Plano inicial — E-commerce de Pijamas & Lingerie (Supabase)

Objetivo: lançar uma loja virtual moderna, rápida e segura para venda de pijamas, lingeries e peças íntimas, com área administrativa separada e banco de dados no Supabase.

1) Resumo executivo

- **Público-alvo:** mulheres 18–45, compradores de presentes, clientes que valorizam conforto e estilo. Mobile-first.
- **Proposta de valor:** variedade (tamanhos/cores), fotos reais de tecido/caimento, entrega rápida, checkout simples (Pix, cartão, boleto), política de troca descomplicada.
- **Stack sugerida:** Next.js 14 (App Router) + TypeScript + Tailwind + shadcn/ui + Supabase (Auth, Postgres, Storage, RLS) + Mercado Pago (ou Stripe) + Melhor Envio/Correios + Resend (e-mail) + Vercel.
- **Diferenciais pro e-commerce de moda íntima:** guia de medidas claro (PP–GG), recomendador de tamanho (questionário rápido), tabela de "transparência do tecido", kits combináveis (sutiã + calcinha), "compre o look", fotos com corpos diversos.

2) Assunções iniciais (ajustáveis)

1. Brasil (BRL, pt-BR, fuso America/Sao_Paulo).
2. Pagamentos: **Mercado Pago** (Pix/Cartão/Boleto). Parcelamento até 6x (configurável).
3. Entregas via **Melhor Envio** integrando **Correios + Jadlog**; prazo de postagem configurado em **1–2 dias úteis**; frete grátis acima de **R$199–R$249** (ajustável); **retirada em loja** opcional.
4. SEO forte e páginas de produto indexáveis (ISR + cache CDN).
5. Admin com controle de permissões (admin/gestor/atendente).

3) Personas rápidas

- **Laura (28)**: compra para si; valoriza conforto e entrega rápida; paga por Pix.
- **Rafa (32)**: compra presente; precisa de guias visuais para tamanhos e kits.
- **Ana (41)**: fidelidade; procura combos e programa de pontos.

4) Requisitos funcionais

Catálogo & busca

1. Listagem de produtos com **filtros** (categoria, tamanho, cor, faixa de preço, coleção, tecido).
2. **Ordenação** (mais vendidos, menor/maior preço, novidades).
3. **Busca** com sugestões instantâneas (produtos, categorias, conteúdo).
4. Páginas de **categoria/coleção** e **landing pages** sazonais (Dia dos Namorados, Inverno, etc.).
5. Página de **produto** com variações (tamanho/cor), fotos múltiplas, tabela de medidas, composição do tecido, recomendações.

Carrinho & checkout

6. Carrinho com **quantidade**, **cupom**, cálculo de frete, estimativa de entrega.
7. Checkout em **3 passos** (dados, frete, pagamento) com **autosalvamento**.
8. Pagamento: Pix (QR + copia e cola), Cartão (tokenização), Boleto (opcional).
9. Emissão e validação de **cupons** (valor fixo, %, frete grátis, “leve 3 pague 2”).
10. **Pedidos** com status (criado, aguardando pagamento, pago, separado, enviado, entregue, cancelado, reembolsado).

Conta do usuário

11. Cadastro/login via **Supabase Auth** (e-mail + senha; opcional: login mágico por e-mail).
12. **Endereços**, **métodos de pagamento salvos** (via gateway), **favoritos**, histórico, rastreio.
13. **Troca/devolução** com abertura de solicitação e etiqueta reversa (manual ou via integração futura).

Conteúdo & relacionamento

14. Avaliação de produtos (1–5) com texto e fotos (moderadas).
15. **Wishlist**/Favoritos e **notificação de volta ao estoque**.
16. Newsletter (coleta de e-mail, integração com GA4/Pixel).
17. Páginas institucionais: Tabela de Medidas, Trocas e Devoluções, Política de Privacidade (LGPD), Contato/WhatsApp.

Administração (área separada)

18. Autenticação com RBAC (roles) e **múltiplos perfis** (admin/gestor/atendente).
19. **CRUD** de produtos, variações, categorias, coleções, banners, cupons, kits.
20. **Estoque** por variação (reserva em checkout, baixa ao pagamento confirmado).
21. Gestão de **pedidos** (filtro por status, confirmação, cancelamento, reembolso, troca).
22. **Promoções** agendadas (período de ativação) e regras (ex.: desconto por categoria).
23. **Relatórios**: vendas por período/categoria, conversão, ticket médio, abandono de carrinho.
24. **Importação CSV** de produtos/estoque e **exportação** de pedidos.

