

DELIVEREI v1 - Resumo Técnico Completo

Projeto: Sistema Multi-tenant de Delivery

Versão: 1.0

Data de Compilação: 16 de outubro de 2025

Repositório: nerdrico2025/deliverei-v1

Status:  MVP Completo e Funcional

Índice

1. Visão Geral do Sistema
2. Stack Tecnológico
3. Funcionalidades Implementadas
4. Funcionalidades Planejadas (Não Implementadas)
5. Estrutura de Arquivos e Hierarquia
6. Modelos de Dados (Prisma Schema)
7. Endpoints da API
8. Integrações Externas
9. Segurança e Multi-tenancy
10. Otimizações e Performance

1. VISÃO GERAL DO SISTEMA

1.1 Descrição do Projeto

DELIVEREI é uma plataforma SaaS completa de delivery multi-tenant que permite que múltiplos estabelecimentos comerciais (restaurantes, lanchonetes, pizzarias, etc.) operem suas lojas virtuais de forma independente e isolada dentro de uma única infraestrutura compartilhada.

1.2 Propósito e Objetivos

Objetivos Principais:

- Fornecer uma solução completa de delivery para pequenos e médios estabelecimentos
- Garantir isolamento completo de dados entre diferentes empresas (multi-tenancy seguro)
- Oferecer interface administrativa intuitiva para gestão de produtos, pedidos e clientes
- Proporcionar experiência de compra otimizada para clientes finais
- Suportar múltiplos métodos de pagamento (PIX, Cartão, Boleto)
- Facilitar comunicação com clientes via WhatsApp e notificações

Público-alvo:

- **Empresas:** Restaurantes, pizzarias, lanchonetes, cafeterias
- **Clientes Finais:** Consumidores que desejam fazer pedidos online
- **Administradores:** Gestores de estabelecimentos comerciais
- **Super Admins:** Equipe de gestão da plataforma DELIVEREI

1.3 Arquitetura Geral

Modelo de Arquitetura: Monorepo Full-stack

```

deliverer-v1/
├── backend/           # API REST em NestJS
│   ├── src/          # Código-fonte do backend
│   ├── prisma/       # Schema e migrations do banco
│   └── ...
├── src/              # Frontend em React
│   ├── components/   # Componentes React
│   ├── pages/        # Páginas da aplicação
│   ├── services/     # Serviços de API
│   └── ...
└── docs/             # Documentação
  
```

Padrão Arquitetural:

- **Backend:** Arquitetura modular baseada em NestJS com separação de concerns (Controllers, Services, DTOs, Guards, Middlewares)
- **Frontend:** Arquitetura baseada em componentes React com Context API para gerenciamento de estado
- **Banco de Dados:** PostgreSQL gerenciado pelo Supabase com ORM Prisma
- **Autenticação:** JWT stateless com refresh tokens
- **Multi-tenancy:** Isolamento por empresa com middleware de tenant resolution

2. STACK TECNOLÓGICO

2.1 Backend

Core Framework e Runtime

- **NestJS** 10.3.0 - Framework Node.js progressivo baseado em TypeScript
- **Node.js** 20+ - Runtime JavaScript
- **TypeScript** 5.3.3 - Superset tipado de JavaScript

Banco de Dados e ORM

- **PostgreSQL** - Banco de dados relacional (via Supabase)
- **Prisma** 5.8.0 - ORM moderno com type-safety
- **Supabase** - Plataforma de banco de dados gerenciado

Autenticação e Segurança

- **Passport** 0.7.0 - Middleware de autenticação
- **Passport-JWT** 4.0.1 - Estratégia JWT para Passport
- **Passport-Local** 1.0.0 - Estratégia local para Passport
- **bcrypt** 5.1.1 - Hash de senhas
- **@nestjs/jwt** 10.2.0 - Módulo JWT do NestJS

Validação e Transformação

- **class-validator** 0.14.1 - Decorators de validação
- **class-transformer** 0.5.1 - Transformação de objetos

Integrações Externas

- **Stripe** 19.1.0 - Gateway de pagamento (assinaturas)
- **Axios** 1.12.2 - Cliente HTTP
- **@nestjs/axios** 4.0.1 - Módulo Axios do NestJS

Cache e Performance

- **Redis** 4.6.12 - Cache in-memory e gerenciamento de sessões

Documentação

- **@nestjs/swagger** 7.4.2 - Documentação automática de API
- **swagger-ui-express** 5.0.1 - UI do Swagger

Utilitários

- **uuid** 9.0.1 - Geração de UUIDs
- **rimraf** 5.0.5 - Limpeza de diretórios
- **rxjs** 7.8.1 - Programação reativa

2.2 Frontend

Core Framework

- **React** 18.3.1 - Biblioteca para interfaces de usuário
- **React DOM** 18.3.1 - Renderização DOM para React
- **TypeScript** 5.5.3 - Superset tipado de JavaScript

Build Tool

- **Vite** 5.4.2 - Build tool moderno e rápido
- **@vitejs/plugin-react** 4.3.1 - Plugin React para Vite

Roteamento

- **React Router DOM** 7.9.3 - Roteamento declarativo

Estilização

- **Tailwind CSS** 3.4.1 - Framework CSS utility-first
- **PostCSS** 8.4.35 - Processador CSS
- **Autoprefixer** 10.4.18 - Plugin PostCSS para prefixos vendor

Comunicação com API

- **Axios** 1.12.2 - Cliente HTTP
- **@supabase/supabase-js** 2.57.4 - Cliente Supabase

Visualização de Dados

- **Recharts** 3.2.1 - Biblioteca de gráficos React

Gerenciamento de Pagamentos

- **@stripe/react-stripe-js** 5.2.0 - Componentes React para Stripe
- **@stripe/stripe-js** 8.0.0 - Cliente JavaScript Stripe

Utilitários

- **date-fns** 4.1.0 - Manipulação de datas
- **lucide-react** 0.344.0 - Ícones React
- **react-helmet-async** 2.0.5 - Gerenciamento de head da página
- **react-qr-code** 2.0.18 - Geração de QR codes

Qualidade de Código

- **ESLint** 9.9.1 - Linter JavaScript/TypeScript
- **typescript-eslint** 8.3.0 - Parser e plugin ESLint para TypeScript

Otimização

- **Terser** 5.44.0 - Minificador JavaScript

2.3 Infraestrutura

Hospedagem e Deploy

- **Render** - Plataforma de hospedagem cloud
- Backend API hospedado como Web Service
- PostgreSQL gerenciado via Render
- **Supabase** - Backend-as-a-Service
- PostgreSQL database hosting
- Connection pooling
- Autenticação (não utilizada atualmente)

Containerização (Desenvolvimento)

- **Docker** - Containerização de aplicações
- **Docker Compose** - Orquestração de containers (Redis)

Controle de Versão

- **Git** - Sistema de controle de versão
- **GitHub** - Hospedagem de repositório e CI/CD

3. FUNCIONALIDADES IMPLEMENTADAS

3.1 Autenticação e Autorização

Recursos Implementados

- ☒ Sistema de login com email e senha
- ☒ Cadastro de novos usuários (signup)
- ☒ Sistema de JWT com access token (15min) e refresh token (7 dias)
- ☒ Renovação automática de tokens (refresh)
- ☒ Logout com invalidação de refresh token
- ☒ Criação de conta a partir de pedido (checkout sem cadastro prévio)

Níveis de Acesso (Roles)

- **SUPER_ADMIN:** Acesso total ao sistema, gestão de empresas e assinaturas
- **ADMIN_EMPRESA:** Acesso à administração da própria empresa
- **CLIENTE:** Acesso à vitrine e funcionalidades de compra

Endpoints

POST /auth/login	- Login
POST /auth/signup	- Cadastro
POST /auth/refresh	- Renovar token
POST /auth/logout	- Logout
POST /auth/create-account-from-order	- Criar conta via checkout

3.2 Multi-tenancy (Empresas)

Recursos Implementados

- ☒ Isolamento completo de dados por empresa
- ☒ Tenant Middleware para extração de tenant via:
 - Subdomínio (ex: `pizza-express.deliverei.com.br`)
 - Slug na URL (ex: `/public/pizza-express`)
- ☒ Validação de tenant em todas as operações
- ☒ Decorator `@CurrentEmpresa()` para injetar empresa atual
- ☒ CRUD completo de empresas (SuperAdmin)

Modelo Empresa

```
{
  id: string
  nome: string
  slug: string (único)
  subdominio: string (único)
  ativo: boolean
  whatsappNumero: string?
  whatsappToken: string?
  asaasCustomerId: string?
}
```

Endpoints

GET	/empresas	- Listar empresas (SuperAdmin)
POST	/empresas	- Criar empresa (SuperAdmin)
GET	/empresas/:id	- Buscar empresa (SuperAdmin)
PATCH	/empresas/:id	- Atualizar empresa (SuperAdmin)
DELETE	/empresas/:id	- Remover empresa (SuperAdmin)

3.3 Gestão de Produtos

Recursos Implementados

- ☒ CRUD completo de produtos
- ☒ Listagem com paginação, busca e filtros
- ☒ Upload de imagens (via URL)
- ☒ Controle de estoque
- ☒ Categorização de produtos
- ☒ Soft delete (produtos inativos)
- ☒ Hard delete (SUPER_ADMIN apenas)
- ☒ Badges de status (lowStock, outOfStock)

Modelo Produto

```
{
  id: string
  nome: string
  descricao: string?
  preco: Decimal
  imagem: string?
  ativo: boolean
  empresaId: string
  estoque: int
  categoria: string?
}
```

Endpoints Administrativos










GET	/produtos	- Listar produtos da empresa
POST	/produtos	- Criar produto
GET	/produtos/:id	- Buscar produto
PATCH	/produtos/:id	- Atualizar produto
DELETE	/produtos/:id	- Soft delete
DELETE	/produtos/:id/hard	- Hard delete (SuperAdmin)

Endpoints Públicos

GET	/public/:slug/produtos	- Produtos da loja
GET	/public/:slug/produtos/:id	- Detalhes do produto
GET	/public/:slug/categorias	- Categorias disponíveis

3.4 Gestão de Pedidos

Recursos Implementados

-  Criação de pedidos com itens
-  Cálculo automático de totais (subtotal, desconto, frete, total)
-  Gestão de status do pedido
-  Histórico de mudanças de status
-  Listagem com filtros e paginação
-  Detalhes completos do pedido
-  Snapshot de dados do cliente e produtos
-  Snapshot de endereço de entrega
-  Integração com sistema de pagamento

Status de Pedido

- **PENDENTE:** Pedido criado, aguardando confirmação
- **CONFIRMADO:** Pedido confirmado pela empresa
- **EM_PREPARO:** Pedido em preparação
- **SAIU_ENTREGA:** Pedido saiu para entrega
- **ENTREGUE:** Pedido entregue ao cliente
- **CANCELADO:** Pedido cancelado

Modelo Pedido

```
{
  id: string
  numero: string (único)
  status: StatusPedido
  subtotal: Decimal
  desconto: Decimal
  total: Decimal
  frete: Decimal
  clienteId: string
  empresaId: string
  enderecoEntrega: string?
  formaPagamento: string?
  cupomDesconto: string?
  observacoes: string?
  itens: ItemPedido[]
}
```

Endpoints

GET	/pedidos	- Listar pedidos da empresa
POST	/pedidos	- Criar pedido
GET	/pedidos/:id	- Buscar pedido
PATCH	/pedidos/:id/status	- Atualizar status
DELETE	/pedidos/:id	- Cancelar pedido

3.5 Carrinho de Compras

Recursos Implementados (Backend)

- ☒ CRUD completo de carrinho
- ☒ Adicionar/remover itens
- ☒ Atualizar quantidade
- ☒ Calcular totais
- ☒ Limpar carrinho
- ☒ Validação de estoque
- ☒ Conversão de carrinho em pedido

Recursos Implementados (Frontend)

