### Exercício 15.3 – Sistema de Gestão de Inventário com Spring Boot e Vaadin

Este exercício propõe a criação de uma aplicação para gerir inventários com um **back-end Spring Boot**, uma base de dados H2 com persistência em disco, e uma interface Vaadin interativa.

### 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:** inventory.
   * **Name:** inventory.
   * **Package Name:** com.example.inventory.
   * **Packaging:** Jar.
   * **Java Version:** 21.
3. **Adicionar dependências:**
   * **Spring Web:** Para criar APIs REST.
   * **Spring Data JPA:** Para integração com a base de dados.
   * **H2 Database:** Para persistência em disco.
   * **Vaadin Spring Boot Starter:** Para criar 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/inventory, crie:
     + entity para a classe Item.
     + repository para a interface ItemRepository.
     + service para a classe ItemService.
     + views para a classe InventoryView.
2. A estrutura deverá ser:

src/  
├── main/  
│ ├── java/com/example/inventory/  
│ │ ├── InventoryApplication.java  
│ │ ├── entity/  
│ │ │ └── Item.java  
│ │ ├── repository/  
│ │ │ └── ItemRepository.java  
│ │ ├── service/  
│ │ │ └── ItemService.java  
│ │ └── views/  
│ │ └── InventoryView.java  
├── resources/  
│ ├── application.properties  
│ └── data.sql

### Parte 3: Implementação do Sistema de Inventário

1. **Classe Principal (InventoryApplication):**

package com.example.inventory;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class InventoryApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(InventoryApplication.class, args);  
 }  
}

1. **Entidade Item:**

package com.example.inventory.entity;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class Item {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String name;  
 private int quantity;  
  
 public Item() {}  
  
 public Item(String name, int quantity) {  
 this.name = name;  
 this.quantity = quantity;  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getQuantity() {  
 return quantity;  
 }  
  
 public void setQuantity(int quantity) {  
 this.quantity = quantity;  
 }  
}

1. **Repositório ItemRepository:**

package com.example.inventory.repository;  
  
import com.example.inventory.entity.Item;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
import java.util.List;  
  
@Repository  
public interface ItemRepository extends JpaRepository<Item, Long> {  
 List<Item> findByNameContainingIgnoreCase(String name);  
}

1. **Serviço ItemService:**

package com.example.inventory.service;  
  
import com.example.inventory.entity.Item;  
import com.example.inventory.repository.ItemRepository;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class ItemService {  
  
 private final ItemRepository itemRepository;  
  
 public ItemService(ItemRepository itemRepository) {  
 this.itemRepository = itemRepository;  
 }  
  
 public List<Item> getAllItems() {  
 return itemRepository.findAll();  
 }  
  
 public List<Item> filterItemsByName(String nameFilter) {  
 return itemRepository.findByNameContainingIgnoreCase(nameFilter);  
 }  
  
 public Item saveItem(Item item) {  
 return itemRepository.save(item);  
 }  
  
 public void deleteItem(Long id) {  
 itemRepository.deleteById(id);  
 }  
}

1. **Vista InventoryView:**

package com.example.inventory.views;  
  
import com.example.inventory.entity.Item;  
import com.example.inventory.service.ItemService;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.component.grid.Grid;  
import com.vaadin.flow.component.notification.Notification;  
import com.vaadin.flow.component.orderedlayout.HorizontalLayout;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.Route;  
  
import java.util.List;  
  
@Route("")  
public class InventoryView extends VerticalLayout {  
  
 private final ItemService itemService;  
 private final Grid<Item> itemGrid = new Grid<>(Item.class, false);  
 private final TextField nameFilter = new TextField("Filtrar por Nome");  
 private final TextField itemNameField = new TextField("Nome do Item");  
 private final TextField itemQuantityField = new TextField("Quantidade");  
 private final Button addButton = new Button("Adicionar");  
 private final Button clearButton = new Button("Limpar");  
  
 public InventoryView(ItemService itemService) {  
 this.itemService = itemService;  
  
 configureGrid();  
 configureForm();  
  
 HorizontalLayout filterLayout = new HorizontalLayout(nameFilter);  
 add(filterLayout, itemGrid, new HorizontalLayout(itemNameField, itemQuantityField, addButton, clearButton));  
  
 updateGrid();  
 }  
  
 private void configureGrid() {  
 itemGrid.addColumn(Item::getName).setHeader("Nome");  
 itemGrid.addColumn(Item::getQuantity).setHeader("Quantidade");  
 itemGrid.addComponentColumn(item -> {  
 Button deleteButton = new Button("Eliminar", click -> {  
 itemService.deleteItem(item.getId());  
 updateGrid();  
 Notification.show("Item eliminado!");  
 });  
 return deleteButton;  
 });  
 itemGrid.setItems(itemService.getAllItems());  
 }  
  
 private void configureForm() {  
 addButton.addClickListener(click -> {  
 String name = itemNameField.getValue();  
 String quantityStr = itemQuantityField.getValue();  
  
 if (name.isEmpty() || quantityStr.isEmpty()) {  
 Notification.show("Todos os campos são obrigatórios!");  
 return;  
 }  
  
 int quantity = Integer.parseInt(quantityStr);  
 Item newItem = new Item(name, quantity);  
 itemService.saveItem(newItem);  
  
 updateGrid();  
 clearForm();  
 Notification.show("Item adicionado!");  
 });  
  
 clearButton.addClickListener(click -> clearForm());  
 }  
  
 private void clearForm() {  
 itemNameField.clear();  
 itemQuantityField.clear();  
 }  
  
 private void updateGrid() {  
 String filter = nameFilter.getValue();  
 List<Item> filteredItems = filter.isEmpty() ? itemService.getAllItems() : itemService.filterItemsByName(filter);  
 itemGrid.setItems(filteredItems);  
 }  
}

### 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

Se precisar da pasta src compactada, avise! 😊