5) Requisitos não funcionais

- **Performance:** LCP < 2,5s, TTFB < 200ms em páginas críticas; imagens otimizadas (next/image).
- **SEO:** schema.org (Product, Offer, Breadcrumb), sitemap, canonical, meta OG.
- **Segurança:** OWASP Top 10; RLS no Supabase; webhooks verificados; rate limit.
- **Acessibilidade:** WCAG 2.2 AA; navegação por teclado; contraste ≥ 4.5:1; foco visível.
- **Escalabilidade:** cache CDN, ISR, DB com índices e partições em tabelas de eventos.
- **Observabilidade:** logs centralizados, métricas, tracing, alertas.

6) Casos de uso (detalhados)

Formato: **Fluxo principal (FP)**, **Alternativos (FA)**, **Exceções (FE)**.

UC-01 — Navegar catálogo

- FP: Usuário acessa /categoria → aplica filtros → ordena → vê produtos paginados.
- FA: Salva filtros como link compartilhável (UTM).
- FE: Sem resultados → exibir “ajuste filtros” + produtos populares.

UC-02 — Ver produto e variações

- FP: Abre /p/{slug} → escolhe cor/tamanho → visualiza preço promo, estoque, tabela medidas.
- FA: Visualiza **kit** (produto + peça combinável) com desconto progressivo.
- FE: Variação sem estoque → botão “avise-me” (notificação e-mail/WhatsApp).

UC-03 — Adicionar ao carrinho

- FP: Seleciona variação/tamanho → adiciona → mini-carrinho confirma e sugere acessórios.
- FA: Adição rápida da listagem (card com seleção de tamanho inline).
- FE: Estoque insuficiente no momento da adição → mensagem clara + quantidade máxima.

UC-04 — Cadastro/Login

- FP: Informa e-mail/senha → Supabase Auth cria perfil → redireciona para conta/checkout.
- FA: Login mágico por e-mail; social login (futuro).
- FE: E-mail já existente → orientar recuperar senha.

UC-05 — Checkout & Pagamento

- FP: Endereço → frete → pagamento (Pix/Cartão) → confirmação → pedido “aguardando pagamento/ pago”.
- FA: **Cupom** aplicado; **cartão salvo** via gateway; **Pix** com contagem regressiva.
- FE: Pagamento recusado/expirado → reter carrinho e oferecer método alternativo.

UC-06 — Acompanhar pedido

- FP: Usuário acessa Meus Pedidos → visualiza status e código de rastreio.
- FA: Recebe e-mails a cada mudança de status.
- FE: Rastreio indisponível → instruções e contato.

UC-07 — Avaliação de produto

- FP: Após entrega, e-mail convida a avaliar (1–5) + comentário + fotos.
- FA: Moderação automática por blacklist + aprovação manual no Admin.
- FE: Conteúdo impróprio → bloqueio e notificação ao usuário.

UC-08 — Admin: Produtos

- FP: Admin cria produto (nome, descrição, tags, categorias) → adiciona variações (tamanho/cor/SKU/GTIN), preço base/promo, estoque, imagens.
- FA: Clonar produto/variação; importação via CSV; upload em lote.
- FE: Validação falha (SKU duplicado) → feedback de campo + prevenção de gravação parcial.

UC-09 — Admin: Pedidos

- FP: Lista pedidos → filtra por status → abre detalhe → altera status → imprime etiqueta.
- FA: Geração de nota e etiqueta via integração (futuro) → anexa rastreio.
- FE: Tentativa de reembolso sem permissão → negar e logar evento (auditoria).

UC-10 — Admin: Promoções/Cupons

- FP: Cadastra cupom (código, desconto, limite, validade, escopo) → ativa/desativa.
- FA: Promo agendada por período; regras combinadas (ex.: categoria “pijamas” + mínimo R$150).
- FE: Cupom expirado → mensagem clara ao cliente.