- ☒ Context API para gerenciamento de estado global
- ☒ Persistência em localStorage
- ☒ CartDrawer lateral responsivo
- ☒ Controle de quantidade (+/-)
- ☒ Upsell/Cross-sell integrado
- ☒ Cálculo automático de subtotal

Endpoints

```
GET    /carrinho          - Buscar carrinho do usuário
POST   /carrinho/itens     - Adicionar item
PATCH /carrinho/itens/:id - Atualizar quantidade
DELETE /carrinho/itens/:id - Remover item
DELETE /carrinho      - Limpar carrinho
POST   /carrinho/checkout - Converter em pedido
```

3.6 Cupons e Descontos

Recursos Implementados

- ☒ CRUD completo de cupons
- ☒ Tipos de desconto: PERCENTUAL, VALOR_FIXO
- ☒ Valor mínimo de compra
- ☒ Período de validade (data início/fim)
- ☒ Controle de uso (máximo de usos)
- ☒ Validação de cupom no checkout
- ☒ Aplicação automática de desconto
- ☒ Histórico de uso

Modelo Cupom

```
{
  id: string
  codigo: string
  descricao: string?
  tipo: 'PERCENTUAL' | 'VALOR_FIXO'
  valor: Decimal
  valorMinimo: Decimal?
  dataInicio: DateTime
  dataFim: DateTime
  ativo: boolean
  usoMaximo: int?
  usoAtual: int
  empresaId: string
}
```





Endpoints

```
GET    /cupons          - Listar cupons da empresa
POST   /cupons        - Criar cupom
GET    /cupons/:id    - Buscar cupom
PATCH /cupons/:id    - Atualizar cupom
DELETE /cupons/:id    - Remover cupom
POST   /cupons/validar - Validar cupom
```

3.7 Pagamentos

Recursos Implementados

- ☒ Integração com Asaas (gateway brasileiro)
- ☒ Suporte a PIX, Cartão de Crédito e Boleto
- ☒ Geração de QR Code PIX
- ☒ Geração de link de boleto

-  Webhook para confirmação de pagamento
-  Controle de status de pagamento
-  Histórico de pagamentos
-  Pagamentos de pedidos e assinaturas

Status de Pagamento

- **PENDENTE:** Pagamento aguardando confirmação
- **APROVADO:** Pagamento confirmado
- **RECUSADO:** Pagamento recusado
- **CANCELADO:** Pagamento cancelado

Modelo Pagamento








```
{
  id: string
  empresaId: string
  assinaturaId: string?
  pedidoId: string?
  tipo: 'ASSINATURA' | 'PEDIDO'
  metodo: 'PIX' | 'CARTAO' | 'BOLETO'
  status: string
  valor: Float
  asaasPaymentId: string?
  asaasInvoiceUrl: string?
  pixQrCode: string?
  pixCopyPaste: string?
  boletoUrl: string?
  dataVencimento: DateTime?
  dataPagamento: DateTime?
}
```

Endpoints

POST	/pagamentos	- Criar pagamento
GET	/pagamentos/:id	- Buscar pagamento
GET	/pagamentos/pedido/:id	- Pagamentos do pedido
POST	/webhooks/asaas	- Webhook Asaas

3.8 Assinaturas (Planos de Empresa)

Recursos Implementados

-  Sistema de planos (BASICO, PROFISSIONAL, ENTERPRISE)
-  Integração com Stripe para pagamentos recorrentes
-  Controle de status (ATIVA, CANCELADA, SUSPENSA, TRIAL)
-  Gestão de ciclos de cobrança
-  Webhook Stripe para sincronização
-  Histórico de assinaturas
-  Dashboard de assinaturas (SuperAdmin)

Modelo Assinatura

```
{
  id: string
  empresaId: string
  plano: 'BASICO' | 'PROFISSIONAL' | 'ENTERPRISE'
  status: 'ATIVA' | 'CANCELADA' | 'SUSPensa' | 'TRIAL'
  stripeCustomerId: string?
  stripeSubscriptionId: string?
  dataInicio: DateTime
  dataFim: DateTime?
  valorMensal: Float
  proximaCobranca: DateTime?
}
```

Endpoints

GET	/assinaturas	- Listar assinaturas (SuperAdmin)
POST	/assinaturas	- Criar assinatura
GET	/assinaturas/:id	- Buscar assinatura
PATCH	/assinaturas/:id	- Atualizar assinatura
DELETE	/assinaturas/:id	- Cancelar assinatura
POST	/webhooks/stripe	- Webhook Stripe

3.9 Dashboard e Relatórios

Recursos Implementados

- ☒ Métricas gerais (vendas, pedidos, clientes)
- ☒ Gráfico de vendas por período
- ☒ Produtos mais vendidos
- ☒ Pedidos recentes
- ☒ Filtros por período (hoje, semana, mês, personalizado)
- ☒ Cálculos de crescimento
- ☒ Cards informativos com estatísticas

Métricas Disponíveis

```
{
  vendasHoje: number
  vendasMes: number
  pedidosHoje: number
  pedidosMes: number
  crescimentoVendas: number
  crescimentoPedidos: number
  produtosMaisVendidos: Array
  pedidosRecentes: Array
  graficoVendas: Array
}
```

Endpoints

GET	/dashboard/metricas	- Métricas gerais
GET	/dashboard/vendas	- Vendas por período
GET	/dashboard/produtos-top	- Produtos mais vendidos

3.10 Notificações

Recursos Implementados

- ☒ Sistema de notificações em tempo real
- ☒ Tipos: PEDIDO, SISTEMA, PROMOCAO
- ☒ Marcação de leitura
- ☒ Listagem com paginação
- ☒ Notificações no header (dropdown)
- ☒ Badge de contador
- ☒ Context API para gerenciamento

Modelo Notificacao

```
{
  id: string
  titulo: string
  mensagem: string
  tipo: 'PEDIDO' | 'SISTEMA' | 'PROMOCAO'
  lida: boolean
  usuarioId: string
  pedidoId: string?
}
```

Endpoints

GET	/notificacoes	- Listar notificações do usuário
POST	/notificacoes	- Criar notificação
PATCH	/notificacoes/:id/lida	- Marcar como lida
PATCH	/notificacoes/ler-todas	- Marcar todas como lidas
DELETE	/notificacoes/:id	- Remover notificação

3.11 Integração com WhatsApp

Recursos Implementados

- ☒ Envio de mensagens via WhatsApp Business API
- ☒ Notificações de pedido
- ☒ Configuração de número e token
- ☒ Histórico de mensagens
- ☒ Status de entrega (enviada, entregue, lida, erro)
- ☒ Template de mensagens

Modelo MensagemWhatsApp

```
{
  id: string
  empresaId: string
  pedidoId: string?
  telefone: string
  mensagem: string
  tipo: 'NOTIFICACAO' | 'CHAT'
  direcao: 'ENVIADA' | 'RECEBIDA'
  status: 'PENDENTE' | 'ENVIADA' | 'ENTREGUE' | 'LIDA' | 'ERRO'
  whatsappId: string?
  erro: string?
}
```

Endpoints

POST	/whatsapp/enviar	- Enviar mensagem
GET	/whatsapp/configuracao	- Buscar config
POST	/whatsapp/configuracao	- Salvar config
GET	/whatsapp/mensagens	- Histórico

3.12 Avaliações

Recursos Implementados

- ☒ Sistema de avaliação de produtos e pedidos
- ☒ Notas de 1 a 5 estrelas
- ☒ Comentários opcionais
- ☒ Listagem de avaliações por produto
- ☒ Média de avaliações
- ☒ Moderação de avaliações

Modelo Avaliacao

```
{
  id: string
  nota: int (1-5)
  comentario: string?
  produtoId: string
  usuarioId: string
  pedidoId: string?
}
```

Endpoints

GET	/avaliacoes	- Listar avaliações
POST	/avaliacoes	- Criar avaliação
GET	/avaliacoes/produto/:id	- Avaliações do produto
DELETE	/avaliacoes/:id	- Remover avaliação

3.13 Endereços

Recursos Implementados

- ☒ CRUD de endereços de usuários

- ☒ Validação de CEP
- ☒ Integração com ViaCEP para autocompletar
- ☒ Snapshot de endereço em pedidos

Modelo Endereco

```
{
  id: string
  usuarioId: string
  cep: string?
  rua: string?
  numero: string?
  complemento: string?
  bairro: string?
  cidade: string?
  uf: string?
}
```

3.14 Webhooks (Logging)

Recursos Implementados

- ☒ Log de webhooks recebidos
- ☒ Suporte a Stripe e Asaas
- ☒ Armazenamento de payload completo
- ☒ Status de processamento
- ☒ Registro de erros
- ☒ Filtros e busca

Modelo WebhookLog

```
{
  id: string
  origem: 'STRIPE' | 'ASAAS'
  evento: string
  payload: Json
  processado: boolean
  erro: string?
}
```

4. FUNCIONALIDADES PLANEJADAS (NÃO IMPLEMENTADAS)

4.1 Sistema de Tickets (Suporte)

Status: Planejado, não implementado

Descrição: Sistema completo de suporte ao cliente com tickets, mensagens, prioridades e status.

Funcionalidades Planejadas:

- Abertura de tickets por empresas
- Sistema de mensagens no ticket
- Níveis de prioridade (baixa, média, alta, urgente)
- Status (aberto, em andamento, resolvido, fechado)

- Área de suporte dedicada
- Notificações de novos tickets

4.2 Upload de Imagens para Cloud

Status: Planejado, não implementado

Descrição: Integração com Cloudinary ou AWS S3 para upload direto de imagens de produtos.

Funcionalidades Planejadas:

- Upload direto via frontend
- Redimensionamento automático
- Compressão de imagens
- CDN para entrega otimizada
- Thumbnails automáticos

Implementação Atual: URLs de imagens inseridas manualmente

4.3 Programa de Fidelidade

Status: Planejado, não implementado

Descrição: Sistema de pontos e recompensas para clientes fiéis.

Funcionalidades Planejadas:

- Acúmulo de pontos por compra
- Resgate de pontos
- Níveis de fidelidade
- Benefícios exclusivos
- Dashboard de pontos

4.4 Entrega com Rastreamento

Status: Planejado, não implementado

Descrição: Integração com serviços de logística para rastreamento em tempo real.

Funcionalidades Planejadas:

- Integração com Correios, Loggi, etc.
- Rastreamento em tempo real
- Notificações de posição
- Estimativa de tempo de entrega
- Histórico de rotas

4.5 Gestão de Promoções Avançadas

Status: Planejado, não implementado

Descrição: Sistema avançado de promoções e campanhas de marketing.

Funcionalidades Planejadas:

- Promoções por categoria
- Combos promocionais
- Compre X leve Y
- Frete grátis condicional
- Desconto progressivo
- Agendamento de promoções

4.6 Integração com Marketplaces

Status: Planejado, não implementado

Descrição: Integração com iFood, Rappi, Uber Eats para sincronização de cardápio e pedidos.

Funcionalidades Planejadas:

- Sincronização de produtos
- Importação de pedidos
- Atualização de estoque
- Gestão unificada
- Relatórios consolidados

4.7 Sistema de Reservas

Status: Planejado, não implementado

Descrição: Para restaurantes que aceitam reservas de mesas.

Funcionalidades Planejadas:

- Calendário de disponibilidade
- Gestão de mesas
- Confirmação por email/WhatsApp
- Histórico de reservas
- Lista de espera

4.8 Multi-idioma (i18n)

Status: Planejado, não implementado

Descrição: Suporte a múltiplos idiomas na plataforma.

Funcionalidades Planejadas:

- Seletor de idioma
- Traduções automáticas
- Gestão de traduções
- Suporte a PT, EN, ES inicialmente

4.9 App Mobile Nativo

Status: Planejado, não implementado

Descrição: Aplicativo nativo para iOS e Android.

