












Integração Frontend com Backend - FASE 1 + FASE 2

Resumo

Este documento descreve a integração completa do frontend React com o backend Node.js/Express do projeto DELIVEREI, implementando todas as funcionalidades das FASE 1 e FASE 2.

Status da Integração

Concluído

-  Serviço de API com Axios configurado
-  Interceptors para autenticação e refresh token
-  Context de Carrinho integrado com backend
-  Página de Login com seleção de empresa (tenant)
-  Página de Vitrine com listagem de produtos do backend
-  Carrinho funcional com API real
-  Sistema de recomendações integrado
-  Página de Checkout completa
-  Página de Confirmação de Pedido
-  Rotas configuradas
-  Build funcionando

Estrutura Criada

Novos Arquivos

```
src/
├── services/
│   ├── apiClient.ts          # Cliente Axios configurado
│   └── backendApi.ts         # API endpoints tipados
├── contexts/
│   └── CartContext.tsx       # Context do carrinho integrado
├── pages/
│   ├── public/
│   │   └── LoginBackend.tsx  # Login com backend real
│   └── storefront/
│       ├── VitrineBackend.tsx      # Vitrine integrada
│       ├── CheckoutBackend.tsx     # Checkout integrado
│       └── OrderConfirmationBackend.tsx # Confirmação integrada
├── components/
│   └── commerce/
│       └── CartDrawerBackend.tsx    # Drawer do carrinho integrado
```

Arquivos Modificados

- `src/App.tsx` - Adicionado CartProvider
- `src/routes/AppRouter.tsx` - Adicionadas rotas backend

- `src/pages/public/Home.tsx` - Link para login backend
- `package.json` - Adicionado axios

Configuração

1. Dependências Instaladas

```
npm install axios
```

2. Variáveis de Ambiente

O frontend está configurado para se conectar ao backend em:

```
http://localhost:3000/api
```

3. Multi-tenancy

O sistema suporta multi-tenancy através do header `x-tenant-slug` :

- Pizza Express: `pizza-express`
- Burger King: `burger-king`

Como Executar

Pré-requisitos

1. Backend rodando na porta 3000
2. Node.js instalado
3. Dependências instaladas

Iniciar o Frontend

```
# Navegar para o diretório do projeto
cd /home/ubuntu/github_repos/delivre-i-v1

# Instalar dependências (se necessário)
npm install

# Iniciar em modo desenvolvimento
npm run dev

# Ou fazer build para produção
npm run build
npm run preview
```

URLs Disponíveis

- **Frontend**: `http://localhost:5173`
- **Backend API**: `http://localhost:3000/api`
- **Login Backend**: `http://localhost:5173/login-backend`
- **Vitrine Backend**: `http://localhost:5173/storefront-backend`

Credenciais de Teste

Pizza Express (slug: pizza-express)

- **Admin:** admin@pizza-express.com / pizza123
- **Cliente:** cliente@exemplo.com / cliente123

Burger King (slug: burger-king)

- **Admin:** admin@burger-king.com / pizza123
- **Cliente:** cliente@exemplo.com / cliente123

Fluxo de Uso

1. Login

1. Acesse <http://localhost:5173/login-backend>
2. Selecione a empresa (Pizza Express ou Burger King)
3. Digite email e senha
4. Clique em “Entrar”

2. Navegação na Vitrine

1. Após login, você será redirecionado para `/storefront-backend`
2. Veja a lista de produtos da empresa selecionada
3. Use a busca para filtrar produtos
4. Clique em “Adicionar ao Carrinho” em qualquer produto

3. Gerenciamento do Carrinho

1. Clique no ícone do carrinho no header
2. Veja os itens adicionados
3. Ajuste quantidades com os botões +/-
4. Remova itens com o botão de lixeira
5. Veja recomendações de produtos relacionados
6. Clique em “Finalizar Pedido”

4. Checkout

1. Preencha o endereço de entrega
2. Selecione a forma de pagamento
3. Adicione cupom de desconto (opcional)
4. Adicione observações (opcional)
5. Revise o resumo do pedido
6. Clique em “Confirmar Pedido”

5. Confirmação

1. Veja o número do pedido
2. Veja o valor total
3. Opções para voltar à loja ou ver pedidos

Endpoints Integrados

Autenticação

- `POST /api/auth/login` - Login com email, senha e empresa
- `POST /api/auth/refresh` - Refresh do token de acesso

Produtos

- `GET /api/produtos` - Listar produtos da empresa
- Query params: `categoria`, `disponivel`

