**Delivery FAT**

**Sistema de Pedidos de Delivery**

Sistema de pedidos desenvolvido como parte do curso da **Fundação FAT**.  
O objetivo é criar uma aplicação web para gerenciar pedidos e produtos de um serviço de entrega, promovendo o aprendizado de tecnologias modernas e boas práticas de desenvolvimento em acompanhamento das aulas práticas do professor Alexandre Erdmann no terceiro período do curso técnico de Desenvolvimento de Sistemas da Fundação FAT, turma 01.

**Equipe de Desenvolvimento**

* Carlos Henrique Garcia Soares
* Professor Alexandre Erdmann

**Escopo e Objetivos**

O sistema visa permitir o **cadastro de produtos**, a **realização e acompanhamento de pedidos**, além de fornecer uma **API documentada** para integração com outros sistemas.  
O projeto busca proporcionar **experiência prática** em desenvolvimento backend com **Java** e **Spring Boot**.

**Objetivos:**

* Facilitar o gerenciamento de pedidos de delivery
* Permitir o cadastro, consulta e atualização de produtos
* Garantir a validação dos dados e a documentação da API

**Requisitos**

**Requisitos Funcionais**

* Cadastro de produtos (nome, preço)
* Listagem de produtos
* Cadastro de pedidos (cliente, valor total, data/hora)
* Listagem de pedidos
* Validação dos dados de entrada
* Documentação da API via Swagger

**Requisitos Não Funcionais**

* Persistência dos dados em banco H2 em memória
* API RESTful
* Código limpo e organizado seguindo boas práticas
* Utilização de anotações para validação
* Documentação automática da API

**Tecnologias Utilizadas**

* Java 17
* Spring Boot 3.1.x
* Maven
* Spring Web
* Spring Data JPA
* MySQL Driver
* H2 Database (para testes locais)
* Lombok
* Validation (Jakarta Bean Validation)
* Swagger (springdoc-openapi)
* ModelMapper

**Descrição das Transações CRUD**

**Produto**

* **Create (POST /produtos)**  
  Cria um novo produto com nome e preço.  
  **Entrada:** JSON com nome, preco.  
  **Validações:** campos obrigatórios, preço positivo.
* **Read (GET /produtos)**  
  Lista todos os produtos cadastrados.  
  **Saída:** lista JSON com id, nome, preco.
* **Read by ID (GET /produtos/{id})**  
  Retorna os dados de um produto específico.  
  **Erro 404:** se o ID não existir.
* **Update (PUT /produtos/{id})**  
  Atualiza os dados de um produto existente.  
  **Entrada:** JSON com campos atualizados.  
  **Validações:** mesmas do create.
* **Delete (DELETE /produtos/{id})**  
  Remove um produto pelo ID.  
  **Erro 404:** se o produto não existir.

**Pedido**

* **Create (POST /pedidos)**  
  Cria um novo pedido com cliente, produtos e total.  
  **Entrada:** JSON com cliente, lista de produtosIds.  
  **Processamento:** calcula o valor total com base nos produtos.
* **Read (GET /pedidos)**  
  Lista todos os pedidos com resumo de informações.  
  **Inclui:** cliente, data, status, valor total.
* **Read by ID (GET /pedidos/{id})**  
  Retorna os dados completos de um pedido.  
  **Inclui:** produtos, cliente, data e status.
* **Update (PUT /pedidos/{id})**  
  Atualiza informações do pedido (ex: status).  
  **Exemplo:** alterar de "PENDENTE" para "ENTREGUE".
* **Delete (DELETE /pedidos/{id})**  
  Cancela um pedido existente.  
  **Regra de negócio:** não pode excluir se já estiver "ENTREGUE".