7) Modelagem de dados (Supabase / Postgres)

Prefixo `public.`; uso de **RLS** em todas as tabelas com dados de usuário.

profiles (1–1 com `auth.users`)

- id (uuid, pk, = auth.uid) · role (enum: customer, admin, staff) · name · phone · created_at

addresses

- id · user_id (fk profiles) · label · recipient · phone · zip · street · number · complement · district · city · state · is_default

categories

- id · name · slug · parent_id (nullable) · position

products

- id · name · slug · description_md · care_instructions_md · category_id · brand · material · gender (fem/masc/unisex) · is_active · created_at

product_images

- id · product_id · url · alt · position · is_primary

variants

- id · product_id · sku · gtin (nullable) · color · size · price · compare_at_price (promo) · stock · weight_g · dimensions_json · is_active

collections

- id · name · slug · description · is_active

collection_products

- collection_id · product_id (pk composto)

carts

- id · user_id (nullable p/ convidado via cookie) · session_id · created_at · updated_at · coupon_code (nullable)

cart_items

- id · cart_id · variant_id · quantity · unit_price_snapshot

orders

- id · user_id · cart_id · status (enum) · total_items · total_discount · shipping_cost · total_paid · payment_status (enum) · payment_method (pix/card/boleto) · shipping_address_id · tracking_code · created_at

order_items

- id · order_id · variant_id · quantity · unit_price · name_snapshot · color_snapshot · size_snapshot

coupons

- id · code · type (percent/fixed/free_shipping/buyxgety) · value · min_subtotal · max_uses · used_count · starts_at · ends_at · scope (all/category/product)

inventory_movements

- id · variant_id · type (reserve/confirm/cancel/adjust) · quantity · order_id (nullable) · reason · created_at

reviews

- id · product_id · user_id · rating · comment · photos_json · is_approved · created_at

banners

- id · title · image_url · link_url · starts_at · ends_at · is_active · position

audit_logs

- id · actor_user_id · action · entity · entity_id · diff_json · created_at

newsletter_subscriptions

- id · email · consent (bool) · created_at

Índices: texto para `slug`, `sku`, `code`; `GIN` para busca full-text (`products.name`, `description`).

Políticas RLS (exemplos)

```
-- profiles: cada usuário só lê/atualiza o próprio perfil
create policy "own_profile_read" on profiles for select using
(auth.uid() = id);
create policy "own_profile_update" on profiles for update using
(auth.uid() = id);

-- orders: dono pode ler; escrita via backend/service role
create policy "own_orders_read" on orders for select using
(auth.uid() = user_id);

-- reviews: usuário autenticado cria; leitura pública de aprovadas
create policy "create_review" on reviews for insert with check
(auth.role() = 'authenticated');
```

```
create policy "read_approved_reviews" on reviews for select using
(is_approved = true);
```

(Admin opera com **Service Key** ou função `admin` via Supabase Edge Functions/Server Routes.)

Triggers úteis

- Ao **confirmar pagamento** → baixa estoque (insert em `inventory_movements`) e atualiza `orders.status`.
- Ao **cancelar pedido** → repõe estoque.

8) Arquitetura de software

- **Frontend**: Next.js 14 + App Router, **TypeScript**, **Tailwind**, **shadcn/ui**.
  - Estado: **TanStack Query** (server cache) + Zustand (ui local).
  - Formulários: **React Hook Form** + **Zod**.
  - Imagens: `next/image` com CDN e blur placeholders.
- **BFF/API**: Next.js **route handlers** como camada fina de backend (validação, orquestração, webhooks de pagamento, cálculo de frete, regras de negócio leves).
- **Dados**: **Supabase** (Postgres, Storage, Auth, RLS). Acesso via Supabase JS no server e client (somente leituras públicas seguras).
- **Pagamentos**: **Mercado Pago** (PIX/Cartão/Boleto). Webhook → API `/api/payments/webhook` → atualiza pedido/estoque.
- **Frete**: **Melhor Envio** (cálculo e etiqueta).
- **E-mail**: **Resend** para transacionais (pedido criado, pago, enviado, entregue) com templates.
- **Deploy**: **Vercel** (preview por PR). Supabase gerenciado.