Carrinho

- `GET /api/carrinho` - Obter carrinho do usuário
- `POST /api/carrinho/itens` - Adicionar item ao carrinho
- `PATCH /api/carrinho/itens/:id` - Atualizar quantidade
- `DELETE /api/carrinho/itens/:id` - Remover item
- `DELETE /api/carrinho` - Limpar carrinho
- `POST /api/carrinho/checkout` - Finalizar pedido
- `GET /api/carrinho/recomendacoes` - Obter recomendações

Componentes Principais

apiClient.ts

Cliente Axios configurado com:

- Base URL do backend
- Interceptor de request (adiciona token e tenant)
- Interceptor de response (refresh token automático)
- Tratamento de erros 401

backendApi.ts

Funções tipadas para todos os endpoints:

- Interfaces TypeScript para requests/responses
- Funções organizadas por domínio (auth, produtos, carrinho)
- Tratamento de erros consistente

CartContext.tsx

Context React para gerenciar estado do carrinho:

- Estado: `cart`, `loading`, `error`, `lastAddedItemId`
- Funções: `fetchCart`, `addItem`, `updateItem`, `removeItem`, `clearCart`
- Integração com API real
- Feedback com toasts

LoginBackend.tsx

Página de login integrada:

- Seleção de empresa (tenant)
- Formulário de email/senha
- Integração com API de autenticação
- Armazenamento de tokens
- Redirecionamento após login

VitrineBackend.tsx

Vitrine de produtos integrada:

- Listagem de produtos do backend
- Busca/filtro de produtos
- Adicionar ao carrinho
- Indicadores de estoque
- Loading states

CartDrawerBackend.tsx

Drawer lateral do carrinho:

- Lista de itens do carrinho
- Ajuste de quantidades
- Remoção de itens
- Recomendações de produtos
- Resumo de valores
- Botão de checkout

CheckoutBackend.tsx

Página de checkout:

- Formulário de endereço
- Seleção de pagamento
- Campo de cupom
- Observações
- Resumo do pedido
- Integração com API



Segurança

Tokens

- Access token armazenado em localStorage
- Refresh token armazenado em localStorage
- Tokens enviados via header Authorization
- Refresh automático em caso de 401

Multi-tenancy

- Tenant slug armazenado em localStorage
- Enviado em todas as requisições via header
- Validado no backend

Logout

- Limpeza de todos os tokens
- Redirecionamento para login
- Executado automaticamente em caso de erro de autenticação



Testes Manuais Realizados



Autenticação

- [x] Login com credenciais válidas

- [x] Login com credenciais inválidas
- [x] Seleção de diferentes empresas
- [x] Refresh token automático
- [x] Logout

✓ Produtos

- [x] Listagem de produtos
- [x] Busca de produtos
- [x] Produtos indisponíveis
- [x] Produtos com estoque baixo

✓ Carrinho

- [x] Adicionar item
- [x] Atualizar quantidade
- [x] Remover item
- [x] Limpar carrinho
- [x] Recomendações

✓ Checkout

- [x] Preenchimento de formulário
- [x] Validação de campos
- [x] Seleção de pagamento
- [x] Cupom de desconto
- [x] Finalização de pedido



Métricas

- **Arquivos criados:** 8
- **Arquivos modificados:** 3
- **Linhas de código:** ~1500
- **Endpoints integrados:** 10
- **Componentes criados:** 5
- **Contexts criados:** 1



Problemas Conhecidos

Nenhum problema conhecido no momento.



Próximos Passos

Melhorias Sugeridas

1. Adicionar testes unitários
2. Adicionar testes de integração
3. Implementar cache de produtos
4. Adicionar paginação na vitrine
5. Implementar filtros avançados
6. Adicionar histórico de pedidos

7. Implementar notificações em tempo real
8. Adicionar suporte a imagens múltiplas
9. Implementar sistema de avaliações
10. Adicionar analytics

Integrações Futuras

1. WhatsApp (N8N)
2. Gateway de pagamento (Asaas)
3. Sistema de notificações
4. Dashboard de métricas
5. Relatórios

Documentação Adicional

- [README-ANALISE.md](#) (./README-ANALISE.md) - Análise do projeto
- [FASE-2-PLANEJAMENTO.md](#) (./FASE-2-PLANEJAMENTO.md) - Planejamento da Fase 2
- [FASE-2-TESTES.md](#) (./FASE-2-TESTES.md) - Testes da Fase 2
- [RELATORIO-FASE-2.md](#) (./RELATORIO-FASE-2.md) - Relatório da Fase 2

Contribuindo

Para contribuir com o projeto:

1. Crie uma branch feature
2. Faça suas alterações
3. Teste localmente
4. Crie um Pull Request

Suporte

Para dúvidas ou problemas:

- Abra uma issue no GitHub
- Entre em contato com a equipe de desenvolvimento

Data de criação: 08/10/2025

Versão: 1.0.0

Status:  Concluído e Testado