Tecnologias Consideradas:

- React Native
- Flutter
- Notificações push
- Geolocalização
- Pagamento in-app

4.10 Análise Avançada (BI)

Status: Planejado, não implementado

Descrição: Business Intelligence com relatórios avançados e insights.

Funcionalidades Planejadas:

- Análise de cohort
- Previsão de demanda
- Análise de abandono de carrinho

- Lifetime Value do cliente
 - Relatórios customizáveis
 - Exportação para Excel/PDF
-

5. ESTRUTURA DE ARQUIVOS E HIERARQUIA

5.1 Backend (/backend)

```

backend/
├── prisma/
│   ├── migrations/                # Histórico de migrations
│   │   ├── 20251013124033_initial_schema/
│   │   │   ├── migration.sql
│   │   │   └── migration_lock.toml
│   ├── schema.prisma              # Schema do banco de dados
│   └── seed.ts                    # Script de seed (dados iniciais)
├── scripts/
│   ├── migrate.sh                 # Helper para migrations
│   └── seed.sh                    # Helper para seed
├── src/
│   ├── app.module.ts              # Módulo raiz da aplicação
│   ├── app.controller.ts          # Controller raiz (health check)
│   ├── app.service.ts             # Service raiz
│   └── main.ts                    # Entry point da aplicação
│   ├── common/                    # Recursos compartilhados
│   │   └── guards/
│   │       └── (guards compartilhados)
│   ├── database/                  # Módulo de banco de dados
│   │   ├── prisma.module.ts       # Módulo global do Prisma
│   │   └── prisma.service.ts      # Serviço singleton do Prisma
│   ├── decorators/                # Decorators customizados
│   │   ├── current-empresa.decorator.ts # @CurrentEmpresa()
│   │   ├── current-user.decorator.ts   # @CurrentUser()
│   │   ├── public.decorator.ts         # @Public()
│   │   └── roles.decorator.ts          # @Roles()
│   ├── filters/                   # Exception filters
│   │   └── all-exceptions.filter.ts    # Global exception filter
│   ├── guards/                    # Guards de autenticação e autorização
│   │   ├── jwt-auth.guard.ts         # Guard JWT padrão
│   │   └── roles.guard.ts            # Guard de roles
│   ├── middleware/                 # Middlewares
│   │   └── tenant.middleware.ts       # Multi-tenancy middleware
│   ├── utils/                      # Utilitários
│   │   ├── date.helpers.ts           # Helpers de data
│   │   ├── response.helpers.ts        # Helpers de resposta API
│   │   ├── validation.helpers.ts      # Helpers de validação
│   │   └── index.ts                  # Barrel export
│   └── modules/                    # Módulos de funcionalidade
│       ├── auth/                   # Autenticação e autorização
│       │   ├── dto/
│       │   │   ├── login.dto.ts
│       │   │   ├── signup.dto.ts
│       │   │   ├── refresh-token.dto.ts
│       │   │   └── create-account-from-order.dto.ts
│       │   └── interfaces/
│       │       ├── jwt-payload.interface.ts
│       │       └── auth-response.interface.ts
│       └── strategies/
│           └── jwt.strategy.ts

```

```

local.strategy.ts
auth.controller.ts
auth.service.ts
auth.module.ts

produtos/ # Gestão de produtos
  dto/
    create-produto.dto.ts
    update-produto.dto.ts
    filtrar-produtos.dto.ts
  produtos.controller.ts
  produtos.service.ts
  produtos.module.ts

carrinho/ # Carrinho de compras
  dto/
    add-item-carrinho.dto.ts
    update-item-carrinho.dto.ts
  carrinho.controller.ts
  carrinho.service.ts
  carrinho.module.ts

pagamentos/ # Gestão de pagamentos
  dto/
    create-pagamento.dto.ts
    asaas-webhook.dto.ts
  pagamentos.controller.ts
  pagamentos.service.ts
  asaas.service.ts # Integração Asaas
  pagamentos.module.ts

assinaturas/ # Gestão de assinaturas
  dto/
    create-assinatura.dto.ts
    update-assinatura.dto.ts
  assinaturas.controller.ts
  assinaturas.service.ts
  stripe.service.ts # Integração Stripe
  assinaturas.module.ts

whatsapp/ # Integração WhatsApp
  dto/
    enviar-mensagem.dto.ts
    config-whatsapp.dto.ts
  whatsapp.controller.ts
  whatsapp.service.ts
  whatsapp.module.ts

webhooks/ # Webhooks (Stripe, Asaas)
  webhooks.controller.ts
  webhooks.service.ts
  webhooks.module.ts

public/ # Endpoints públicos
  dto/
    create-pedido-publico.dto.ts
  public.controller.ts
  public.service.ts
  public.module.ts

pedidos/ # Gestão de pedidos
  dto/
    create-pedido.dto.ts

```

```

├── ┌───┐ update-status-pedido.dto.ts
├── ┌───┐ filtrar-pedidos.dto.ts
├── ┌───┐ pedidos.controller.ts
├── ┌───┐ pedidos.service.ts
├── ┌───┐ pedidos.module.ts
├── ┌───┐
├── ┌───┐ cupons/                                # Gestão de cupons
├── ┌───┐ ┌───┐ dto/
├── ┌───┐ ┌───┐ create-cupom.dto.ts
├── ┌───┐ ┌───┐ update-cupom.dto.ts
├── ┌───┐ ┌───┐ validar-cupom.dto.ts
├── ┌───┐ ┌───┐ cupons.controller.ts
├── ┌───┐ ┌───┐ cupons.service.ts
├── ┌───┐ ┌───┐ cupons.module.ts
├── ┌───┐
├── ┌───┐ avaliacoes/                          # Sistema de avaliações
├── ┌───┐ ┌───┐ dto/
├── ┌───┐ ┌───┐ create-avaliacao.dto.ts
├── ┌───┐ ┌───┐ avaliacoes.controller.ts
├── ┌───┐ ┌───┐ avaliacoes.service.ts
├── ┌───┐ ┌───┐ avaliacoes.module.ts
├── ┌───┐
├── ┌───┐ notificacoes/                       # Sistema de notificações
├── ┌───┐ ┌───┐ dto/
├── ┌───┐ ┌───┐ create-notificacao.dto.ts
├── ┌───┐ ┌───┐ notificacoes.controller.ts
├── ┌───┐ ┌───┐ notificacoes.service.ts
├── ┌───┐ ┌───┐ notificacoes.module.ts
├── ┌───┐
├── ┌───┐ dashboard/                          # Dashboard e métricas
├── ┌───┐ ┌───┐ dashboard.controller.ts
├── ┌───┐ ┌───┐ dashboard.service.ts
├── ┌───┐ ┌───┐ dashboard.module.ts
├── ┌───┐
├── ┌───┐ test/                               # Testes
├── ┌───┐ ┌───┐ (arquivos de teste)
├── ┌───┐
├── ┌───┐ .env.example                        # Template de variáveis de ambiente
├── ┌───┐ .gitignore
├── ┌───┐ docker-compose.yml                  # Redis container
├── ┌───┐ nest-cli.json                       # Configuração do NestJS CLI
├── ┌───┐ package.json
├── ┌───┐ tsconfig.json                       # Configuração TypeScript
├── ┌───┐ README.md                           # Documentação do backend
├── ┌───┐ SUPABASE-SETUP.md                   # Guia de setup do Supabase

```

5.2 Frontend (/)

```

src/
├── main.tsx # Entry point da aplicação
├── App.tsx # Componente raiz
├── index.css # Estilos globais (Tailwind)
├── theme.ts # Tokens de tema
├── vite-env.d.ts # Types do Vite
├──
├── auth/ # Autenticação
│   ├── AuthContext.tsx # Context de autenticação
│   ├── types.ts # Tipos de roles
│   └── useRoleGuard.ts # Hook de proteção por role
├── components/ # Componentes React
│   ├── index.ts # Barrel export
│   ├──
│   ├── auth/ # Componentes de autenticação
│   │   ├── RequireAuth.tsx # HOC de proteção de rotas
│   │   └── index.ts
│   ├──
│   ├── commerce/ # Componentes de e-commerce
│   │   ├── CartDrawer.tsx # Drawer do carrinho (lateral)
│   │   ├── CartDrawerBackend.tsx # Versão backend integrada
│   │   ├── ProductCard.tsx # Card de produto
│   │   ├── UpsellStrip.tsx # Strip de upsell/cross-sell
│   │   └── index.ts
│   ├──
│   ├── common/ # Componentes reutilizáveis
│   │   ├── Badge.tsx # Badge de status
│   │   ├── Button.tsx # Botão customizado
│   │   ├── Card.tsx # Card container
│   │   ├── Container.tsx # Container principal
│   │   ├── ErrorBoundary.tsx # Error boundary
│   │   ├── Input.tsx # Input customizado
│   │   ├── Loading.tsx # Indicador de loading
│   │   ├── types.ts # Tipos compartilhados
│   │   └── index.ts
│   ├──
│   ├── dashboard/ # Componentes de dashboard
│   │   ├── DateRangeFilter.tsx # Filtro de data
│   │   ├── SalesChart.tsx # Gráfico de vendas
│   │   └── index.ts
│   ├──
│   ├── layout/ # Layouts e navegação
│   │   ├── DashboardShell.tsx # Shell do dashboard
│   │   ├── PublicHeader.tsx # Header público
│   │   ├── StoreSidebar.tsx # Sidebar da loja
│   │   ├── StoreTopbarActions.tsx # Ações da topbar
│   │   ├── StorefrontHeader.tsx # Header da vitrine
│   │   ├── SuperAdminSidebar.tsx # Sidebar do super admin
│   │   ├── Topbar.tsx # Topbar principal
│   │   └── index.ts
│   ├──
│   ├── system/ # Componentes de sistema
│   │   ├── ImpersonationBanner.tsx # Banner de impersonação
│   │   └── index.ts
│   ├──
│   ├── ModalAvaliacao.tsx # Modal de avaliação
│   ├── NotificacoesDropdown.tsx # Dropdown de notificações
│   ├── PagamentoPix.tsx # Componente de pagamento PIX
│   └── README.md # Documentação de componentes
├── contexts/ # Context API