Diagrama (alto nível)

Usuário ⇄ Next.js (SSR/ISR + API Routes) ⇄ Supabase (Auth/Postgres/Storage) ↘ Gateways (Mercado Pago, Melhor Envio, Resend)

9) UI/UX — diretrizes e componentes

- **Design system**: cores neutras com acentos suaves; tipografia legível; ícones lucide.
- **Componentes**: Header sticky com **busca**; mega-menu; cards com swatches de cor; **seletor de tamanho** acessível; mini-carrinho; passo-a-passo de checkout; toasts claros.
- **Guia de medidas** com fotos reais e equivalência (PP/P/M/G/GG) + centímetros.
- **Provas sociais**: avaliações, UGC (galeria com moderação), contagem de compras (discreta).
- **Redução de atrito**: login só no fim do checkout; salvar carrinho no dispositivo; preenchimento automático de CEP.
- **Microcopy**: mensagens humanizadas (ex.: “Este tecido é macio e não pinica”); transparência sobre forro/espessura.
- **A/B tests**: botões “Comprar agora” vs “Adicionar ao carrinho”; posição da tabela de medidas.

10) Acessibilidade (WCAG 2.2 AA)

- Fluxo 100% navegável por **teclado**; foco sempre visível.
- **Leitores de tela**: usar semântica correta (nav, main, header, footer), `aria-live` em toasts.
- **Imagens**: `alt` descritivo com material/cor; variações com `aria-selected`.
- Alvos de toque ≥ **44px**; erros de formulário descrevem **o problema e a solução**.
- Contraste ≥ 4.5:1; preferir **texto real** à imagem de texto.

11) Segurança (OWASP Top 10 aplicado)

- **RLS** em todas as tabelas com dados sensíveis + **least privilege**.
- Validação **Zod** em toda entrada; sanitização de HTML (se houver rich text).
- **CSP** rígido; **HSTS**, `SameSite=Lax`, `HttpOnly`, `Secure` cookies.
- **Rate limiting** por IP (ex.: Upstash Redis) em login/checkout/webhooks.
- **Verificação de webhooks** (assinatura + idempotência).
- **Proteção CSRF** nos POSTs autênticos (tokens) onde necessário.
- **Logs de auditoria** no Admin (quem fez o quê e quando).

12) Performance & SEO

- **ISR** para PDPs/PLPs; cache de 5–15 min com revalidação sob demanda no Admin.
- **Critical CSS** com Tailwind JIT; lazy de imagens (LQIP) e componentes pesados.
- **Prefetch inteligente** de rotas próximas; evitar hydration desnecessária.
- **Schema.org**: `Product`, `Offer`, `AggregateRating`, `BreadcrumbList`.
- **Sitemap** e `robots.txt`; **canonical** e noindex para páginas administrativas.

13) Integrações

- **Pagamentos**: Mercado Pago (PIX, Cartão, Boleto). Alternativa: Stripe (PIX via parceiros, Cartão), PagSeguro.
- **Frete**: Melhor Envio + Correios/Jadlog. Cálculo no checkout e geração de etiquetas no Admin.
- **E-mail**: Resend/SendGrid com domínios autenticados (SPF/DKIM/DMARC).
- **Analytics**: GA4 + Meta Pixel com **Consent Mode**.

14) Observabilidade, CI/CD & Qualidade

- **CI** (GitHub Actions): lint (ESLint), typecheck, testes (Jest/Testing Library), E2E (Playwright), build.
- **CD**: Vercel (preview por PR; produção ao merge).
- **Métricas**: Web Vitals (Vercel Analytics), logs (pino/console + Logflare), erros (Sentry).
- **Backup**: políticas do Supabase + dump diário das tabelas críticas.

15) Roadmap de entrega (6–8 semanas sugeridas)

1. **Fundação (Semana 1)**: setup do repo, Next.js, Tailwind, shadcn, Supabase (projeto, auth, RLS base), design tokens.

2. **Catálogo (Sem. 2)**: schema de produtos/variações, listagens, PDP, busca & filtros.
3. **Carrinho/Checkout (Sem. 3)**: carrinho persistente, cálculo de frete, cupons.
4. **Pagamentos (Sem. 4)**: integração Mercado Pago (Pix/Cartão), webhooks, status de pedidos, baixa de estoque.
5. **Admin (Sem. 5)**: CRUD produtos/categorias/estoque, promoções, pedidos, banners.
6. **Qualidade (Sem. 6)**: testes, acessibilidade, SEO, performance; conteúdo e fotos.
7. **Go-Live (Sem. 7)**: migração/seed, domínio, e-mail transacional, política de trocas.
8. **Pós-lançamento (Sem. 8)**: A/B tests, kits/compre-o-look, newsletter, melhorias.

