          title: 'Ative as automações',
          body: 'Ligue/desligue confirmação e lembrete automático. Ajuste as horas do lembrete conforme seu fluxo.',
          target: 'prefs' as const,
        },
        {
          title: 'Edite os templates',
          body: 'Personalize as mensagens e use variáveis como {nome}, {data} e {hora}.',
          target: 'templates' as const,
        },
        {
          title: 'Teste com um agendamento',
          body: 'Selecione um agendamento e envie uma confirmação para validar o WhatsApp em produção.',
          target: 'test' as const,
        },
      ] as const,
    []
  )

  const [mensagemConfirmacao, setMensagemConfirmacao] = useState(defaultConfirmacao)
  const [mensagemLembrete, setMensagemLembrete] = useState(defaultLembrete)
  const [mensagemCancelamento, setMensagemCancelamento] = useState(defaultCancelamento)
  const [presetKey, setPresetKey] = useState(templatePresets[0]?.key ?? 'padrao_servicos')
  const [lastField, setLastField] = useState<'confirmacao' | 'lembrete' | 'cancelamento'>('confirmacao')
  const confirmacaoRef = useRef<HTMLTextAreaElement | null>(null)
  const lembreteRef = useRef<HTMLTextAreaElement | null>(null)
  const cancelamentoRef = useRef<HTMLTextAreaElement | null>(null)
  const [enviarConfirmacao, setEnviarConfirmacao] = useState(true)
  const [enviarLembrete, setEnviarLembrete] = useState(false)
  const [enviarCancelamento, setEnviarCancelamento] = useState(true)
  const [lembreteHorasAntes, setLembreteHorasAntes] = useState(24)
  const [loading, setLoading] = useState(true)
  const [saving, setSaving] = useState(false)
  const [error, setError] = useState<string | null>(null)
  const [saved, setSaved] = useState(false)
  const [schemaIncompleto, setSchemaIncompleto] = useState(false)

  const [agLoading, setAgLoading] = useState(false)
  const [agendamentosConfirmados, setAgendamentosConfirmados] = useState<AgendamentoConfirmadoRow[]>([])
```

```
  const [agendamentoSelecionadoId, setAgendamentoSelecionadoId] = useState('')
  const [sendingConfirmacao, setSendingConfirmacao] = useState(false)
  const [sendConfirmacaoResult, setSendConfirmacaoResult] = useState<string | null>(null)
  const [agError, setAgError] = useState<string | null>(null)

  const isMissingColumnError = (message: string) => message.toLowerCase().includes('does not exist') &&
message.toLowerCase().includes('column')

  useEffect(() => {
    const run = async () => {
      if (!usuarioId) {
        setLoading(false)
        return
      }
      setLoading(true)
      setError(null)
      setSchemaIncompleto(false)

      const baseSelect = 'mensagem_confirmacao,mensagem_lembrete,mensagem_cancelamento'
      const baseFallbackSelect = 'mensagem_confirmacao,mensagem_lembrete'
      const baseFirst = await supabase.from('usuarios').select(baseSelect).eq('id', usuarioId).maybeSingle()

      if (baseFirst.error) {
        if (!isMissingColumnError(baseFirst.error.message)) {
          setError(baseFirst.error.message)
          setLoading(false)
          return
        }
        const baseSecond = await supabase.from('usuarios').select(baseFallbackSelect).eq('id', usuarioId).maybeSingle()
        if (baseSecond.error) {
          setError(baseSecond.error.message)
          setLoading(false)
          return
        }
        const baseRow = (baseSecond.data ?? null) as unknown as {
          mensagem_confirmacao?: string | null
          mensagem_lembrete?: string | null
          mensagem_cancelamento?: string | null
        } | null
        setMensagemConfirmacao(baseRow?.mensagem_confirmacao ?? defaultConfirmacao)
        setMensagemLembrete(baseRow?.mensagem_lembrete ?? defaultLembrete)
        setMensagemCancelamento(defaultCancelamento)
      } else {
        const baseRow = (baseFirst.data ?? null) as unknown as {
          mensagem_confirmacao?: string | null
          mensagem_lembrete?: string | null
          mensagem_cancelamento?: string | null
        } | null
        setMensagemConfirmacao(baseRow?.mensagem_confirmacao ?? defaultConfirmacao)
        setMensagemLembrete(baseRow?.mensagem_lembrete ?? defaultLembrete)
        setMensagemCancelamento(baseRow?.mensagem_cancelamento ?? defaultCancelamento)
      }

      const { data: extraData, error: extraErr } = await supabase
        .from('usuarios')
        .select('enviar_confirmacao,enviar_lembrete,enviar_cancelamento,lembrete_horas_antes')
        .eq('id', usuarioId)
        .maybeSingle()

      if (extraErr) {
        if (isMissingColumnError(extraErr.message)) {
          setSchemaIncompleto(true)
          setEnviarConfirmacao(true)
          setEnviarLembrete(false)
          setEnviarCancelamento(true)
          setLembreteHorasAntes(24)
          setLoading(false)
          return
        }
        setError(extraErr.message)
        setLoading(false)
        return
```