```

AssinaturaContext.tsx	# Context de assinatura
CartContext.tsx	# Context do carrinho
ClientAuthContext.tsx	# Context de auth do cliente
NotificacoesContext.tsx	# Context de notificações
hooks/	# Hooks customizados
index.ts	# Barrel export
useApi.ts	# Hook de chamadas API
useCart.ts	# Hook do carrinho
useDebounce.ts	# Hook de debounce
useForm.ts	# Hook de formulários
usePagination.ts	# Hook de paginação
README.md	# Documentação de hooks
layouts/	# Layouts de página
AdminLayout.tsx	# Layout administrativo
pages/	# Páginas da aplicação
public/	# Páginas públicas
Home.tsx	# Landing page
Login.tsx	# Login (mock)
LoginBackend.tsx	# Login (backend real)
storefront/	# Vitrine e checkout
Vitrine.tsx	# Listagem de produtos (mock)
VitrineBackend.tsx	# Listagem (backend real)
Checkout.tsx	# Checkout (mock)
CheckoutBackend.tsx	# Checkout (backend real)
ClientLogin.tsx	# Login de cliente
OrderConfirmation.tsx	# Confirmação (mock)
OrderConfirmationBackend.tsx	# Confirmação (backend)
admin/	# Área administrativa
store/	# Admin da loja (ADMIN_EMPRESA)
Dashboard.tsx	# Dashboard da loja
Products.tsx	# Gestão de produtos
ProductEdit.tsx	# Edição de produto
Orders.tsx	# Gestão de pedidos
Clients.tsx	# Gestão de clientes
ClientEdit.tsx	# Edição de cliente
Settings.tsx	# Configurações da loja
super/	# Admin superadmin
Dashboard.tsx	# Dashboard superadmin
Companies.tsx	# Gestão de empresas
Subscriptions.tsx	# Gestão de assinaturas
Tickets.tsx	# Sistema de tickets
Settings.tsx	# Configurações globais
Dashboard.tsx	# Dashboard unificado
Pedidos.tsx	# Pedidos da empresa
Cupons.tsx	# Gestão de cupons
ConfiguracaoWhatsApp.tsx	# Configuração WhatsApp
Webhooks.tsx	# Log de webhooks
assinaturas/	# Assinaturas
CheckoutAssinatura.tsx	# Checkout de assinatura
MinhaAssinatura.tsx	# Detalhes da assinatura
Planos.tsx	# Listagem de planos
cliente/	# Área do cliente

```

├── MeusPedidos.tsx      # Histórico de pedidos
├── MinhasAvaliacoes.tsx # Avaliações feitas
├── pagamentos/          # Pagamentos
│   ├── DetalhesPagamento.tsx # Detalhes de pagamento
│   └── HistoricoPagamentos.tsx # Histórico
├── support/             # Área de suporte
│   ├── Layout.tsx       # Layout de suporte
│   └── Tickets.tsx       # Sistema de tickets
├── routes/              # Configuração de rotas
│   ├── AppRouter.tsx     # Router principal
│   ├── admin.routes.tsx  # Rotas admin
│   ├── client.routes.tsx # Rotas cliente
│   ├── public.routes.tsx # Rotas públicas
│   ├── storefront.routes.tsx # Rotas vitrine
│   ├── super.routes.tsx  # Rotas superadmin
│   ├── support.routes.tsx # Rotas suporte
│   ├── types.ts          # Tipos de rotas
│   └── index.ts          # Barrel export
├── services/            # Serviços de API
│   ├── api.ts            # API principal (mock)
│   ├── api.types.ts      # Tipos de API
│   ├── api.utils.ts      # Utilitários de API
│   ├── apiClient.ts      # Cliente Axios configurado
│   ├── backendApi.ts     # API backend real
│   ├── dashboardApi.ts   # API de dashboard
│   └── index.ts          # Barrel export
├── ui/                  # UI system
│   ├── feedback/
│   │   └── ToastContext.tsx # Context de toast
│   └── theme/
│       └── tokens.ts        # Tokens de design
├── utils/               # Utilitários
│   ├── assinaturaGuards.ts # Guards de assinatura
│   ├── formatters.ts      # Formataadores (moeda, data)
│   ├── safeStorage.ts     # Wrapper seguro localStorage
│   └── statusColors.ts     # Cores de status
├── # Arquivos na raiz do frontend
├── index.html           # HTML principal
├── package.json          # Dependências frontend
├── vite.config.ts        # Configuração Vite
├── tailwind.config.js    # Configuração Tailwind
├── postcss.config.js     # Configuração PostCSS
├── tsconfig.json         # TypeScript config
├── tsconfig.app.json     # TypeScript app config
├── tsconfig.node.json    # TypeScript node config
├── eslint.config.js      # ESLint config
└── .gitignore

```


6. MODELOS DE DADOS (PRISMA SCHEMA)

6.1 Visão Geral dos Modelos

O schema Prisma define 16 modelos principais que representam as entidades do sistema:

1. **Empresa** - Empresas/Lojas multi-tenant
2. **Usuario** - Usuários do sistema (admins, clientes)
3. **Produto** - Produtos das lojas
4. **Pedido** - Pedidos de clientes
5. **ItemPedido** - Itens individuais dos pedidos
6. **RefreshToken** - Tokens de renovação JWT
7. **Carrinho** - Carrinhos de compra
8. **ItemCarrinho** - Itens dos carrinhos
9. **Cupom** - Cupons de desconto
10. **Avaliacao** - Avaliações de produtos/pedidos
11. **Notificacao** - Notificações de usuários
12. **Assinatura** - Assinaturas de empresas
13. **Pagamento** - Pagamentos (pedidos e assinaturas)
14. **WebhookLog** - Log de webhooks recebidos
15. **MensagemWhatsApp** - Mensagens WhatsApp
16. **Endereco** - Endereços de usuários

6.2 Enums

```
enum Role {  
  SUPER_ADMIN      // Administrador geral do sistema  
  ADMIN_EMPRESA    // Administrador de uma empresa  
  CLIENTE          // Cliente final  
}  
  
enum StatusPedido {  
  PENDENTE          // Pedido criado, aguardando processamento  
  CONFIRMADO        // Pedido confirmado pela empresa  
  EM_PREPARO        // Pedido sendo preparado  
  SAIU_ENTREGA      // Pedido saiu para entrega  
  ENTREGUE          // Pedido entregue  
  CANCELADO         // Pedido cancelado  
}
```

6.3 Modelos Detalhados

6.3.1 Empresa

```
model Empresa {
  id          String    @id @default(uuid())
  nome        String
  slug        String    @unique
  subdominio  String    @unique
  ativo       Boolean    @default(true)
  whatsappNumero String?
  whatsappToken String?
  asaasCustomerId String?
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  // Relações
  usuarios      Usuario[]
  produtos      Produto[]
  pedidos       Pedido[]
  carrinhos     Carrinho[]
  cupons        Cupom[]
  assinatura    Assinatura?
  pagamentos    Pagamento[]
  mensagensWhatsApp MensagemWhatsApp[]
}
```

Principais Campos:

- slug e subdominio são únicos para cada empresa
- asaasCustomerId para integração com gateway de pagamento
- Relações 1:N com a maioria das entidades do sistema

6.3.2 Usuario

```
model Usuario {
  id          String    @id @default(uuid())
  email       String    @unique
  senha       String    // Hash bcrypt
  nome        String
  telefone    String?
  cpf         String?
  role        Role      @default(CLIENTE)
  empresaId   String?   // Null para SUPER_ADMIN
  ativo       Boolean    @default(true)
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  // Relações
  empresa      Empresa?    @relation(fields: [empresaId], references: [id], onDelete: Cascade)
  pedidos      Pedido[]
  carrinho     Carrinho?
  avaliacoes   Avaliacao[]
  notificacoes Notificacao[]
  endereco     Endereco?

  @@index([empresaId])
}
```

Principais Campos:

- `role` define o nível de acesso (SUPER_ADMIN, ADMIN_EMPRESA, CLIENTE)
- `empresaId` é null para SUPER_ADMIN (acesso multi-tenant)
- Relação 1:1 com Carrinho e Endereco

6.3.3 Produto

```

model Produto
  id          String    @id @default(uuid())
  nome        String
  descricao   String?
  preco       Decimal   @db.Decimal(10, 2)
  imagem      String?
  ativo       Boolean   @default(true)
  empresaId   String
  estoque     Int       @default(0)
  categoria   String?
  createdAt   DateTime  @default(now())
  updatedAt   DateTime  @updatedAt

  // Relações
  empresa      Empresa      @relation(fields: [empresaId], references: [id], onDelete: Cascade)
  itensCarrinho ItemCarrinho[]
  itensPedido  ItemPedido[]
  avaliacoes   Avaliacao[]

  @@index([empresaId])
  @@index([ativo])
  @@index([empresaId, ativo])
  @@index([empresaId, categoria])
}

```

Índices:

- Índices compostos para otimizar queries multi-tenant
- `[empresaId, ativo]` para listar produtos ativos por empresa
- `[empresaId, categoria]` para filtros por categoria

6.3.4 Pedido

```

model Pedido {
  id                String           @id @default(uuid())
  numero            String           @unique // Número sequencial do pedido
  status            StatusPedido    @default(PENDENTE)
  subtotal          Decimal          @db.Decimal(10, 2)
  desconto          Decimal          @default(0) @db.Decimal(10, 2)
  total             Decimal          @db.Decimal(10, 2)
  frete             Decimal          @default(0) @db.Decimal(10, 2)
  clienteId         String
  empresaId         String
  enderecoEntrega   String?
  formaPagamento    String?
  cupomDesconto      String?
  observacoes        String?
  createdAt          DateTime        @default(now())
  updatedAt          DateTime        @updatedAt

  // Relações
  cliente            Usuario          @relation(fields: [clienteId], references:
[id], onDelete: Cascade)
  empresa            Empresa          @relation(fields: [empresaId], references:
[id], onDelete: Cascade)
  itens              ItemPedido[]
  avaliacoes         Avaliacao[]
  notificacoes        Notificacao[]
  pagamentos          Pagamento[]
  mensagensWhatsApp  MensagemWhatsApp[]

  @@index([clienteId])
  @@index([empresaId])
  @@index([status])
  @@index([empresaId, createdAt])
  @@index([empresaId, status])
  @@index([createdAt])
}

```

Cálculos:

- total = subtotal - desconto + frete
- subtotal é calculado a partir dos itens
- desconto aplicado via cupom

6.3.5 ItemPedido

```
model ItemPedido {
  id          String    @id @default(uuid())
  pedidoId    String
  produtoId   String
  quantidade  Int
  precoUnitario Decimal @db.Decimal(10, 2)
  subtotal    Decimal @db.Decimal(10, 2)
  observacoes String?
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  // Relações
  pedido Pedido @relation(fields: [pedidoId], references: [id], onDelete: Cascade)
  produto Produto @relation(fields: [produtoId], references: [id], onDelete: Cascade)

  @@index([pedidoId])
  @@index([produtoId])
  @@index([pedidoId, produtoId])
}
```

Snapshot de Preço:

- `precoUnitario` é snapshot do preço no momento do pedido
- Mesmo que o produto mude de preço, o pedido mantém o histórico

6.3.6 Carrinho e ItemCarrinho

```

model Carrinho {
  id          String    @id @default(uuid())
  usuarioId   String    @unique
  empresaId   String
  createdAt   DateTime  @default(now())
  updatedAt   DateTime  @updatedAt

  usuario Usuario      @relation(fields: [usuarioId], references: [id], onDelete: Cascade)
  empresa Empresa      @relation(fields: [empresaId], references: [id], onDelete: Cascade)
  itens       ItemCarrinho[]

  @@index([usuarioId])
  @@index([empresaId])
}

model ItemCarrinho {
  id          String    @id @default(uuid())
  carrinhoId   String
  produtoId    String
  quantidade    Int      @default(1)
  precoUnitario Decimal  @db.Decimal(10, 2)
  observacoes  String?
  createdAt    DateTime  @default(now())
  updatedAt    DateTime  @updatedAt

  carrinho Carrinho @relation(fields: [carrinhoId], references: [id], onDelete: Cascade)
  produto  Produto  @relation(fields: [produtoId], references: [id], onDelete: Cascade)

  @@index([carrinhoId])
  @@index([produtoId])
}

```

6.3.7 Cupom

```
model Cupom {
  id          String    @id @default(uuid())
  codigo      String
  descricao   String?
  tipo        String    // PERCENTUAL, VALOR_FIXO
  valor       Decimal   @db.Decimal(10, 2)
  valorMinimo Decimal?  @db.Decimal(10, 2)
  dataInicio  DateTime
  dataFim     DateTime
  ativo       Boolean   @default(true)
  usoMaximo   Int?
  usoAtual    Int       @default(0)
  empresaId   String
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  empresa Empresa @relation(fields: [empresaId], references: [id], onDelete: Cascade)

  @@unique([empresaId, codigo])
  @@index([empresaId])
  @@index([empresaId, ativo])
}
```

Validações:

- Data atual entre dataInicio e dataFim
- usoAtual < usoMaximo (se definido)
- Valor do pedido >= valorMinimo (se definido)

6.3.8 Assinatura

```
model Assinatura {
  id          String    @id @default(uuid())
  empresaId   String    @unique
  plano       String    // BASICO, PROFISSIONAL, ENTERPRISE
  status      String    // ATIVA, CANCELADA, SUSPENSA, TRIAL
  stripeCustomerId String?
  stripeSubscriptionId String?
  dataInicio  DateTime
  dataFim     DateTime?
  valorMensal Float
  proximaCobranca DateTime?
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  empresa Empresa @relation(fields: [empresaId], references: [id], onDelete: Cascade)
  pagamentos Pagamento[]

  @@index([empresaId])
  @@index([status])
}
```