16) Critérios de aceite (amostra)

- Checkout conclui **Pix e Cartão** ponta-a-ponta em sandbox.
- Core Web Vitals "Good" nas páginas Home, PLP, PDP e Checkout em mobile 4G.
- WCAG 2.2 AA mínimo nas rotas públicas.
- Admin com RBAC funcional e auditoria.
- Estoque sincroniza com pedidos (baixa/estorno) e logs de inventário completos.

17) Prompt final estruturado (para outra IA desenvolver)

Resumo do projeto

Construa um e-commerce de pijamas e lingerie com Next.js 14 (App Router), TypeScript, Tailwind, shadcn/ui, Supabase (Auth/Postgres/Storage, RLS), pagamentos (Mercado Pago), frete (Melhor Envio/Correios), e-mail (Resend). Foco em SEO, performance, acessibilidade (WCAG AA) e área Admin separada com RBAC.

Requisitos

- Funcionais: catálogo com filtros, PDP com variações, carrinho/checkout 3 passos, cupons, Pix/Cartão, conta do usuário (endereços, pedidos, favoritos),

avaliações, wishlist, newsletter, páginas institucionais, Admin com CRUD de produtos/variações/categorias/coleções/banners/cupons, gestão de estoque e pedidos, relatórios, import/export CSV.
- Não funcionais: SEO/schema.org, Core Web Vitals, RLS, logs auditoria, rate limit, observabilidade, CI/CD.

Casos de uso

- UC-01 a UC-10 conforme detalhado acima (navegação, PDP, adicionar ao carrinho, cadastro/login, checkout/pagamento, acompanhar pedido, avaliação, Admin: produtos/pedidos/promoções).

Arquitetura recomendada

- Next.js App Router + API Routes (webhooks e BFF).
- Supabase como origem de dados com RLS.
- Mercado Pago para pagamentos; Melhor Envio para cálculo de frete; Resend para e-mails.
- Deploy na Vercel; logs/erros via Sentry; GA4 + Pixel com Consent Mode.

Banco de dados (DDL resumido)

- Crie tabelas: profiles, addresses, categories, products, product_images, variants, collections, collection_products, carts, cart_items, orders, order_items, coupons, inventory_movements, reviews, banners, audit_logs, newsletter_subscriptions.
- Aplique índices para `slug`, `sku`, `code` e full-text search em `products`.
- Defina enums para `orders.status` e `orders.payment_status`.
- Adicione triggers para baixa/estorno de estoque ao mudar status do pedido.
- Aplique **RLS** conforme exemplos.

Páginas e rotas

- Público: `/`, `/categoria/[slug]`, `/colecao/[slug]`, `/p/[slug]`, `/busca`, `/sacola`, `/checkout`, `/conta` (login/register), `/meus-pedidos`, `/favoritos`, `/tabela-de-medidas`, `/trocas-e-devolucoes`, `/privacidade`, `/contato`.
- Admin (prefixo `/admin`): dashboard, produtos, variações, categorias, coleções, banners, cupons, pedidos, estoque, relatórios, usuários (somente leitura), configurações.

Componentes chave

- Card de produto com swatches; seletor de tamanho acessível; mini-carrinho; componente de frete por CEP; checkout em passos; tabela de medidas; avaliação com upload moderado.

Integrações

- Mercado Pago: criar preferências, capturar pagamentos, webhooks com verificação de assinatura e idempotência; mapear status → `orders.payment_status` e `orders.status`.
- Melhor Envio: cálculo de frete por CEP, serviços Correios/Jadlog; salvar cotações no pedido.
- Resend: templates de e-mail (pedido criado/pago/enviado/entregue); domínio autenticado (SPF/DKIM/DMARC).

Qualidade e segurança

- Testes unitários (Jest), E2E (Playwright).
- Acessibilidade com `@axe-core/playwright`.
- CSP, HSTS, cookies seguros, rate limit, validação Zod, sanitização de rich text.