```
      }
      const row =
        (extraData ?? null) as unknown as {
          enviar_confirmacao?: boolean | null
          enviar_lembrete?: boolean | null
          enviar_cancelamento?: boolean | null
          lembrete_horas_antes?: number | null
        } | null
      setEnviarConfirmacao(row?.enviar_confirmacao ?? true)
      setEnviarLembrete(row?.enviar_lembrete ?? false)
      setEnviarCancelamento(row?.enviar_cancelamento ?? true)
      setLembreteHorasAntes(typeof row?.lembrete_horas_antes === 'number' ? row?.lembrete_horas_antes : 24)
      setLoading(false)
    }
    run().catch((e: unknown) => {
      setError(e instanceof Error ? e.message : 'Erro ao carregar mensagens')
      setLoading(false)
    })
  }, [usuarioId])

  useEffect(() => {
    const run = async () => {
      if (!usuarioId) return
      setAgLoading(true)
      setSendConfirmacaoResult(null)
      setAgError(null)

      const baseSelect = 'id,data,hora_inicio,cliente_nome,cliente_telefone,status,confirmacao_enviada'
      const fallbackSelect = 'id,data,hora_inicio,cliente_nome,cliente_telefone,status'
      const isMissingColumnError = (message: string) => message.toLowerCase().includes('does not exist') &&
message.toLowerCase().includes('column')

      const first = await supabase
        .from('agendamentos')
        .select(baseSelect)
        .eq('usuario_id', usuarioId)
        .order('data', { ascending: false })
        .order('hora_inicio', { ascending: false })
        .limit(30)

      if (first.error) {
        if (!isMissingColumnError(first.error.message)) {
          setAgendamentosConfirmados([])
          setAgError(first.error.message)
          setAgLoading(false)
          return
        }

        const second = await supabase
          .from('agendamentos')
          .select(fallbackSelect)
          .eq('usuario_id', usuarioId)
          .order('data', { ascending: false })
          .order('hora_inicio', { ascending: false })
          .limit(30)

        if (second.error) {
          setAgendamentosConfirmados([])
          setAgError(second.error.message)
          setAgLoading(false)
          return
        }

        const list = (second.data ?? []) as unknown as AgendamentoConfirmadoRow[]
        setAgendamentosConfirmados(list)
        setAgendamentoSelecionadoId((prev) => prev || (list[0]?.id ?? ''))
        setAgLoading(false)
        return
      }

      const list = (first.data ?? []) as unknown as AgendamentoConfirmadoRow[]
      setAgendamentosConfirmados(list)
```

```
      setAgendamentoSelecionadoId((prev) => prev || (list[0]?.id ?? ''))
      setAgLoading(false)
    }

    run().catch(() => {
      setAgError('Erro ao carregar agendamentos')
      setAgLoading(false)
    })
  }, [usuarioId])

  const selectedAgendamento = useMemo(() => {
    if (!agendamentoSelecionadoId) return null
    return agendamentosConfirmados.find((a) => a.id === agendamentoSelecionadoId) ?? null
  }, [agendamentoSelecionadoId, agendamentosConfirmados])

  const save = async () => {
    if (!usuarioId) return
    setSaving(true)
    setSaved(false)
    setError(null)

    const baseUpdate = { mensagem_confirmacao: mensagemConfirmacao, mensagem_lembrete: mensagemLembrete,
mensagem_cancelamento: mensagemCancelamento }
    const first = await supabase.from('usuarios').update(baseUpdate).eq('id', usuarioId)

    if (first.error) {
      if (isMissingColumnError(first.error.message)) {
        const fallbackUpdate = { mensagem_confirmacao: mensagemConfirmacao, mensagem_lembrete: mensagemLembrete }
        const second = await supabase.from('usuarios').update(fallbackUpdate).eq('id', usuarioId)
        if (second.error) {
          setError(second.error.message)
          setSaving(false)
          return
        }
      } else {
        setError(first.error.message)
        setSaving(false)
        return
      }
    }