6.3.9 Pagamento

```

model Pagamento {
  id          String      @id @default(uuid())
  empresaId   String
  assinaturaId String?
  pedidoId    String?
  tipo        String      // ASSINATURA, PEDIDO
  metodo      String      // PIX, CARTAO, BOLETO
  status      String      // PENDENTE, APROVADO, RECUSADO, CANCELADO
  valor       Float
  asaasPaymentId String?
  asaasInvoiceUrl String?
  pixQrCode   String?
  pixCopyPaste String?
  boletoUrl   String?
  dataVencimento DateTime?
  dataPagamento DateTime?
  createdAt   DateTime    @default(now())
  updatedAt   DateTime    @updatedAt

  empresa      Empresa      @relation(fields: [empresaId], references: [id],
onDelete: Cascade)
  assinatura    Assinatura?  @relation(fields: [assinaturaId], references: [id], onDelete: SetNull)
  pedido        Pedido?     @relation(fields: [pedidoId], references: [id], onDelete: SetNull)

  @@index([empresaId])
  @@index([assinaturaId])
  @@index([pedidoId])
  @@index([status])
}

```

6.3.10 Outros Modelos

RefreshToken:

```

model RefreshToken {
  id      String      @id @default(uuid())
  token    String      @unique
  usuarioId String
  expiresAt DateTime
  createdAt DateTime @default(now())

  @@index([usuarioId])
  @@index([token])
}

```

Avaliacao:


```

model Avaliacao {
    id          String    @id @default(uuid())
    nota        Int       // 1-5
    comentario  String?
    produtoId   String
    usuarioId   String
    pedidoId    String?
    createdAt   DateTime @default(now())

    produto Produto @relation(fields: [produtoId], references: [id], onDelete: Cascade)
    usuario Usuario @relation(fields: [usuarioId], references: [id], onDelete: Cascade)
    pedido Pedido? @relation(fields: [pedidoId], references: [id], onDelete: SetNull)

    @@index([produtoId])
    @@index([usuarioId])
    @@index([pedidoId])
}

```

Notificacao:

```

model Notificacao {
    id          String    @id @default(uuid())
    titulo      String
    mensagem    String
    tipo        String    // PEDIDO, SISTEMA, PROMOCAO
    lida        Boolean    @default(false)
    usuarioId   String
    pedidoId    String?
    createdAt   DateTime @default(now())

    usuario Usuario @relation(fields: [usuarioId], references: [id], onDelete: Cascade)
    pedido  Pedido? @relation(fields: [pedidoId], references: [id], onDelete: SetNull)

    @@index([usuarioId])
    @@index([pedidoId])
    @@index([usuarioId, lida])
}

```

WebhookLog:

```

model WebhookLog {
    id          String    @id @default(uuid())
    origem      String    // STRIPE, ASAAS
    evento      String
    payload     Json
    processado  Boolean    @default(false)
    erro        String?
    createdAt   DateTime @default(now())

    @@index([origem])
    @@index([processado])
}

```

MensagemWhatsApp:

```

model MensagemWhatsApp {
  id          String    @id @default(uuid())
  empresaId   String
  pedidoId    String?
  telefone    String
  mensagem    String
  tipo        String    // NOTIFICACAO, CHAT
  direcao     String    // ENVIADA, RECEBIDA
  status      String    // PENDENTE, ENVIADA, ENTREGUE, LIDA, ERRO
  whatsappId  String?
  erro        String?
  createdAt   DateTime  @default(now())

  empresa Empresa @relation(fields: [empresaId], references: [id], onDelete: Cascade)
  pedido  Pedido? @relation(fields: [pedidoId], references: [id], onDelete: SetNull)

  @@index([empresaId])
  @@index([pedidoId])
  @@index([status])
}

```

Endereco:

```

model Endereco {
  id          String    @id @default(uuid())
  usuarioId   String    @unique
  cep         String?
  rua         String?
  numero      String?
  complemento String?
  bairro      String?
  cidade      String?
  uf          String?
  createdAt   DateTime  @default(now())
  updatedAt   DateTime  @updatedAt

  usuario Usuario @relation(fields: [usuarioId], references: [id], onDelete: Cascade)

  @@index([usuarioId])
}

```

6.4 Relacionamentos Principais

```

Empresa (1) ———< (N) Usuario
Empresa (1) ———< (N) Produto
Empresa (1) ———< (N) Pedido
Empresa (1) ———< (N) Carrinho
Empresa (1) ———< (N) Cupom
Empresa (1) ——— (1) Assinatura

Usuario (1) ———< (N) Pedido
Usuario (1) ——— (1) Carrinho
Usuario (1) ——— (1) Endereco
Usuario (1) ———< (N) Avaliacao
Usuario (1) ———< (N) Notificacao

Pedido (1) ———< (N) ItemPedido
Pedido (1) ———< (N) Pagamento
Pedido (1) ———< (N) Avaliacao

Carrinho (1) ———< (N) ItemCarrinho

Produto (1) ———< (N) ItemPedido
Produto (1) ———< (N) ItemCarrinho
Produto (1) ———< (N) Avaliacao

Assinatura (1) ———< (N) Pagamento

```

7. ENDPOINTS DA API

7.1 Base URL

```

Desenvolvimento: http://localhost:3000
Produção: https://delivere-backend.onrender.com

```

7.2 Estrutura de Resposta Padrão

```

// Resposta de Sucesso
{
  success: true,
  data: { ... },
  message?: string
}

// Resposta de Erro
{
  success: false,
  error: {
    code: string,
    message: string,
    details?: any
  }
}

```

7.3 Autenticação (Público)

POST /auth/login

Realizar login no sistema.

Request:

```
{
  "email": "admin@pizzaexpress.com",
  "senha": "admin123"
}
```

Response (200):

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIs... ",
  "refreshToken": "550e8400-e29b-41d4-a716-446655440000",
  "user": {
    "id": "uuid",
    "email": "admin@pizzaexpress.com",
    "nome": "Admin Pizza Express",
    "role": "ADMIN_EMPRESA",
    "empresaId": "uuid"
  }
}
```

POST /auth/signup

Criar nova conta de usuário.

Request:

```
{
  "email": "novo@email.com",
  "senha": "senha123",
  "nome": "Novo Usuário",
  "telefone": "11999999999",
  "empresaId": "uuid" // Opcional para cliente
}
```

Response (201):

```
{
  "accessToken": "...",
  "refreshToken": "...",
  "user": { ... }
}
```

POST /auth/refresh

Renovar access token usando refresh token.

Request:

```
{
  "refreshToken": "550e8400-e29b-41d4-a716-446655440000"
}
```

Response (200):

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIs... ",
  "refreshToken": "550e8400-e29b-41d4-a716-446655440000"
}
```

POST /auth/logout

Realizar logout e invalidar refresh token.

Headers:

```
Authorization: Bearer {accessToken}
```

Request:

```
{
  "refreshToken": "550e8400-e29b-41d4-a716-446655440000"
}
```

Response (200):

```
{
  "message": "Logout realizado com sucesso"
}
```

POST /auth/create-account-from-order

Criar conta a partir de um checkout sem cadastro.

Request:

```
{
  "email": "cliente@email.com",
  "nome": "Cliente Novo",
  "telefone": "11999999999",
  "pedidoId": "uuid"
}
```

7.4 Produtos (Autenticado)**GET /produtos**

Listar produtos da empresa do usuário autenticado.

Headers:

```
Authorization: Bearer {accessToken}
```

Query Parameters:

- `page` (number): Página atual (padrão: 1)
- `limit` (number): Itens por página (padrão: 10)
- `search` (string): Busca por nome
- `categoria` (string): Filtrar por categoria
- `ativo` (boolean): Filtrar por status

Response (200):

```
{
  "data": [
    {
      "id": "uuid",
      "nome": "Pizza Margherita",
      "descricao": "Molho de tomate, mussarela, manjericão",
      "preco": 35.90,
      "imagem": "https://...",
      "ativo": true,
      "estoque": 50,
      "categoria": "Pizzas",
      "empresaId": "uuid",
      "createdAt": "2025-10-01T10:00:00Z"
    }
  ],
  "meta": {
    "total": 100,
    "page": 1,
    "limit": 10,
    "totalPages": 10
  }
}
```

POST /produtos

Criar novo produto.

Headers:

```
Authorization: Bearer {accessToken}
```

Request:

```
{
  "nome": "Pizza Margherita",
  "descricao": "Molho de tomate, mussarela, manjericão",
  "preco": 35.90,
  "imagem": "https://...",
  "estoque": 50,
  "categoria": "Pizzas"
}
```

Response (201):

```
{
  "id": "uuid",
  "nome": "Pizza Margherita",
  ...
}
```

GET /produtos/:id

Buscar produto por ID.

Headers:

```
Authorization: Bearer {accessToken}
```

Response (200):

```
{
  "id": "uuid",
  "nome": "Pizza Margherita",
  ...
}
```

PATCH /produtos/:id

Atualizar produto.

Headers:

```
Authorization: Bearer {accessToken}
```

Request:

```
{
  "preco": 39.90,
  "estoque": 45
}
```

DELETE /produtos/:id

Soft delete (desativar) produto.

Headers:

```
Authorization: Bearer {accessToken}
```

Response (200):

```
{
  "message": "Produto desativado com sucesso"
}
```

DELETE /produtos/:id/hard

Hard delete (remover permanentemente) - apenas SUPER_ADMIN.

Headers:

```
Authorization: Bearer {accessToken}
```

7.5 Público (Sem autenticação)

GET /public/:slug/info

Informações da loja.

Response (200):

```
{
  "id": "uuid",
  "nome": "Pizza Express",
  "slug": "pizza-express",
  "subdominio": "pizza-express",
  "ativo": true
}
```

GET /public/:slug/produtos

Listar produtos da loja pública.

Query Parameters:

- page , limit , search , categoria

Response (200):

```
{
  "data": [...],
  "meta": { ... }
}
```

GET /public/:slug/produtos/:id

Detalhes do produto público.

GET /public/:slug/categorias

Listar categorias disponíveis.

Response (200):

```
{
  "categorias": ["Pizzas", "Bebidas", "Sobremesas"]
}
```

7.6 Pedidos (Autenticado)

GET /pedidos

Listar pedidos da empresa.

Headers:


```
Authorization: Bearer {accessToken}
```

Query Parameters:

- `page`, `limit`
- `status` (StatusPedido): Filtrar por status
- `dataInicio`, `dataFim` (ISO date): Filtro de período

Response (200):

```
{
  "data": [
    {
      "id": "uuid",
      "numero": "001",
      "status": "CONFIRMADO",
      "total": 85.90,
      "cliente": {
        "id": "uuid",
        "nome": "Cliente",
        "email": "cliente@email.com"
      },
      "itens": [
        {
          "id": "uuid",
          "produtoId": "uuid",
          "quantidade": 2,
          "precoUnitario": 35.90,
          "subtotal": 71.80
        }
      ],
      "createdAt": "2025-10-16T14:30:00Z"
    }
  ],
  "meta": { ... }
}
```

POST /pedidos

Criar novo pedido.

Headers:

```
Authorization: Bearer {accessToken}
```

Request:

```
{
  "clienteId": "uuid",
  "itens": [
    {
      "produtoId": "uuid",
      "quantidade": 2,
      "observacoes": "Sem cebola"
    }
  ],
  "enderecoEntrega": "Rua X, 123",
  "formaPagamento": "PIX",
  "cupomDesconto": "PROMO10",
  "observacoes": "Entregar no portão"
}
```

Response (201):

```
{
  "id": "uuid",
  "numero": "002",
  "status": "PENDENTE",
  "subtotal": 71.80,
  "desconto": 7.18,
  "total": 64.62,
  "itens": [ ... ]
}
```

GET /pedidos/:id

Buscar pedido por ID.