Etapas numeradas

1. Setup e infraestrutura; 2) DB & RLS; 3) Catálogo/Busca; 4) PDP & variações; 5) Carrinho/Checkout; 6) Integração de pagamentos & webhooks; 7) Admin CRUD & estoque; 8) Relatórios & cupons; 9) Acessibilidade/SEO/performance; 10) Testes/E2E; 11) Conteúdo/mídia; 12) Go-live.

Observações importantes

- Fotos com **diversidade de corpos** e close do tecido;
- Tabela de medidas sempre a 1 clique;
- Política de trocas clara (7–30 dias, conforme estratégia);
- LGPD: consentimento de cookies/marketing e opt-out em e-mails.

18) Ideias extras de valor

- **“Compre o look”** (kit pijama + robe com desconto).
- **Assinatura** (ex.: cuecas/lingeries trimestrais com desconto).
- **Provador de tamanho** (perguntas rápidas: altura, peso, preferências de caimento).

- **UGC** com hashtag e moderação; vitrines editoriais (lookbook).
- **Live shopping** (futuro) e cupons timeboxed durante a live.

19) Próximos passos para começar agora

1. Confirmar **gateway de pagamento** (Mercado Pago por padrão).
2. Confirmar **frete** (Melhor Envio).
3. Validar **política de trocas/devoluções** e prazos.
4. Fornecer **identidade visual** (paleta, logotipo, tom de voz).
5. Juntar **catálogo inicial** (planilha CSV: produto, variações, preço, estoque, imagens).
6. Criar contas de **Resend/GA4/Pixel** e chaves de API.

20) Parâmetros confirmados — pacote “melhores escolhas”

Pagamentos

- Mercado Pago (Pix/Cartão/Boleto). Manter Pix como destaque no mobile; parcelamento até 6x (configurável).

Frete

- Melhor Envio com Correios + Jadlog; prazo de postagem 1–2 dias úteis; frete grátis acima de R$199–R$249 (configurável no Admin); Retirada em loja (opcional).

Política de trocas/devoluções (CDC)

- Arrependimento: até 7 dias após recebimento, frete por conta da loja.
- Troca por tamanho/cor: até 30 dias; frete de ida do cliente e volta da loja.
- Defeito: troca/devolução sem custo para o cliente.

Identidade visual

- Paleta: nude | rosé | marsala/berinjela | off-white.
- Tipografia: títulos Playfair Display ou Cinzel Display; textos Inter ou Montserrat.
- Referências: Intimissimi, Savage X Fenty, Hope, Liz.

Catálogo inicial

- 24–36 SKUs; 5–7 fotos por produto (frente, costas, close de tecido/renda, fecho, 2 poses, tabela de medidas).

Domínio e e-mails

- Registrar .com.br (e opcional .com).
- Caixas: contato@, suporte@, pedidos@, no-reply@ (transacionais) via Google Workspace.

MVP x Fase 2

- MVP: Wishlist + Newsletter com cupom de 1ª compra (10%).
- Fase 2: Avaliações com fotos + moderação e incentivos pós-compra.

21) Backlog inicial (épicos → histórias → tarefas)

Épico A — Fundação & Infra

Histórias:

- A1. Como visitante, quero acessar o site com carregamento rápido em mobile.

Critérios: LCP < 2,5s, TTFB < 200ms na Home.

Tarefas: setup Next.js 14, Tailwind, shadcn/ui; ESLint/Prettier; Vercel; variáveis de ambiente.

- A2. Como dev, quero Supabase configurado com Auth/RLS.

Critérios: login e registro funcionais; políticas RLS ativas nas tabelas base.

Tarefas: criar projeto, schemas, roles, policies base; seeds mínimos.

Épico B — Catálogo & Busca

Histórias:

- B1. Como cliente, quero filtrar por tamanho/cor/tecido/preço.

Critérios: filtros URL-driven; combinações persistem em reload.

Tarefas: PLP com TanStack Query; componentes de filtro; index/text search.

- B2. Como cliente, quero ver combos/coleções.

Critérios: páginas de coleção indexáveis; breadcrumbs e schema.org.

Épico C — PDP & Variações

Histórias:

- C1. Como cliente, quero escolher tamanho/cor e ver estoque e preço promocional.