    const { error: extraErr } = await supabase
      .from('usuarios')
      .update({ enviar_confirmacao: enviarConfirmacao, enviar_lembrete: enviarLembrete, enviar_cancelamento:
enviarCancelamento, lembrete_horas_antes: lembreteHorasAntes })
      .eq('id', usuarioId)

    if (extraErr) {
      if (isMissingColumnError(extraErr.message)) {
        setSchemaIncompleto(true)
        setSaving(false)
        setSaved(true)
        setTimeout(() => setSaved(false), 2000)
        return
      }
      setError(extraErr.message)
      setSaving(false)
      return
    }

    setSaving(false)
    setSaved(true)
    setTimeout(() => setSaved(false), 2000)
  }

  const getPreset = (key: string) => templatePresets.find((p) => p.key === key) ?? templatePresets[0]

  const applyPreset = (target: 'confirmacao' | 'lembrete' | 'cancelamento' | 'both' | 'all', keyOverride?: string) => {
    const preset = getPreset(keyOverride ?? presetKey)
    if (!preset) return
    if (target === 'confirmacao' || target === 'both' || target === 'all') setMensagemConfirmacao(preset.confirmacao)
    if (target === 'lembrete' || target === 'both' || target === 'all') setMensagemLembrete(preset.lembrete)
```

```
    if (target === 'cancelamento' || target === 'all') setMensagemCancelamento(preset.cancelamento)
  }

  const insertVar = (key: string) => {
    const token = `{${key}}`
    const active = lastField === 'confirmacao' ? confirmacaoRef.current : lastField === 'lembrete' ? lembreteRef.current : cancelamentoRef.current
    const getter = lastField === 'confirmacao' ? mensagemConfirmacao : lastField === 'lembrete' ? mensagemLembrete : mensagemCancelamento
    const setter =
      lastField === 'confirmacao' ? setMensagemConfirmacao : lastField === 'lembrete' ? setMensagemLembrete : setMensagemCancelamento
    if (!active) {
      setter(`${getter}${getter ? ' ' : ''}${token}`)
      return
    }

    const start = active.selectionStart ?? getter.length
    const end = active.selectionEnd ?? getter.length
    const next = `${getter.slice(0, start)}${token}${getter.slice(end)}`
    setter(next)
    requestAnimationFrame(() => {
      active.focus()
      const pos = start + token.length
      active.setSelectionRange(pos, pos)
    })
  }

  if (!usuarioId) {
    return (
      <AppShell>
        <div className="text-slate-700">{appPrincipal ? 'Acesso restrito.' : 'Carregando…'}</div>
      </AppShell>
    )
  }

  return (
    <PageTutorial usuarioId={usuarioId} page="mensagens">
      {({ tutorialOpen, tutorialStep, setTutorialStep, resetTutorial, closeTutorial }) => (
        <AppShell>
          <div className="mx-auto w-full max-w-3xl space-y-6">
            <div className="flex flex-col items-start justify-between gap-3 sm:flex-row sm:items-center">
              <div>
                <div className="text-sm font-semibold text-slate-500">Configurações</div>
                <div className="text-xl font-semibold text-slate-900">Mensagens automáticas</div>
              </div>
              <Button variant="secondary" onClick={resetTutorial} className="w-full sm:w-auto">
                Rever tutorial
              </Button>
            </div>

            <div className="space-y-3">
              {error ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-700">{error}</div> : null}
              {schemaIncompleto ? (
                <div className="rounded-xl border border-amber-200 bg-amber-50 p-3 text-sm text-amber-800">
                  Seu Supabase ainda não tem as colunas de automação do WhatsApp. Execute o SQL do WhatsApp (automação) no painel de Admin.
                </div>
              ) : null}
              {saved ? <div className="rounded-xl border border-emerald-200 bg-emerald-50 p-3 text-sm text-emerald-800">Salvo.</div> : null}
            </div>

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'prefs'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
              <Card>
```