PATCH /pedidos/:id/status

Atualizar status do pedido.

Request:

```
{
  "status": "EM_PREPARO"
}
```

DELETE /pedidos/:id

Cancelar pedido.

7.7 Carrinho (Autenticado)**GET /carrinho**

Buscar carrinho do usuário.

Headers:

```
Authorization: Bearer {accessToken}
```

Response (200):

```
{
  "id": "uuid",
  "usuarioId": "uuid",
  "empresaId": "uuid",
  "itens": [
    {
      "id": "uuid",
      "produto": {
        "id": "uuid",
        "nome": "Pizza Margherita",
        "preco": 35.90
      },
      "quantidade": 2,
      "precoUnitario": 35.90,
      "subtotal": 71.80
    }
  ],
  "totalItens": 2,
  "subtotal": 71.80
}
```

POST /carrinho/itens

Adicionar item ao carrinho.

Request:

```
{
  "produtoId": "uuid",
  "quantidade": 2,
  "observacoes": "Sem cebola"
}
```

PATCH /carrinho/itens/:id

Atualizar quantidade do item.

Request:

```
{
  "quantidade": 3
}
```

DELETE /carrinho/itens/:id

Remover item do carrinho.

DELETE /carrinho

Limpar carrinho.

POST /carrinho/checkout

Converter carrinho em pedido.

Request:

```
{
  "enderecoEntrega": "Rua X, 123",
  "formaPagamento": "PIX",
  "cupomDesconto": "PROM010",
  "observacoes": "...",
}
```

7.8 Cupons (Autenticado)

GET /cupons

Listar cupons da empresa.

Headers:

```
Authorization: Bearer {accessToken}
```

Response (200):

```
{
  "data": [
    {
      "id": "uuid",
      "codigo": "PROM010",
      "descricao": "10% de desconto",
      "tipo": "PERCENTUAL",
      "valor": 10,
      "valorMinimo": 50,
      "dataInicio": "2025-10-01T00:00:00Z",
      "dataFim": "2025-10-31T23:59:59Z",
      "ativo": true,
      "usoMaximo": 100,
      "usoAtual": 45
    }
  ]
}
```

POST /cupons

Criar cupom.

GET /cupons/:id

Buscar cupom.

PATCH /cupons/:id

Atualizar cupom.

DELETE /cupons/:id

Remover cupom.

POST /cupons/validar

Validar cupom para uso.

Request:

```
{
  "codigo": "PROMO10",
  "valorPedido": 75.00
}
```

Response (200):

```
{
  "valido": true,
  "cupom": { ... },
  "descontoCalculado": 7.50
}
```

7.9 Pagamentos (Autenticado)

POST /pagamentos

Criar pagamento.

Request:

```
{
  "pedidoId": "uuid",
  "metodo": "PIX",
  "valor": 85.90
}
```

Response (201):

```
{
  "id": "uuid",
  "status": "PENDENTE",
  "metodo": "PIX",
  "valor": 85.90,
  "pixQrCode": "data:image/png;base64,...",
  "pixCopyPaste": "00020126580014br.gov.bcb.pix..."
}
```

GET /pagamentos/:id

Buscar pagamento.

GET /pagamentos/pedido/:id

Pagamentos do pedido.

7.10 Assinaturas (SuperAdmin)

GET /assinaturas

Listar assinaturas.

Headers:

```
Authorization: Bearer {accessToken}
X-User-Role: SUPER_ADMIN
```

POST /assinaturas

Criar assinatura.

GET /assinaturas/:id

Buscar assinatura.

PATCH /assinaturas/:id

Atualizar assinatura.

DELETE /assinaturas/:id

Cancelar assinatura.

7.11 Dashboard (Autenticado)**GET /dashboard/metricas**

Métricas gerais da empresa.

Query Parameters:

- `dataInicio` , `dataFim` (ISO date)

Response (200):

```
{
  "vendasHoje": 1250.00,
  "vendasMes": 45800.00,
  "pedidosHoje": 25,
  "pedidosMes": 450,
  "crescimentoVendas": 15.5,
  "crescimentoPedidos": 12.3,
  "ticketMedio": 101.78
}
```

GET /dashboard/vendas

Vendas por período para gráfico.

Response (200):

```
{
  "vendas": [
    { "data": "2025-10-01", "valor": 1250.00 },
    { "data": "2025-10-02", "valor": 1450.00 },
    ...
  ]
}
```

GET /dashboard/produtos-top

Produtos mais vendidos.

Response (200):

```
{
  "produtos": [
    {
      "id": "uuid",
      "nome": "Pizza Margherita",
      "quantidadeVendida": 125,
      "valorTotal": 4487.50
    },
    ...
  ]
}
```

7.12 Notificações (Autenticado)

GET /notificacoes

Listar notificações do usuário.

Query Parameters:

- `lida` (boolean): Filtrar por lidas/não lidas

Response (200):

```
{
  "data": [
    {
      "id": "uuid",
      "titulo": "Novo pedido",
      "mensagem": "Pedido #001 foi realizado",
      "tipo": "PEDIDO",
      "lida": false,
      "pedidoId": "uuid",
      "createdAt": "2025-10-16T14:30:00Z"
    }
  ],
  "naoLidas": 5
}
```

POST /notificacoes

Criar notificação.

PATCH /notificacoes/:id/lida

Marcar como lida.

PATCH /notificacoes/ler-todas

Marcar todas como lidas.

DELETE /notificacoes/:id

Remover notificação.

7.13 WhatsApp (Autenticado)

POST /whatsapp/enviar

Enviar mensagem WhatsApp.

Request:

```
{
  "telefone": "5511999999999",
  "mensagem": "Seu pedido #001 está confirmado!",
  "pedidoId": "uuid"
}
```

GET /whatsapp/configuracao

Buscar configuração WhatsApp da empresa.

POST /whatsapp/configuracao

Salvar configuração WhatsApp.

Request:

```
{
  "whatsappNumero": "5511999999999",
  "whatsappToken": "token_da_api"
}
```

GET /whatsapp/mensagens

Histórico de mensagens.

7.14 Avaliações (Autenticado)

GET /avaliacoes

Listar avaliações.

POST /avaliacoes

Criar avaliação.

Request:

```
{
  "produtoId": "uuid",
  "pedidoId": "uuid",
  "nota": 5,
  "comentario": "Excelente produto!"
}
```

GET /avaliacoes/produto/:id

Avaliações do produto.

DELETE /avaliacoes/:id

Remover avaliação.

7.15 Webhooks (Público)

POST /webhooks/asaas

Webhook Asaas para confirmação de pagamentos.

Headers:


```
asaas-access-token: {token}
```

POST /webhooks/stripe

Webhook Stripe para sincronização de assinaturas.

Headers:

```
stripe-signature: {signature}
```

8. INTEGRAÇÕES EXTERNAS

8.1 Supabase (PostgreSQL)

Descrição: Plataforma de backend-as-a-service que fornece banco de dados PostgreSQL gerenciado.

Uso no Projeto:

- Banco de dados principal
- Connection pooling para melhor performance
- Backups automáticos
- Dashboard de gestão

Configuração:

```
DATABASE_URL="postgresql://postgres:[PROJECT-REF]:[PASSWORD]@[HOST]:6543/postgres?pg-bouncer=true"
```

Recursos Utilizados:

- PostgreSQL 15
- Connection pooling (PgBouncer)
- SSL/TLS por padrão

8.2 Render

Descrição: Plataforma cloud para hospedagem de aplicações e bancos de dados.

Uso no Projeto:

- Hospedagem do backend (Web Service)
- PostgreSQL gerenciado (alternativa ao Supabase)
- Deploy automático via GitHub
- SSL/TLS gratuito

Configuração:

- **Web Service:** Backend Node.js
- **Build Command:** `cd backend && npm install && npx prisma generate && npm run build`
- **Start Command:** `cd backend && npm run start:prod`
- **Environment Variables:** DATABASE_URL, JWT_SECRET, etc.

8.3 Asaas (Gateway de Pagamento)

Descrição: Gateway de pagamento brasileiro que suporta PIX, Cartão de Crédito e Boleto.

Uso no Projeto:

- Processamento de pagamentos de pedidos
- Geração de QR Code PIX
- Geração de boletos
- Webhook para confirmação de pagamento

Endpoints Utilizados:

- `POST /v3/payments` - Criar pagamento
- `GET /v3/payments/:id` - Consultar pagamento
- `POST /v3/customers` - Criar cliente

Configuração:

```
ASAAS_API_KEY="your_api_key"  
ASAAS_API_URL="https://sandbox.asaas.com/api/v3" # ou production
```

Webhook:

```
POST /webhooks/asaas
```

Eventos Suportados:

- `PAYMENT_RECEIVED` - Pagamento confirmado
- `PAYMENT_OVERDUE` - Pagamento vencido
- `PAYMENT_DELETED` - Pagamento cancelado

8.4 Stripe (Pagamentos Recorrentes)

Descrição: Plataforma global de pagamentos, utilizada para assinaturas recorrentes das empresas.

Uso no Projeto:

- Gestão de assinaturas (planos mensais)
- Cobrança recorrente automática
- Gerenciamento de clientes (empresas)
- Webhook para sincronização

Endpoints Utilizados:

- `POST /v1/customers` - Criar cliente
- `POST /v1/subscriptions` - Criar assinatura
- `POST /v1/subscriptions/:id` - Atualizar assinatura
- `DELETE /v1/subscriptions/:id` - Cancelar assinatura

Configuração:

```
STRIPE_SECRET_KEY="sk_test_..."  
STRIPE_WEBHOOK_SECRET="whsec_..."
```

Webhook:

```
POST /webhooks/stripe
```

Eventos Suportados:

- `customer.subscription.created` - Assinatura criada
- `customer.subscription.updated` - Assinatura atualizada
- `customer.subscription.deleted` - Assinatura cancelada
- `invoice.payment_succeeded` - Pagamento bem-sucedido
- `invoice.payment_failed` - Pagamento falhou

8.5 WhatsApp Business API

Descrição: API oficial do WhatsApp para envio de mensagens empresariais.

Uso no Projeto:

- Notificações de pedido
- Confirmações
- Atualizações de status
- Suporte ao cliente

Configuração:

```
// Armazenado no banco por empresa
{
  whatsappNumero: "5511999999999",
  whatsappToken: "token_da_api"
}
```

Funcionalidades:

- Envio de mensagens de texto
- Templates pré-aprovados
- Rastreamento de status de entrega
- Histórico de mensagens

8.6 Redis

Descrição: Banco de dados in-memory usado para cache e sessões.

Uso no Projeto:

- Cache de queries frequentes
- Armazenamento de sessões
- Rate limiting
- Cache de tokens JWT invalidados (logout)

Configuração (Docker):

```
# docker-compose.yml
services:
  redis:
    image: redis:7-alpine
    ports:
      - "6379:6379"
    volumes:
      - redis-data:/data
```

Configuração (App):

```
REDIS_HOST=localhost
REDIS_PORT=6379
REDIS_PASSWORD= # Opcional
```

8.7 ViaCEP (Integração Planejada)

Descrição: API pública para consulta de CEP brasileiro.

Uso no Projeto:

- Autocompletar endereço ao digitar CEP
- Validação de CEP

Endpoint:

```
GET https://viacep.com.br/ws/{cep}/json/
```

8.8 Cloudinary/AWS S3 (Integração Planejada)

Descrição: Serviço de armazenamento e otimização de imagens.

Uso no Projeto (Futuro):

- Upload direto de imagens de produtos
- Redimensionamento automático
- Compressão
- CDN para entrega otimizada

9. SEGURANÇA E MULTI-TENANCY

9.1 Arquitetura Multi-tenant

O sistema implementa **multi-tenancy isolado** onde cada empresa tem seus próprios dados completamente separados através do `empresaId`.

