

FASE 2 - Carrinho e Checkout - Implementação Completa

Resumo da Implementação

A Fase 2 do projeto DELIVEREI foi implementada com sucesso, adicionando funcionalidades completas de carrinho de compras e checkout ao sistema.

Alterações no Banco de Dados

Novos Models Criados

1. Carrinho

```
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[]
}
```

Características:

- Um carrinho por usuário (relação 1:1)
- Multi-tenancy: cada carrinho pertence a uma empresa
- Cascade delete: carrinho é deletado quando usuário é removido

2. ItemCarrinho

```
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)
}
```

Características:

- Armazena preço unitário no momento da adição (histórico de preços)
- Suporta observações personalizadas por item
- Quantidade mínima de 1

3. 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

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

Características:

- Registra itens do pedido com preços históricos
- Calcula e armazena subtotal por item
- Mantém observações do carrinho

Alterações no Model Pedido

Campos adicionados:

- subtotal : Valor total dos itens antes de descontos
- desconto : Valor do desconto aplicado
- enderecoEntrega : Endereço de entrega do pedido
- formaPagamento : Forma de pagamento escolhida
- cupomDesconto : Código do cupom aplicado (se houver)
- Relação itens : Array de ItemPedido

Migration SQL

Arquivo: `prisma/migrations/20251008182048_fase_2_carrinho_checkout/migration.sql`

A migration inclui:

- Criação das tabelas `carrinhos`, `itens_carrinho` e `itens_pedido`
- Alteração da tabela `pedidos` com novos campos
- Criação de índices para otimização de queries
- Configuração de foreign keys com cascade delete

**Endpoints Implementados****Base URL**

Todos os endpoints requerem autenticação JWT e respeitam multi-tenancy através do header `X-Tenant-ID`.

1. Obter Carrinho

```
GET /carrinho
Authorization: Bearer {token}
X-Tenant-ID: {empresaId}
```

Resposta:

```
{
  "id": "uuid",
  "usuarioId": "uuid",
  "empresaId": "uuid",
  "createdAt": "2025-10-08T18:00:00.000Z",
  "updatedAt": "2025-10-08T18:00:00.000Z",
  "itens": [
    {
      "id": "uuid",
      "carrinhoId": "uuid",
      "produtoId": "uuid",
      "quantidade": 2,
      "precoUnitario": "25.90",
      "observacoes": "Sem cebola",
      "produto": {
        "id": "uuid",
        "nome": "Pizza Margherita",
        "descricao": "Pizza tradicional",
        "preco": "25.90",
        "imagem": "url",
        "categoria": "Pizzas"
      }
    }
  ],
  "subtotal": 51.80,
  "totalItens": 2
}
```

2. Adicionar Item ao Carrinho

```
POST /carrinho/itens
Authorization: Bearer {token}
X-Tenant-ID: {empresaId}
Content-Type: application/json
```

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

Validações:

- Produto deve existir e estar ativo
- Produto deve pertencer à empresa (multi-tenancy)
- Estoque deve ser suficiente
- Se item já existe, incrementa quantidade

Resposta:

```
{
  "id": "uuid",
  "carrinhoId": "uuid",
  "produtoId": "uuid",
  "quantidade": 2,
  "precoUnitario": "25.90",
  "observacoes": "Sem cebola",
  "produto": {
    "id": "uuid",
    "nome": "Pizza Margherita",
    "preco": "25.90"
  }
}
```

3. Atualizar Item do Carrinho

```
PATCH /carrinho/itens/:id
Authorization: Bearer {token}
X-Tenant-ID: {empresaId}
Content-Type: application/json

{
  "quantidade": 3,
  "observacoes": "Sem cebola e sem tomate"
}
```

Validações:

- Item deve pertencer ao carrinho do usuário
- Estoque deve ser suficiente para nova quantidade
- Multi-tenancy verificado

4. Remover Item do Carrinho

```
DELETE /carrinho/itens/:id
Authorization: Bearer {token}
X-Tenant-ID: {empresaId}
```

Resposta:

```
{
  "message": "Item removido com sucesso"
}
```

5. Limpar Carrinho

```
DELETE /carrinho
Authorization: Bearer {token}
X-Tenant-ID: {empresaId}
```

Resposta:

```
{  
  "message": "Carrinho limpo com sucesso"  
}
```

6. Checkout (Finalizar Pedido)

```
POST /carrinho/checkout  
Authorization: Bearer {token}  
X-Tenant-ID: {empresaId}  
Content-Type: application/json  
  
{  
  "enderecoEntrega": "Rua das Flores, 123 - Centro - São Paulo/SP",  
  "formaPagamento": "Cartão de Crédito",  
  "cupomDesconto": "PRIMEIRACOMPRA",  
  "observacoes": "Entregar após as 18h"  
}
```

