🛒 Market Online Backend

📌 Visão Geral

O **Market Online Backend** é uma API REST desenvolvida em **Java 21** com **Spring Boot**, seguindo boas práticas de arquitetura, design de APIs e organização de código.
Este projeto está sendo construído de forma **educacional e profissional**, com foco em **planejamento, escalabilidade e clareza**, simulando um backend real de um sistema de marketplace.

A branch `Planejamento Dev API Backend` representa a fase de **design e estruturação da aplicação**, antes da implementação completa das regras de negócio.

🕐 Objetivos do Projeto

- Criar uma API REST bem estruturada e documentada
- Aplicar conceitos de **Design First**
- Utilizar boas práticas de **Backend Java**
- Servir como portfólio e base de aprendizado
- Facilitar futuras evoluções (auth, pagamentos, pedidos, etc.)

Metodologia Utilizada

🔧 Design First

A API é planejada antes da implementação: - Definição de recursos - Estrutura de endpoints - Modelagem de entidades - DTOs claros e objetivos

🔧 Separação de Responsabilidades

Cada camada possui uma função bem definida: - Controller → Entrada e saída de dados - Service → Regras de negócio - Repository → Acesso ao banco - Entity → Persistência - DTO → Comunicação externa

---

🛠️ Tecnologias Utilizadas

- **Java 21**
- **Spring Boot**
- **Spring Web**
- **Spring Data JPA**
- **PostgreSQL**
- **Hibernate**
- **Maven**
- **Jakarta Validation**

📁 Estrutura do Projeto

```
market-online-backend
│
├── src/main/java/com/marketonline
│   ├── controller      # Camada REST (endpoints)
│   ├── service         # Regras de negócio
│   ├── repository      # Acesso ao banco de dados
│   ├── entity          # Entidades JPA
│   ├── dto             # Data Transfer Objects
│   ├── enums           # Enumerações do domínio
│   └── config          # Configurações gerais
│
├── src/main/resources
│   ├── application.yml # Configurações do Spring Boot
│   └── db              # Scripts futuros de banco (opcional)
│
├── pom.xml             # Dependências do projeto
└── README.md
```

🧰 Modelagem Inicial

📦 Entidades

- **Product**
- **Category** (Enum)

Exemplo de responsabilidades: - Entity → Representa a tabela no banco - Enum → Restringe valores válidos

---

📦 DTOs (Data Transfer Objects)

Os DTOs são utilizados para: - Evitar exposição direta das entidades - Garantir validações de entrada - Melhorar segurança e clareza da API

Exemplos: - `ProductRequestDTO` - `ProductResponseDTO`

🧪 Validações

Utilizamos **Jakarta Validation** para garantir integridade dos dados:

- `@NotNull`
- `@NotBlank`
- `@Size`

Essas validações atuam diretamente nos DTOs.

🗄️ Banco de Dados

- **PostgreSQL**
- Hibernate como ORM
- Dialeto configurado:

```
spring:
  jpa:
    database-platform: org.hibernate.dialect.PostgreSQLDialect
```

> O banco **não é criado automaticamente**. Ele deve existir previamente ou ser provisionado via serviço (Railway, Supabase, etc.).

🗜️ Status da Branch

📍 **Branch:** `Planejamento Dev API Backend`

🔧 Estrutura do projeto definida
🔧 Dependências configuradas
🔧 DTOs implementados
🔧 Enums criados
🔧 Entities em andamento
⏳ Controllers e Services serão implementados nas próximas etapas

---

🔮 Próximos Passos

- Implementar Controllers REST
- Criar Services com regras de negócio
- Implementar Repositories
- Tratamento global de exceções
- Documentação com Swagger/OpenAPI
- Autenticação (JWT)

💻 Autor

Mateus Amaral
Analista de Suporte com mais de 10 anos de experiência, em transição para Desenvolvimento Back-End.

📄 Licença

Este projeto é de uso educacional e livre para estudos e melhorias.