9.1.1 Estratégias de Isolamento

1. Database-level Isolation (Prisma)

```
// Todas as queries incluem empresaId automaticamente
await prisma.produto.findMany({
  where: {
    empresaId: currentEmpresa.id,
    ativo: true
  }
});
```

2. Tenant Resolution (Middleware)

O `TenantMiddleware` extrai o tenant de duas formas:

Método 1: Subdomínio

```

pizza-express.delivre.com.br → Empresa: Pizza Express
burger-king.delivre.com.br → Empresa: Burger King

```

Método 2: Slug na URL (Público)

```

/public/pizza-express/produtos → Empresa: Pizza Express
/public/burger-king/produtos → Empresa: Burger King

```

Implementação:

```

// backend/src/middleware/tenant.middleware.ts
export class TenantMiddleware implements NestMiddleware {
  async use(req: Request, res: Response, next: NextFunction) {
    const host = req.hostname;
    const subdomain = this.extractSubdomain(host);

    if (subdomain) {
      const empresa = await this.prisma.empresa.findUnique({
        where: { subdominio: subdomain }
      });
      req['empresa'] = empresa;
    }

    // Alternativa: slug na URL
    const slug = req.params.slug;
    if (slug) {
      const empresa = await this.prisma.empresa.findUnique({
        where: { slug }
      });
      req['empresa'] = empresa;
    }

    next();
  }
}

```

3. Decorator @CurrentEmpresa()

```

@Get()
async listarProdutos(@CurrentEmpresa() empresa: Empresa) {
  return this.produtosService.findAll(empresa.id);
}

```

9.1.2 Validação de Tenant

Todas as operações validam se o usuário pertence ao tenant:








```
// backend/src/utils/validation.helpers.ts
export class TenantValidator {
  static validateAccess(user: Usuario, empresaId: string): void {
    if (user.role === 'SUPER_ADMIN') {
      return; // Super admin tem acesso a tudo
    }

    if (user.empresaId !== empresaId) {
      throw new ForbiddenException('Acesso negado a este recurso');
    }
  }
}
```

9.1.3 Correções de Vazamento Multi-tenant

Durante o desenvolvimento, foram identificados e corrigidos **7 vazamentos de dados**:

Vazamentos Corrigidos:

1.  Dashboard sem filtro de empresa
2.  Listagem de pedidos sem empresald
3.  Cupons acessíveis entre empresas
4.  Notificações compartilhadas
5.  Carrinho sem validação de empresa
6.  Produtos visíveis entre empresas
7.  Avaliações sem isolamento

Exemplo de Correção:

```
// ANTES (vulnerável)
async findAll() {
  return this.prisma.pedido.findMany();
}

// DEPOIS (seguro)
async findAll(empresaId: string) {
  return this.prisma.pedido.findMany({
    where: { empresaId }
  });
}
```

9.2 Autenticação e Autorização

9.2.1 JWT (JSON Web Tokens)

Estratégia:

- Access Token: Curta duração (15 minutos)
- Refresh Token: Longa duração (7 dias)

Payload do JWT:

```
interface JwpPayload {
  sub: string;      // userId
  email: string;
  role: Role;
  empresaId?: string; // Null para SUPER_ADMIN
  iat: number;      // Issued at
  exp: number;      // Expiration
}
```

Geração:

```
const accessToken = this.jwtService.sign(payload, {
  expiresIn: '15m'
});

const refreshToken = uuid();
await this.prisma.refreshToken.create({
  data: {
    token: refreshToken,
    usuarioId: user.id,
    expiresAt: new Date(Date.now() + 7 * 24 * 60 * 60 * 1000) // 7 dias
  }
});
```

9.2.2 Guards

JwtAuthGuard:

```
@Injectable()
export class JwtAuthGuard extends AuthGuard('jwt') {
  canActivate(context: ExecutionContext) {
    const request = context.switchToHttp().getRequest();
    const isPublic = this.reflector.get('isPublic', context.getHandler());

    if (isPublic) {
      return true;
    }

    return super.canActivate(context);
  }
}
```

RolesGuard:

```
@Injectable()
export class RolesGuard implements CanActivate {
  canActivate(context: ExecutionContext): boolean {
    const requiredRoles = this.reflector.get<Role[]>('roles', context.getHandler());

    if (!requiredRoles) {
      return true;
    }

    const { user } = context.switchToHttp().getRequest();
    return requiredRoles.some(role => user.role === role);
  }
}
```

Uso:

```

@Controller('produtos')
@UseGuards(JwtAuthGuard, RolesGuard)
export class ProdutosController {
  @Get()
  @Roles('ADMIN_EMPRESA', 'SUPER_ADMIN')
  async findAll() { ... }

  @Post()
  @Roles('ADMIN_EMPRESA')
  async create() { ... }
}

```

9.2.3 Roles e Permissões**SUPER_ADMIN:**

- Acesso total ao sistema
- Gestão de empresas
- Gestão de assinaturas
- Visualização de métricas globais
- Hard delete de recursos

ADMIN_EMPRESA:

- Acesso apenas à própria empresa
- Gestão de produtos
- Gestão de pedidos
- Gestão de clientes
- Visualização de métricas da empresa
- Configurações da loja

CLIENTE:

- Visualização de vitrine
- Gestão do próprio carrinho
- Criação de pedidos
- Histórico de pedidos
- Avaliações

9.3 Proteção de Dados Sensíveis**9.3.1 Hash de Senhas****Bcrypt com salt rounds = 10:**

```

import * as bcrypt from 'bcrypt';

async hashPassword(senha: string): Promise<string> {
  const salt = await bcrypt.genSalt(10);
  return bcrypt.hash(senha, salt);
}

async comparePassword(senha: string, hash: string): Promise<boolean> {
  return bcrypt.compare(senha, hash);
}

```


9.3.2 Sanitização de Respostas

Remove campos sensíveis:

```
// Nunca expor senha no retorno
const { senha, ...userWithoutPassword } = user;
return userWithoutPassword;

// Helper de resposta
export class ResponseHelpers {
  static sanitizeUser(user: Usuario) {
    const { senha, ...sanitized } = user;
    return sanitized;
  }
}
```

9.3.3 Validação de Input

class-validator nos DTOs:

```
export class CreateUsuarioDto {
  @IsEmail()
  email: string;

  @IsString()
  @MinLength(6)
  @MaxLength(100)
  senha: string;

  @IsString()
  @MinLength(3)
  nome: string;

  @IsOptional()
  @IsPhoneNumber('BR')
  telefone?: string;
}
```

9.4 CORS e Origem Permitida

Configuração:

```
// main.ts
app.enableCors({
  origin: [
    'http://localhost:3001',
    'https://delivereit-frontent.vercel.app',
    /\.delivereit\.com\.br$/ // Subdomínios
  ],
  credentials: true,
  methods: ['GET', 'POST', 'PUT', 'PATCH', 'DELETE'],
  allowedHeaders: ['Content-Type', 'Authorization']
});
```

9.5 Rate Limiting (Planejado)

Implementação Futura:

```
import { ThrottlerModule } from '@nestjs/throttler';

@Module({
  imports: [
    ThrottlerModule.forRoot({
      ttl: 60,      // 60 segundos
      limit: 10,    // 10 requisições
    }),
  ],
})
```

9.6 SQL Injection Prevention

Prisma ORM:

- Queries parametrizadas automaticamente
- Escape automático de valores
- Type-safe queries

```
// SEGURO: Prisma escapa automaticamente
const produtos = await prisma.produto.findMany({
  where: {
    nome: {
      contains: userInput // Escapado automaticamente
    }
  }
});
```

9.7 Logs e Auditoria

NestJS Logger:

```
export class ProdutosService {
  private readonly logger = new Logger(ProdutosService.name);

  async create(data: CreateProdutoDto, empresaId: string) {
    this.logger.log(`Criando produto: ${data.nome} para empresa ${empresaId}`);
    // ...
  }
}
```

Webhook Logging:

```
// Todos os webhooks são logados
await prisma.webhookLog.create({
  data: {
    origem: 'ASAAS',
    evento: event.type,
    payload: event.data,
    processado: false
  }
});
```

10. OTIMIZAÇÕES E PERFORMANCE

10.1 Backend Optimizations

10.1.1 Queries Prisma Otimizadas

Problema Eliminado: N+1 Queries

ANTES (N+1):

```
// 1 query para buscar pedidos
const pedidos = await prisma.pedido.findMany();

// N queries para buscar itens de cada pedido
for (const pedido of pedidos) {
  const itens = await prisma.itemPedido.findMany({
    where: { pedidoId: pedido.id }
  });
}
// Total: 1 + N queries
```

DEPOIS (Otimizado):

```
// 1 query com include
const pedidos = await prisma.pedido.findMany({
  include: {
    itens: {
      include: {
        produto: true
      }
    },
    cliente: {
      select: {
        id: true,
        nome: true,
        email: true
      }
    }
  }
});
// Total: 1 query
```

10.1.2 Índices Compostos para Multi-tenancy

Schema Prisma Otimizado:

```

model Produto {
  // ... campos

  @@index([empresaId])           // Busca por empresa
  @@index([ativo])               // Busca por status
  @@index([empresaId, ativo])    // Combinado
  @@index([empresaId, categoria]) // Filtro por categoria
}

model Pedido {
  // ... campos

  @@index([empresaId])
  @@index([status])
  @@index([empresaId, createdAt]) // Ordenação por data
  @@index([empresaId, status])   // Dashboard filtros
  @@index([createdAt])           // Relatórios globais
}

model ItemPedido {
  // ... campos

  @@index([pedidoId])
  @@index([produtoId])
  @@index([pedidoId, produtoId]) // Combinado para joins
}

```

Impacto:

- Redução de 70% no tempo de queries multi-tenant
- Dashboard 5x mais rápido
- Listagem de pedidos 3x mais rápida

10.1.3 Transações Prisma**Operações Atômicas:**

```

async createPedido(data: CreatePedidoDto, empresaId: string, clienteId: string) {
  return await this.prisma.$transaction(async (tx) => {
    // 1. Criar pedido
    const pedido = await tx.pedido.create({
      data: {
        empresaId,
        clienteId,
        status: 'PENDENTE',
        subtotal: 0,
        total: 0
      }
    });

    // 2. Adicionar itens
    let subtotal = 0;
    for (const item of data.itens) {
      const produto = await tx.produto.findUnique({
        where: { id: item.produtoId }
      });

      // Verificar estoque
      if (produto.estoque < item.quantidade) {
        throw new BadRequestException('Estoque insuficiente');
      }

      // Criar item do pedido
      await tx.itemPedido.create({
        data: {
          pedidoId: pedido.id,
          produtoId: item.produtoId,
          quantidade: item.quantidade,
          precoUnitario: produto.preco,
          subtotal: produto.preco.mul(item.quantidade)
        }
      });

      // Decrementar estoque
      await tx.produto.update({
        where: { id: item.produtoId },
        data: {
          estoque: {
            decrement: item.quantidade
          }
        }
      });

      subtotal += produto.preco.toNumber() * item.quantidade;
    }

    // 3. Atualizar total do pedido
    const total = subtotal - (data.desconto || 0) + (data.frete || 0);

    return await tx.pedido.update({
      where: { id: pedido.id },
      data: {
        subtotal,
        desconto: data.desconto || 0,
        frete: data.frete || 0,
        total
      },
      include: {
        itens: true
      }
    });
  });
}

```

```

    }
  });
});
}

```

Benefícios:

- Atomicidade: Tudo ou nada
- Consistência de dados
- Rollback automático em caso de erro
- Prevenção de race conditions

