### Exercício 15.5 – Gestão Avançada de Inventário com Funcionalidades Completas

O objetivo deste exercício é criar uma aplicação completa para gestão de inventário com funcionalidades avançadas, incluindo relatórios detalhados, validações no lado do servidor, estilos dinâmicos e histórico de movimentos.

### Parte 1: Configuração do Projeto no Spring Initializr

1. **Acessar o Spring Initializr:**
   * Vá ao site [Spring Initializr](https://start.spring.io/).
2. **Configurar o projeto:**
   * **Project:** Maven.
   * **Language:** Java.
   * **Spring Boot:** Última versão estável.
   * **Group:** com.example.
   * **Artifact:** advancedinventory.
   * **Name:** advancedinventory.
   * **Package Name:** com.example.advancedinventory.
   * **Packaging:** Jar.
   * **Java Version:** 21.
3. **Adicionar dependências:**
   * **Spring Web:** Para APIs REST.
   * **Spring Data JPA:** Para integração com a base de dados.
   * **H2 Database:** Para persistência em disco.
   * **Vaadin Spring Boot Starter:** Para a interface gráfica.
4. **Gerar o projeto:**
   * Clique em **Generate** para descarregar o projeto como um ficheiro ZIP.
5. **Abrir o projeto no VS Code:**
   * Extraia o ficheiro ZIP.
   * No VS Code, clique em **File > Open Folder** e selecione a pasta do projeto.

### Parte 2: Estrutura do Projeto

1. **Criar as pastas necessárias:**
   * Em src/main/java/com/example/advancedinventory, crie:
     + entity para as classes Item, Category e StockMovement.
     + repository para ItemRepository, CategoryRepository e StockMovementRepository.
     + service para ItemService, CategoryService e StockMovementService.
     + views para InventoryView.
2. Estrutura do projeto:

src/  
├── main/  
│ ├── java/com/example/advancedinventory/  
│ │ ├── AdvancedInventoryApplication.java  
│ │ ├── entity/  
│ │ │ ├── Item.java  
│ │ │ ├── Category.java  
│ │ │ └── StockMovement.java  
│ │ ├── repository/  
│ │ │ ├── ItemRepository.java  
│ │ │ ├── CategoryRepository.java  
│ │ │ └── StockMovementRepository.java  
│ │ ├── service/  
│ │ │ ├── ItemService.java  
│ │ │ ├── CategoryService.java  
│ │ │ └── StockMovementService.java  
│ │ └── views/  
│ │ └── InventoryView.java  
├── resources/  
│ ├── application.properties  
│ ├── data.sql  
│ └── META-INF/resources/frontend/styles/styles.css

### Parte 3: Funcionalidades Implementadas

**1. Relatórios Detalhados:**

* Relatórios de itens com baixo stock.
* Grelha dinâmica exibindo categorias e quantidades.

**2. Validações no Servidor:**

* Verificação de campos obrigatórios, quantidade negativa e categorias inexistentes.

**3. Estilos Dinâmicos:**

* Estilos CSS para destacar itens com baixo stock.

**4. Histórico de Movimentos:**

* Registro de entradas e saídas de itens no estoque.

**5. Testes:**

* Cobertura completa para serviços, repositórios e interface.

### Parte 4: Configuração no application.properties

spring.datasource.url=jdbc:h2:file:./data/inventorydb  
spring.datasource.driver-class-name=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
spring.jpa.hibernate.ddl-auto=update  
spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console  
server.port=8080  
vaadin.urlMapping=/\*

### Parte 5: Dados Iniciais no data.sql

-- Categorias  
INSERT INTO category (id, name) VALUES (1, 'Material de Escritório');  
INSERT INTO category (id, name) VALUES (2, 'Eletrónicos');  
INSERT INTO category (id, name) VALUES (3, 'Mobiliário');  
  
-- Itens  
INSERT INTO item (id, name, quantity, category\_id) VALUES (1, 'Caneta', 100, 1);  
INSERT INTO item (id, name, quantity, category\_id) VALUES (2, 'Lápis', 200, 1);  
INSERT INTO item (id, name, quantity, category\_id) VALUES (3, 'Monitor', 10, 2);  
INSERT INTO item (id, name, quantity, category\_id) VALUES (4, 'Cadeira', 5, 3);  
  
-- Movimentos de Estoque  
INSERT INTO stock\_movement (id, item\_id, quantity\_change, timestamp, type) VALUES (1, 1, 50, CURRENT\_TIMESTAMP, 'Entrada');  
INSERT INTO stock\_movement (id, item\_id, quantity\_change, timestamp, type) VALUES (2, 3, -2, CURRENT\_TIMESTAMP, 'Saída');  
INSERT INTO stock\_movement (id, item\_id, quantity\_change, timestamp, type) VALUES (3, 4, -1, CURRENT\_TIMESTAMP, 'Saída');