Processo:

1. Valida que carrinho não está vazio
2. Verifica estoque de todos os itens
3. Calcula subtotal, desconto e total
4. Cria pedido com status PENDENTE
5. Cria itens do pedido
6. Atualiza estoque dos produtos
7. Limpa carrinho
8. Retorna pedido criado

Resposta:

```
{
  "id": "uuid",
  "numero": "PED-1728408000000",
  "status": "PENDENTE",
  "subtotal": "51.80",
  "desconto": "0.00",
  "total": "51.80",
  "clienteId": "uuid",
  "empresaId": "uuid",
  "enderecoEntrega": "Rua das Flores, 123 - Centro - São Paulo/SP",
  "formaPagamento": "Cartão de Crédito",
  "cupomDesconto": "PRIMEIRACOMPRA",
  "observacoes": "Entregar após as 18h",
  "createdAt": "2025-10-08T18:00:00.000Z",
  "itens": [
    {
      "id": "uuid",
      "pedidoId": "uuid",
      "produtoId": "uuid",
      "quantidade": 2,
      "precoUnitario": "25.90",
      "subtotal": "51.80",
      "observacoes": "Sem cebola",
      "produto": {
        "id": "uuid",
        "nome": "Pizza Margherita"
      }
    }
  ]
}
```

7. Obter Recomendações

```
GET /carrinho/recomendacoes
Authorization: Bearer {token}
X-Tenant-ID: {empresaId}
```

Lógica de Recomendação:

- Se carrinho vazio: retorna produtos mais recentes
- Se carrinho com itens:
 - Busca produtos da mesma categoria
 - Exclui produtos já no carrinho
 - Ordena por preço (maior primeiro - upsell)
 - Retorna até 5 produtos

Resposta:

```
[
  {
    "id": "uuid",
    "nome": "Pizza Calabresa",
    "descricao": "Pizza com calabresa",
    "preco": "28.90",
    "imagem": "url",
    "categoria": "Pizzas",
    "estoque": 50
  }
]
```

Segurança e Multi-tenancy

Validações Implementadas

1. **Autenticação JWT:** Todos os endpoints protegidos
2. **Multi-tenancy:**
 - Header `X-Tenant-ID` obrigatório
 - Validação de que produtos pertencem à empresa
 - Usuário só acessa seu próprio carrinho
3. **Validação de Estoque:** Verificado em todas as operações
4. **Validação de Propriedade:** Usuário só manipula seus próprios itens

Transações

O checkout utiliza transação do Prisma para garantir:

- Atomicidade: tudo ou nada
- Consistência: estoque sempre correto
- Isolamento: sem race conditions

```
await this.prisma.$transaction(async (tx) => {
  // Criar pedido
  // Criar itens do pedido
  // Atualizar estoque
  // Limpar carrinho
});
```

Regras de Negócio

Carrinho

- Um carrinho por usuário por empresa
- Carrinho criado automaticamente no primeiro acesso
- Preço unitário congelado no momento da adição
- Quantidade mínima: 1

Checkout

- Carrinho não pode estar vazio
- Todos os produtos devem ter estoque suficiente
- Número do pedido gerado automaticamente: `PED-{timestamp}`
- Status inicial: PENDENTE

- Estoque decrementado automaticamente
- Carrinho limpo após checkout bem-sucedido

Recomendações

- Baseadas em categoria dos produtos no carrinho
- Produtos com maior preço primeiro (upsell)
- Máximo de 5 recomendações
- Exclui produtos já no carrinho



Testes Sugeridos

Fluxo Completo de Teste

1. Login

```
curl -X POST http://localhost:3000/auth/login \  
-H "Content-Type: application/json" \  
-H "X-Tenant-ID: {empresaId}" \  
-d '{  
  "email": "cliente@example.com",  
  "senha": "senha123"  
}
```

1. Listar Produtos Disponíveis

```
curl -X GET http://localhost:3000/public/produtos \  
-H "X-Tenant-ID: {empresaId}"
```

1. Adicionar Produto ao Carrinho

```
curl -X POST http://localhost:3000/carrinho/itens \  
-H "Authorization: Bearer {token}" \  
-H "X-Tenant-ID: {empresaId}" \  
-H "Content-Type: application/json" \  
-d '{  
  "produtoId": "{produtoId}",  
  "quantidade": 2,  
  "observacoes": "Sem cebola"  
}
```