10.1.4 Respostas API Padronizadas

Helper Functions:

```

// backend/src/utils/response.helpers.ts
export class ResponseHelpers {
  static success<T>(data: T, message?: string) {
    return {
      success: true,
      data,
      ...(message && { message })
    };
  }

  static error(message: string, code?: string, details?: any) {
    return {
      success: false,
      error: {
        code: code || 'UNKNOWN_ERROR',
        message,
        ...(details && { details })
      }
    };
  }

  static paginated<T>(data: T[], meta: PaginationMeta) {
    return {
      success: true,
      data,
      meta: {
        total: meta.total,
        page: meta.page,
        limit: meta.limit,
        totalPages: Math.ceil(meta.total / meta.limit)
      }
    };
  }
}

```

Uso:

```
@Get()
async findAll(@Query() query: FiltrarProdutosDto, @CurrentEmpresa() empresa: Empresa)
{
  const { data, total } = await this.produtosService.findAll(empresa.id, query);

  return ResponseHelpers.paginated(data, {
    total,
    page: query.page || 1,
    limit: query.limit || 10
  });
}
```

10.1.5 Error Handling Centralizado

All Exceptions Filter:

```
// backend/src/filters/all-exceptions.filter.ts
@Catch()
export class AllExceptionsFilter implements ExceptionFilter {
  catch(exception: unknown, host: ArgumentsHost) {
    const ctx = host.switchToHttp();
    const response = ctx.getResponse<Response>();
    const request = ctx.getRequest<Request>();

    let status = HttpStatus.INTERNAL_SERVER_ERROR;
    let message = 'Internal server error';
    let code = 'INTERNAL_ERROR';

    if (exception instanceof HttpException) {
      status = exception.getStatus();
      message = exception.message;
    } else if (exception instanceof Prisma.PrismaClientKnownRequestError) {
      ({ status, message, code } = this.handlePrismaError(exception));
    }

    response.status(status).json({
      success: false,
      error: {
        code,
        message,
        timestamp: new Date().toISOString(),
        path: request.url
      }
    });
  }

  private handlePrismaError(error: Prisma.PrismaClientKnownRequestError) {
    switch (error.code) {
      case 'P2002':
        return {
          status: HttpStatus.CONFLICT,
          message: 'Registro duplicado',
          code: 'DUPLICATE_ERROR'
        };
      case 'P2025':
        return {
          status: HttpStatus.NOT_FOUND,
          message: 'Registro não encontrado',
          code: 'NOT_FOUND'
        };
      default:
        return {
          status: HttpStatus.BAD_REQUEST,
          message: 'Erro de validação',
          code: 'VALIDATION_ERROR'
        };
    }
  }
}
```

10.2 Frontend Optimizations (PR #35)

10.2.1 Code Splitting com Lazy Loading

Implementação:


```
// src/routes/AppRouter.tsx
import { lazy, Suspense } from 'react';
import { Loading } from '@components/common';

// Lazy load de rotas
const Dashboard = lazy(() => import('@pages/admin/Dashboard'));
const Products = lazy(() => import('@pages/admin/store/Products'));
const Vitrine = lazy(() => import('@pages/storefront/VitrineBackend'));

export function AppRouter() {
  return (
    <Suspense fallback={<Loading />}>
      <Routes>
        <Route path="/admin/dashboard" element={<Dashboard />} />
        <Route path="/admin/products" element={<Products />} />
        <Route path="/loja/:slug" element={<Vitrine />} />
      </Routes>
    </Suspense>
  );
}
```

Impacto:

- Bundle inicial: 850KB → 520KB (-39%)
- First Contentful Paint: -2.5s
- Time to Interactive: -3.1s

10.2.2 Chunking de Vendors**vite.config.ts:**

```
export default defineConfig({
  build: {
    rollupOptions: {
      output: {
        manualChunks: {
          'react-vendor': ['react', 'react-dom', 'react-router-dom'],
          'date-vendor': ['date-fns'],
          'chart-vendor': ['recharts']
        }
      }
    },
    minify: 'terser',
    terserOptions: {
      compress: {
        drop_console: true,
        drop_debugger: true
      }
    }
  }
});
```

Benefícios:

- Melhor caching (vendors mudam raramente)
- Carregamentos subsequentes 5x mais rápidos
- Redução de re-downloads

10.2.3 Context Optimization**ANTES (re-renders desnecessários):**

```
export function CartContext({ children }) {
  const [items, setItems] = useState([]);

  const value = {
    items,
    addItem,
    removeItem
    // Novo objeto a cada render!
  };

  return <Context.Provider value={value}>{children}</Context.Provider>;
}
```

DEPOIS (otimizado):

```
export function CartContext({ children }) {
  const [items, setItems] = useState([]);

  // Memoizar callbacks
  const addItem = useCallback((item) => {
    setItems(prev => [...prev, item]);
  }, []);

  const removeItem = useCallback((id) => {
    setItems(prev => prev.filter(item => item.id !== id));
  }, []);

  // Memoizar valores computados
  const total = useMemo(() => {
    return items.reduce((sum, item) => sum + item.price * item.quantity, 0);
  }, [items]);

  // Memoizar value object
  const value = useMemo(() => ({
    items,
    total,
    addItem,
    removeItem
  }), [items, total, addItem, removeItem]);

  return <Context.Provider value={value}>{children}</Context.Provider>;
}
```

Impacto:

- 70% menos re-renders
- Componentes filhos re-renderizam apenas quando necessário

10.2.4 Error Boundary

Implementação:

```
// src/components/common/ErrorBoundary.tsx
export class ErrorBoundary extends Component<Props, State> {
  componentDidCatch(error: Error, errorInfo: ErrorInfo) {
    console.error('Error caught by boundary:', error, errorInfo);
    // Enviar para Sentry/LogRocket
  }

  render() {
    if (this.state.hasError) {
      return (
        <div className="error-fallback">
          <h1>Oops! Algo deu errado</h1>
          <button onClick={this.handleReset}>Tentar novamente</button>
        </div>
      );
    }

    return this.props.children;
  }
}
```

Uso:

```
// src/App.tsx
<ErrorBoundary>
  <Suspense fallback={<Loading />}>
    <AppRouter />
  </Suspense>
</ErrorBoundary>
```

10.2.5 API Client Otimizado**Interceptors e Timeout:**

```
// src/services/apiClient.ts
const apiClient = axios.create({
  baseURL: import.meta.env.VITE_API_URL,
  timeout: 30000, // 30 segundos
  headers: {
    'Content-Type': 'application/json'
  }
});

// Request interceptor (adicionar token)
apiClient.interceptors.request.use(
  (config) => {
    const token = safeStorage.get('token');
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (error) => Promise.reject(error)
);

// Response interceptor (tratamento de erros)
apiClient.interceptors.response.use(
  (response) => response,
  async (error) => {
    if (error.response?.status === 401) {
      // Token expirado, tentar refresh
      const refreshToken = safeStorage.get('refreshToken');
      if (refreshToken) {
        try {
          const { data } = await axios.post('/auth/refresh', { refreshToken });
          safeStorage.set('token', data.accessToken);
          // Retry original request
          return apiClient(error.config);
        } catch (refreshError) {
          // Refresh falhou, fazer logout
          safeStorage.clear();
          window.location.href = '/login';
        }
      }
    }
    return Promise.reject(error);
  }
);
```

10.3 Database Optimizations

10.3.1 Connection Pooling

Supabase com PgBouncer:

```
DATABASE_URL="postgresql://...?pgbouncer=true"
```

Configuração Prisma:

```
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}
```

Pooling Settings:

- Pool mode: Transaction
- Default pool size: 20
- Max client connections: 100

10.3.2 Query Optimization Examples**Dashboard Vendas (Otimizado):**

```
// Agregação eficiente com Prisma
const vendas = await prisma.pedido.groupBy({
  by: ['createdAt'],
  where: {
    empresaId,
    createdAt: {
      gte: dataInicio,
      lte: dataFim
    },
    status: {
      notIn: ['CANCELADO']
    }
  },
  _sum: {
    total: true
  },
  _count: {
    id: true
  },
  orderBy: {
    createdAt: 'asc'
  }
});
```

Produtos Mais Vendidos:

```

const produtosTop = await prisma.itemPedido.groupBy({
  by: ['produtoId'],
  where: {
    pedido: {
      empresaId,
      status: 'ENTREGUE',
      createdAt: {
        gte: dataInicio
      }
    }
  },
  _sum: {
    quantidade: true,
    subtotal: true
  },
  orderBy: {
    _sum: {
      quantidade: 'desc'
    }
  },
  take: 10
});

// Buscar detalhes dos produtos (1 query)
const produtoIds = produtosTop.map(p => p.produtoId);
const produtos = await prisma.produto.findMany({
  where: {
    id: { in: produtoIds }
  },
  select: {
    id: true,
    nome: true,
    imagem: true,
    categoria: true
  }
});

```

10.4 Performance Metrics

10.4.1 Backend Performance

Antes das Otimizações:

- Tempo médio de resposta: ~800ms
- Queries por request: 15-20 (N+1)
- Dashboard load time: 3.5s

Depois das Otimizações:

- Tempo médio de resposta: ~200ms (-75%)
- Queries por request: 2-3 (includes)
- Dashboard load time: 0.8s (-77%)

10.4.2 Frontend Performance

Antes (PR #33):

- Initial bundle: 850 KB
- First Contentful Paint: 4.2s
- Time to Interactive: 5.8s
- Lighthouse Score: 65/100

Depois (PR #35):

- Initial bundle: 520 KB (-39%)
- First Contentful Paint: 1.7s (-60%)
- Time to Interactive: 2.7s (-53%)
- Lighthouse Score: 92/100

10.5 Monitoring e Observabilidade (Planejado)

Ferramentas Recomendadas:

- **Sentry:** Error tracking
- **LogRocket:** Session replay
- **New Relic / Datadog:** APM
- **Prometheus + Grafana:** Métricas

Métricas a Monitorar:

- Response time por endpoint
- Taxa de erro (4xx, 5xx)
- Throughput (req/s)
- Database query time
- Memory usage
- CPU usage



REFERÊNCIAS E DOCUMENTAÇÃO

Documentos do Projeto

- `README.md` - Guia principal do projeto
- `backend/README.md` - Documentação do backend
- `backend/SUPABASE-SETUP.md` - Guia de setup Supabase
- `OPTIMIZATIONS.md` - Detalhes de otimizações PR #35
- `FASE-2-PLANEJAMENTO.md` - Planejamento Fase 2
- `RELATORIO_FINAL_PROJETO.md` - Estado final do projeto
- `ACTION_PLAN.md` - Plano de ação e tarefas

Documentação Externa

- [NestJS Documentation](https://docs.nestjs.com) (https://docs.nestjs.com)
- [Prisma Documentation](https://www.prisma.io/docs) (https://www.prisma.io/docs)
- [React Documentation](https://react.dev) (https://react.dev)
- [Supabase Documentation](https://supabase.com/docs) (https://supabase.com/docs)



STATUS DO PROJETO

Último Update: 16 de outubro de 2025, 18:15 UTC



Completado

- [x] Fase 1: Fundação (Auth, Produtos, Multi-tenancy)
- [x] Fase 2: Features Core (Pedidos, Carrinho, Pagamentos)

- [x] Fase 3: Features Avançadas (Cupons, Notificações, Dashboard)
- [x] Fase 4: Integrações (WhatsApp, Asaas, Stripe)
- [x] Refatoração Backend (Performance, Segurança)
- [x] Refatoração Frontend Parte 1 (Organização, Hooks)
- [x] Refatoração Frontend Parte 2 (Otimizações) - PR #35



Deploy

- Backend: Render.com
- Frontend: Vercel (planejado)
- Database: Supabase PostgreSQL



Próximos Passos

1. Merge PR #35 (aguardando aprovação)
2. Deploy frontend para produção
3. Implementar monitoramento (Sentry)
4. Adicionar testes automatizados
5. Implementar sistema de tickets
6. Upload de imagens (Cloudinary)
7. App mobile (React Native)

FIM DO DOCUMENTO

Este documento foi compilado automaticamente a partir da análise completa do código-fonte, documentação existente e histórico do projeto DELIVEREI v1.