Sistema de E-commerce Integrado com Serviço de Pagamentos (Mercado Pago)

Equipe:

Lucas Rodrigues da Silva – 2322657

1. Objetivo do Sistema

O objetivo deste projeto é desenvolver um sistema integrado composto por múltiplos serviços que se comunicam entre si por meio de APIs REST, utilizando tecnologias modernas de frontend, backend, persistência de dados e integração externa via um provedor de pagamento (Mercado Pago).

O sistema possibilita:

- Autenticação de usuários
- Gerenciamento de produtos (CRUD completo)
- Criação de pedidos
- Processamento de pagamentos reais usando a API do Mercado Pago
- Integração entre sistemas distintos (Frontend, Backend e Serviço de Pagamento)

2. Arquitetura do Sistema

A solução proposta é composta por quatro serviços principais, funcionando de forma integrada:

✓ 1. Frontend (React.js + Vite)

Responsável pela interface do usuário.

Funcionalidades:

- Login e registro
- Listagem de produtos
- Criação, edição e exclusão de produtos
- Carrinho de compras
- Tela de pagamento (checkout)
- Consumo das APIs do backend via Axios

✓ 2. Backend API (Node.js + Express)

Responsável pela lógica de negócio central.

Principais módulos:

- /api/auth – Autenticação (JWT + bcrypt)
- /api/products – CRUD completo de produtos
- /api/orders – Criação e gerenciamento de pedidos

- /api/payment/process – Integração com Mercado Pago
- Middleware de autenticação
- Comunicação com MongoDB

✓ 3. Serviço de Pagamento (Mercado Pago API)

Este é o segundo sistema obrigatório na AV3.

O backend se comunica com o Mercado Pago via:

```bash
POST https://api.mercadopago.com/v1/payments
```

Enviando:

- valor da compra
- descrição
- método de pagamento
- token de cartão (ou PIX)
- dados do cliente

E recebendo:

- status do pagamento (approved / rejected / in_process)
- ID da transação
- link de pagamento (QR Code ou Checkout Pro)
- confirmação do Mercado Pago

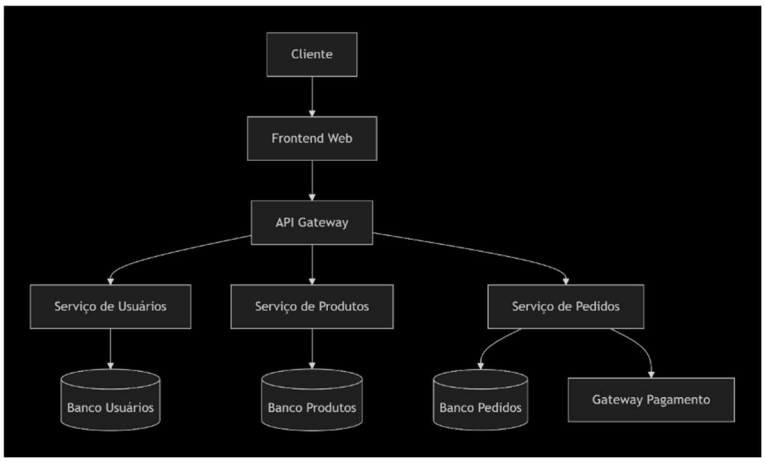
✓ 4. Banco de Dados – MongoDB Atlas

Responsável pela persistência dos dados.

Coleções:

- users (autenticação + perfil)
- products (CRUD completo)
- orders (pedidos e pagamentos)

3. Diagrama da Arquitetura



4. Autenticação e Segurança

- JWT Token armazenado no LocalStorage
- Middleware no backend para proteger rotas
- Proteção de criação/edição de produtos
- Controle de acesso baseado em usuário autenticado

5. Fluxos de Integração (importante para AV3)

- Fluxo 1 – Autenticação
- Usuário envia email + senha → */api/auth/login*
- Backend válida e gera JWT
- Frontend salva o token
- Todas as requisições seguintes são autenticadas

Fluxo 2 – CRUD de produtos

- Frontend chama */api/products*
- Backend consulta MongoDB
- Frontend exibe lista de produtos
- Edição e exclusão são protegidas com JWT

Fluxo 3 – Checkout + Mercado Pago

- Usuário adiciona produtos ao carrinho
- Front cria pedido → */api/orders*
- Backend chama Mercado Pago → */v1/payments*
- Mercado Pago retorna status da transação
- Backend salva o pedido
- Front mostra comprovante / status (approved, pending, rejected)

Este fluxo garante os dois sistemas integrados, que é o núcleo da AV3.

6. Tecnologias Utilizadas

Frontend:

- React.js
- Vite
- Axios
- CSS3

Backend:

- Node.js
- Express
- JWT + bcrypt
- Axios (para integrar Mercado Pago)

Banco:

- MongoDB Atlas
- Integração Externa:

- Mercado Pago API

Deploy:

- Frontend: Vercel
- Backend: Railway / Render
- Banco: MongoDB Atlas

7. Estrutura do Repositório GitHub

```bash
/frontend
   /src
      /components
      /pages
      /context
      /services
/backend
   /src
      /routes
      /controllers
      /models
      /middleware
/docs
   arquitetura.pdf
   diagrama.png
README.md
```

Próximos Passos (até a apresentação final após 25/11)

- ✓ Concluir CRUD (falta exclusão – DELETE)
- ✓ Finalizar integração Mercado Pago
- ✓ Criar página de checkout
- ✓ Implementar carrinho
- ✓ Fazer deploy
- ✓ Finalizar README.md
- ✓ Apresentação em sala