```
              <div className="space-y-4 p-4 sm:p-6">
            <div className="text-sm font-semibold text-slate-900">Preferências de envio</div>
            {loading ? (
              <div className="text-sm text-slate-600">Carregando…</div>
            ) : (
              <div className="space-y-4">
                <label className="flex items-center gap-2 text-sm text-slate-700">
                  <input
                    type="checkbox"
                    checked={enviarConfirmacao}
                    onChange={(e) => setEnviarConfirmacao(e.target.checked)}
                    disabled={!canEditTemplates || saving}
                    className="h-4 w-4"
                  />
                  Enviar confirmação ao confirmar agendamento
                </label>
                <label className="flex items-center gap-2 text-sm text-slate-700">
                  <input
                    type="checkbox"
                    checked={enviarLembrete}
                    onChange={(e) => setEnviarLembrete(e.target.checked)}
                    disabled={!canEditTemplates || saving}
                    className="h-4 w-4"
                  />
                  Enviar lembrete automático
                </label>
                <div className="space-y-2">
                  <div className="text-sm font-medium text-slate-700">Enviar lembrete X horas antes</div>
                  <div className="flex items-center gap-3">
                    <input
                      type="range"
                      min={4}
                      max={72}
                      step={1}
                      value={lembreteHorasAntes}
                      onChange={(e) => setLembreteHorasAntes(Number(e.target.value))}
                      disabled={!canEditTemplates || saving || !enviarLembrete}
                      className="w-full"
                    />
                    <div className="w-14 text-right text-sm font-semibold text-slate-900">{lembreteHorasAntes}h</div>
                  </div>
                </div>
              </div>
            )}
                </div>
              </Card>
            </div>

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'test'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
              <Card>
                <div className="space-y-4 p-4 sm:p-6">
            <div className="text-sm font-semibold text-slate-900">Validar confirmação automática</div>
            <div className="text-sm text-slate-600">Salve o template antes de testar o envio.</div>

            {agLoading ? <div className="text-sm text-slate-600">Carregando agendamentos…</div> : null}

            {!agLoading && agError ? <div className="rounded-xl border border-rose-200 bg-rose-50 p-3 text-sm text-rose-
700">{agError}</div> : null}

            {!agLoading && agendamentosConfirmados.length === 0 ? (
              <div className="space-y-3">
                <div className="text-sm text-slate-600">Nenhum agendamento encontrado.</div>
                <div>
                  <Button variant="secondary" onClick={() => navigate('/dashboard')}>
                    Ir para Agenda
                  </Button>
```

```
              </div>
            </div>
          ) : null}

          {!agLoading && agendamentosConfirmados.length > 0 ? (
            <div className="space-y-2">
              <div className="text-sm font-medium text-slate-700">Agendamento</div>
              <select
                className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900
outline-none focus:ring-2 focus:ring-slate-300"
                value={agendamentoSelecionadoId}
                onChange={(e) => setAgendamentoSelecionadoId(e.target.value)}
                disabled={!canTestConfirmacao || saving || sendingConfirmacao}
              >
                <option value="">Selecione…</option>
                {agendamentosConfirmados.map((a) => {
                  const data = a.data ? a.data.split('-').reverse().join('/') : ''
                  const hora = a.hora_inicio ?? ''
                  const nome = a.cliente_nome ?? 'Cliente'
                  const tel = a.cliente_telefone ? ` • ${a.cliente_telefone}` : ''
                  const flag = a.confirmacao_enviada ? ' • já enviada' : ''
                  const status = (a.status ?? '').trim() ? ` • ${(a.status ?? '').trim()}` : ''
                  return (
                    <option key={a.id} value={a.id}>
                      {data} {hora} • {nome}
                      {tel}
                      {status}
                      {flag}
                    </option>
                  )
                })}
              </select>
            </div>
          ) : null}

          {!agLoading && selectedAgendamento && (selectedAgendamento.status ?? '').trim().toLowerCase() !==
'confirmado' ? (
            <div className="text-sm text-amber-800 rounded-xl border border-amber-200 bg-amber-50 p-3">
              Esse agendamento ainda não está como “confirmado”. A confirmação automática só envia após confirmar.
            </div>
          ) : null}