Critérios: botão Comprar desabilita sem variação; mostra aviso “avise-me” quando sem estoque.

Tarefas: seletor acessível; swatches; tabela de medidas; UGC placeholder.

Épico D — Carrinho & Checkout

Histórias:

- D1. Como cliente, quero calcular frete por CEP e aplicar cupom.

Critérios: cotações via Melhor Envio; cupons percentuais e valor fixo; regra frete grátis.

Tarefas: serviço de frete; regras de cupom; mini-carrinho.

- D2. Como cliente, quero finalizar em 3 passos com autosave.

Critérios: recuperação de sessão em refresh; validação Zod; toasts acessíveis.

Épico E — Pagamentos (Mercado Pago)

Histórias:

- E1. Pix com QR/copia-e-cola e confirmação automática.

Critérios: webhook verificado e idempotente; status do pedido muda para “pago”.

Tarefas: criação de preferência, assinatura do webhook, mapping de status.

- E2. Cartão com parcelamento até 6x.

Critérios: transação autorizada e capturada; erros claros; retentativas.

Épico F — Logística (Melhor Envio)

Histórias:

- F1. Cálculo de frete Correios/Jadlog com prazo.

Critérios: exibe prazos; regra de frete grátis aplicável por subtotal.

Tarefas: integração SDK/API; cache de cotações.

- F2. Retirada em loja (se habilitado).

Critérios: opção visível para CEPs configurados; custo zero.

Épico G — Conta do Usuário

Histórias:

- G1. Endereços e pedidos.

Critérios: CRUD de endereços; timeline de status; rastreamento.

Épico H — Admin

Histórias:

- H1. CRUD de produtos/variações com estoque por variação.

Critérios: validação de SKU único; upload múltiplo; auditoria.

- H2. Pedidos (alterar status, rastreio) e cupons.

Critérios: filtros por status; exportação CSV.

Épico I — Conteúdo, SEO e UX

Histórias:

- I1. Páginas institucionais (trocas/devoluções, privacidade).
- I2. Newsletter e banner com UTM.

Épico J — Observabilidade & Qualidade

Histórias:

- J1. Sentry + Web Vitals; logs.
- J2. Testes E2E essenciais (checkout Pix e Cartão).