1. Ver Carrinho

```
curl -X GET http://localhost:3000/carrinho \  
-H "Authorization: Bearer {token}" \  
-H "X-Tenant-ID: {empresaId}"
```

1. Atualizar Quantidade


```
curl -X PATCH http://localhost:3000/carrinho/itens/{itemId} \
-H "Authorization: Bearer {token}" \
-H "X-Tenant-ID: {empresaId}" \
-H "Content-Type: application/json" \
-d '{
  "quantidade": 3
}'
```

1. Ver Recomendações







```
curl -X GET http://localhost:3000/carrinho/recomendacoes \
-H "Authorization: Bearer {token}" \
-H "X-Tenant-ID: {empresaId}"
```

1. Fazer Checkout







```
curl -X POST http://localhost:3000/carrinho/checkout \
-H "Authorization: Bearer {token}" \
-H "X-Tenant-ID: {empresaId}" \
-H "Content-Type: application/json" \
-d '{
  "enderecoEntrega": "Rua das Flores, 123",
  "formaPagamento": "Cartão de Crédito",
  "observacoes": "Entregar após as 18h"
}'
```

Casos de Teste

Testes Positivos

-  Adicionar item ao carrinho
-  Atualizar quantidade de item
-  Remover item do carrinho
-  Limpar carrinho completo
-  Fazer checkout com sucesso
-  Ver recomendações

Testes Negativos

-  Adicionar produto inexistente
-  Adicionar produto de outra empresa
-  Adicionar quantidade maior que estoque
-  Fazer checkout com carrinho vazio
-  Atualizar item de outro usuário
-  Checkout sem estoque suficiente

Estrutura de Arquivos Criados

```

backend/
├── prisma/
│   ├── schema.prisma (atualizado)
│   └── migrations/
│       └── 20251008182048_fase_2_carrinho_checkout/
│           └── migration.sql
├── src/
│   ├── app.module.ts (atualizado)
│   └── modules/
│       └── carrinho/
│           ├── carrinho.module.ts
│           ├── carrinho.controller.ts
│           ├── carrinho.service.ts
│           └── dto/
│               ├── index.ts
│               ├── adicionar-item-carrinho.dto.ts
│               ├── atualizar-item-carrinho.dto.ts
│               └── checkout.dto.ts
└── FASE-2-IMPLEMENTACA0.md
  
```

Próximos Passos (Fase 3)

Sugestões para a próxima fase:

1. Sistema de Cupons

- Model Cupom com validações
- Tipos de desconto (percentual, fixo)
- Regras de uso (mínimo, máximo, validade)

2. Gestão de Pedidos

- Endpoints para listar pedidos
- Atualizar status do pedido
- Histórico de pedidos do cliente
- Dashboard de pedidos para admin

3. Notificações

- Email de confirmação de pedido
- Notificações de mudança de status
- SMS/WhatsApp para atualizações

4. Relatórios

- Produtos mais vendidos
- Receita por período
- Análise de carrinho abandonado

Notas Técnicas

Dependências Adicionadas

- @nestjs/swagger@7.4.2 : Documentação automática da API
- swagger-ui-express : Interface Swagger UI

Considerações de Performance

- Índices criados em campos frequentemente consultados
- Uso de transações para operações críticas
- Eager loading de relações necessárias

Melhorias Futuras

- ☐ Implementar lógica real de cupons de desconto
- ☐ Adicionar validação de CEP no endereço
- ☐ Implementar carrinho abandonado (notificações)
- ☐ Cache de recomendações
- ☐ Histórico de preços dos produtos
- ☐ Limite de itens por carrinho
- ☐ Validação de horário de funcionamento



Checklist de Implementação

- ☒ Schema Prisma atualizado
- ☒ Migration SQL criada
- ☒ Models Carrinho e ItemCarrinho implementados
- ☒ Model ItemPedido implementado
- ☒ Model Pedido atualizado
- ☒ DTOs criados e validados
- ☒ CarrinhoService implementado
- ☒ CarrinhoController implementado
- ☒ CarrinhoModule criado
- ☒ Integração com AppModule
- ☒ Multi-tenancy implementado
- ☒ Validações de segurança
- ☒ Sistema de recomendações
- ☒ Lógica de checkout com transação
- ☒ Atualização de estoque
- ☒ Documentação completa
- ☒ Build sem erros



Conclusão

A Fase 2 foi implementada com sucesso, adicionando funcionalidades robustas de carrinho e checkout ao sistema DELIVEREI. O código está pronto para testes e integração com o frontend.

Status: ☒ COMPLETO E PRONTO PARA MERGE