          <div className="flex justify-end">
            <Button
              onClick={async () => {
                if (!agendamentoSelecionadoId) return
                if ((selectedAgendamento?.status ?? '').trim().toLowerCase() !== 'confirmado') return
                setSendConfirmacaoResult(null)
                setSendingConfirmacao(true)
                const res = await sendConfirmacaoWhatsapp(agendamentoSelecionadoId)
                if (!res.ok) {
                  if (
                    typeof res.body === 'object' &&
                    res.body !== null &&
                    (res.body as Record<string, unknown>).error === 'supabase_gateway_invalid_jwt'
                  ) {
                    setSendConfirmacaoResult('JWT inválido para chamar a Edge Function. Saia e entre novamente. Se
persistir, reimplante a função com verify_jwt=false (--no-verify-jwt).')
                    setSendingConfirmacao(false)
                    return
                  }
                  if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error
=== 'jwt_project_mismatch') {
                    setSendConfirmacaoResult('Sessão do Supabase pertence a outro projeto. Saia e entre novamente no
sistema.')
                    setSendingConfirmacao(false)
                    return
                  }
                  if (typeof res.body === 'object' && res.body !== null && (res.body as Record<string, unknown>).error
=== 'invalid_jwt') {
                    setSendConfirmacaoResult('Sessão inválida no Supabase. Saia e entre novamente no sistema.')
                    setSendingConfirmacao(false)
```

```
                      return
                    }
                    if (typeof res.body === 'object' && res.body !== null) {
                      const hint = (res.body as Record<string, unknown>).hint
                      if (typeof hint === 'string' && hint.trim()) {
                        setSendConfirmacaoResult(hint)
                        setSendingConfirmacao(false)
                        return
                      }
                      const code = (res.body as Record<string, unknown>).error
                      if (code === 'instance_not_connected') {
                        setSendConfirmacaoResult('WhatsApp não conectado. Vá em Configurações > WhatsApp e conecte a instância (QR Code).')
                        setSendingConfirmacao(false)
                        return
                      }
                    }
                    const details = typeof res.body === 'string' ? res.body : JSON.stringify(res.body)
                    setSendConfirmacaoResult(`Falha ao enviar confirmação (HTTP ${res.status}): ${details}`)
                    setSendingConfirmacao(false)
                    return
                  }

                  const details = typeof res.body === 'string' ? res.body : JSON.stringify(res.body)
                  setSendConfirmacaoResult(details || 'Enviado.')
                  setSendingConfirmacao(false)
                }}
                disabled={
                  !canTestConfirmacao ||
                  saving ||
                  loading ||
                  sendingConfirmacao ||
                  !agendamentoSelecionadoId ||
                  (selectedAgendamento?.status ?? '').trim().toLowerCase() !== 'confirmado'
                }
                className="w-full sm:w-auto"
              >
                {sendingConfirmacao ? 'Enviando…' : 'Enviar confirmação agora'}
              </Button>
            </div>

            {sendConfirmacaoResult ? (
              <pre className="rounded-xl border border-slate-200 bg-slate-50 p-3 text-xs text-slate-800 overflow-auto whitespace-pre-wrap">{sendConfirmacaoResult}</pre>
            ) : null}
                </div>
              </Card>
            </div>

            <div
              className={
                tutorialOpen && tutorialSteps[tutorialStep]?.target === 'templates'
                  ? 'ring-2 ring-slate-900 ring-offset-2 ring-offset-slate-50 rounded-xl'
                  : ''
              }
            >
              <div className="space-y-6">
                <Card>
                  <div className="space-y-4 p-4 sm:p-6">
                    <div className="text-sm font-semibold text-slate-900">Modelos prontos</div>
                    <div className="text-sm text-slate-600">
                      Escolha um modelo e ajuste se quiser. As variáveis são preenchidas automaticamente com dados reais do agendamento.
                    </div>

                    <div className="grid grid-cols-1 gap-3 sm:grid-cols-2">
                      <label className="block">
                        <div className="text-sm font-medium text-slate-700 mb-1">Modelo</div>
                        <select
                          className="w-full rounded-lg border border-slate-200 bg-white px-3 py-2 text-sm text-slate-900 outline-none focus:ring-2 focus:ring-slate-300"
                          value={presetKey}
```

```
                          onChange={(e) => {
                            const nextKey = e.target.value
                            setPresetKey(nextKey)
                            applyPreset('all', nextKey)
                          }}
                          disabled={!canEditTemplates || saving || loading}
                        >
                          {templatePresets.map((p) => (
                            <option key={p.key} value={p.key}>
                              {p.title}
                            </option>
                          ))}
                        </select>
                      </label>