22) DDL inicial (Supabase) — Núcleo

```
-- Tipos
create type order_status as enum
('created','awaiting_payment','paid','separating','shipped','deliver
ed','canceled','refunded');
create type payment_method as enum ('pix','card','boleto');
create type payment_status as enum
('pending','approved','rejected','refunded','expired');

-- Perfis
create table public.profiles (
  id uuid primary key default auth.uid(),
  role text not null default 'customer' check (role in
('customer','admin','staff')),
  name text,
  phone text,
  created_at timestamptz not null default now()
);

-- Categorias
create table public.categories (
  id bigint generated by default as identity primary key,
  name text not null,
  slug text not null unique,
  parent_id bigint references public.categories(id) on delete set
null,
  position int not null default 0
);

-- Produtos
create table public.products (
  id bigint generated by default as identity primary key,
  name text not null,
  slug text not null unique,
  description_md text,
  care_instructions_md text,
```

```
  category_id bigint references public.categories(id) on delete set
null,
  brand text,
  material text,
  gender text check (gender in ('fem','masc','unisex')),
  is_active boolean not null default true,
  created_at timestamptz not null default now()
);

create index products_search_idx on public.products using gin
(to_tsvector('portuguese', coalesce(name,'') || ' ' ||
coalesce(description_md,'')));

-- Imagens de produto
create table public.product_images (
  id bigint generated by default as identity primary key,
  product_id bigint not null references public.products(id) on
delete cascade,
  url text not null,
  alt text,
  position int not null default 0,
  is_primary boolean not null default false
);

-- Variações
create table public.variants (
  id bigint generated by default as identity primary key,
  product_id bigint not null references public.products(id) on
delete cascade,
  sku text not null unique,
  gtin text,
  color text,
  size text,
  price numeric(12,2) not null,
  compare_at_price numeric(12,2),
  stock int not null default 0,
  weight_g int,
  dimensions_json jsonb,
  is_active boolean not null default true
);

-- Carrinhos
```

```
create table public.carts (
  id uuid primary key default gen_random_uuid(),
  user_id uuid references public.profiles(id) on delete set null,
  session_id text,
  coupon_code text,
  created_at timestamptz not null default now(),
  updated_at timestamptz not null default now()
);

-- Itens do carrinho
create table public.cart_items (
  id uuid primary key default gen_random_uuid(),
  cart_id uuid not null references public.carts(id) on delete
cascade,
  variant_id bigint not null references public.variants(id),
  quantity int not null check (quantity > 0),
  unit_price_snapshot numeric(12,2) not null
);

-- Pedidos
create table public.orders (
  id uuid primary key default gen_random_uuid(),
  user_id uuid references public.profiles(id) on delete set null,
  cart_id uuid references public.carts(id),
  status order_status not null default 'created',
  total_items numeric(12,2) not null default 0,
  total_discount numeric(12,2) not null default 0,
  shipping_cost numeric(12,2) not null default 0,
  total_paid numeric(12,2) not null default 0,
  payment_status payment_status not null default 'pending',
  payment_method payment_method,
  shipping_address_id uuid,
  tracking_code text,
  created_at timestamptz not null default now()
);

-- Itens do pedido
create table public.order_items (
  id uuid primary key default gen_random_uuid(),
  order_id uuid not null references public.orders(id) on delete
cascade,
  variant_id bigint not null references public.variants(id),
```

```
  quantity int not null check (quantity > 0),
  unit_price numeric(12,2) not null,
  name_snapshot text,
  color_snapshot text,
  size_snapshot text
);

-- Cupons
create table public.coupons (
  id uuid primary key default gen_random_uuid(),
  code text not null unique,
  type text not null check (type in
('percent','fixed','free_shipping','buyxgety')),
  value numeric(12,2),
  min_subtotal numeric(12,2),
  max_uses int,
  used_count int not null default 0,
  starts_at timestamptz,
  ends_at timestamptz,
  scope text check (scope in ('all','category','product'))
);

-- Índices úteis
create index cart_items_cart_idx on public.cart_items(cart_id);
create index order_items_order_idx on public.order_items(order_id);
```

Observação: habilite RLS nas tabelas com dados de usuário e aplique as políticas já definidas na seção 7.

23) Estrutura de pastas (Next.js — App Router)

```
/app
  /(public)
    /page.tsx
    /categoria/[slug]/page.tsx
    /colecao/[slug]/page.tsx
    /p/[slug]/page.tsx
    /busca/page.tsx
    /sacola/page.tsx
```

```
    /checkout/page.tsx
    /conta/(auth)/login/page.tsx
    /conta/(auth)/register/page.tsx
    /meus-pedidos/page.tsx
    /favoritos/page.tsx
    /tabela-de-medidas/page.tsx
    /trocas-e-devolucoes/page.tsx
    /privacidade/page.tsx
    /contato/page.tsx
  /admin
    /page.tsx
    /produtos/page.tsx
    /produtos/[id]/page.tsx
    /pedidos/page.tsx
    /cupons/page.tsx
    /banners/page.tsx
    /relatorios/page.tsx
  /api
    /payments/webhook/route.ts
    /frete/cotacao/route.ts
    /cupom/validar/route.ts
/components
/lib
/styles
```

24) Roteiros de teste E2E (Playwright) — MVP

1. **Checkout Pix (feliz)**

Dado um carrinho com 2 itens → quando informo endereço válido e CEP → e escolho frete econômico → e seleciono Pix → então vejo QR e status do pedido muda para “aguardando pagamento”; ao simular webhook “approved” → status vai para “pago” e estoque baixa.

2. **Checkout Cartão (autorizado)**

Dado um carrinho → quando preencho cartão válido (sandbox) e parcelamento 3x → então recebo confirmação e-mail “pedido pago”.

3. **Cupom e frete grátis**

Dado subtotal ≥ R$199 → quando aplico cupom 10% → então frete grátis permanece; total calculado corretamente.

4. **RLS e privacidade**

Um usuário logado não consegue acessar pedido de outro usuário (403); Admin consegue via service role.

5. **Acessibilidade**

Todas as páginas públicas sem erros críticos do `@axe-core/playwright`; navegação por teclado cobre checkout.