                      <div className="grid grid-cols-2 gap-2 sm:self-end md:grid-cols-4">
                        <Button
                          variant="secondary"
                          onClick={() => applyPreset('confirmacao')}
                          disabled={!canEditTemplates || saving || loading}
                          className="w-full h-10"
                        >
                          Aplicar confirmação
                        </Button>
                        <Button
                          variant="secondary"
                          onClick={() => applyPreset('lembrete')}
                          disabled={!canEditTemplates || saving || loading}
                          className="w-full h-10"
                        >
                          Aplicar lembrete
                        </Button>
                        <Button
                          variant="secondary"
                          onClick={() => applyPreset('cancelamento')}
                          disabled={!canEditTemplates || saving || loading}
                          className="w-full h-10"
                        >
                          Aplicar cancelamento
                        </Button>
                        <Button
                          variant="secondary"
                          onClick={() => applyPreset('all')}
                          disabled={!canEditTemplates || saving || loading}
                          className="w-full h-10"
                        >
                          Aplicar tudo
                        </Button>
                      </div>
                    </div>
                  </div>
                </Card>

                <Card>
                  <div className="space-y-4 p-4 sm:p-6">
                    <div className="text-sm font-semibold text-slate-900">Variáveis</div>
                    <div className="text-sm text-slate-600">Clique para inserir no campo selecionado (confirmação/lembrete/cancelamento).</div>

                    <div className="grid grid-cols-2 gap-2 sm:flex sm:flex-wrap">
                      {templateVars.map((v) => (
                        <Button
                          key={v.key}
                          variant="secondary"
                          onClick={() => insertVar(v.key)}
                          disabled={!canEditTemplates || saving || loading}
                          className="w-full sm:w-auto"
                        >
                          {v.label}
                        </Button>
                      ))}
                    </div>
```

```
                </div>
              </Card>

              <Card>
                <div className="space-y-4 p-4 sm:p-6">
                  <div className="text-sm font-semibold text-slate-900">Mensagem de confirmação</div>
                  {loading ? (
                    <div className="text-sm text-slate-600">Carregando…</div>
                  ) : (
                    <textarea
                      className="w-full min-h-[160px] rounded-lg border border-slate-200 bg-white px-3 py-2 text-base
text-slate-900 outline-none focus:ring-2 focus:ring-slate-300 sm:text-sm"
                      value={mensagemConfirmacao}
                      onChange={(e) => setMensagemConfirmacao(e.target.value)}
                      disabled={!canEditTemplates || saving}
                      ref={confirmacaoRef}
                      onFocus={() => setLastField('confirmacao')}
                    />
                  )}
                </div>
              </Card>

              <Card>
                <div className="space-y-4 p-4 sm:p-6">
                  <div className="text-sm font-semibold text-slate-900">Mensagem de lembrete</div>
                  {loading ? (
                    <div className="text-sm text-slate-600">Carregando…</div>
                  ) : (
                    <textarea
                      className="w-full min-h-[160px] rounded-lg border border-slate-200 bg-white px-3 py-2 text-base
text-slate-900 outline-none focus:ring-2 focus:ring-slate-300 sm:text-sm"
                      value={mensagemLembrete}
                      onChange={(e) => setMensagemLembrete(e.target.value)}
                      disabled={!canEditTemplates || saving}
                      ref={lembreteRef}
                      onFocus={() => setLastField('lembrete')}
                    />
                  )}
                </div>
              </Card>

              <Card>
                <div className="space-y-4 p-4 sm:p-6">
                  <div className="text-sm font-semibold text-slate-900">Mensagem de cancelamento</div>
                  {loading ? (
                    <div className="text-sm text-slate-600">Carregando…</div>
                  ) : (
                    <textarea
                      className="w-full min-h-[160px] rounded-lg border border-slate-200 bg-white px-3 py-2 text-base
text-slate-900 outline-none focus:ring-2 focus:ring-slate-300 sm:text-sm"
                      value={mensagemCancelamento}
                      onChange={(e) => setMensagemCancelamento(e.target.value)}
                      disabled={!canEditTemplates || saving}
                      ref={cancelamentoRef}
                      onFocus={() => setLastField('cancelamento')}
                    />
                  )}
                </div>
              </Card>

              {canEditTemplates ? (
                <div className="flex justify-end">
                  <Button onClick={save} disabled={saving || loading} className="w-full sm:w-auto">
                    Salvar
                  </Button>
                </div>
              ) : (
                <div className="text-sm text-slate-600">Edição disponível apenas para a conta master.</div>
              )}
            </div>
          </div>
        </div>
```

```
          <TutorialOverlay
            open={tutorialOpen}
            steps={tutorialSteps}
            step={tutorialStep}
            onStepChange={setTutorialStep}
            onClose={closeTutorial}
            titleFallback="Mensagens"
          />
        </AppShell>
      )}
    </PageTutorial>
  )
}
```