### Exercício 15.5 – 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. Aceder ao Spring Initializr:
   * Vá ao site Spring Initializr.
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.
   * Spring Boot Starter Security: Para implementar autenticação e autorização.
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 as classes de visualização (InventoryView e LoginView).
     + config para a configuração de segurança (SecurityConfig).
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  
│ │ │ └── LoginView.java  
│ │ └── config/  
│ │ └── SecurityConfig.java  
├── resources/  
│ ├── application.properties  
│ └── data.sql

#### Parte 3: Implementação do Sistema de Inventário

**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);  
 }  
}

**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;  
 }  
}

**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);  
}

**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);  
 }  
}

**Configuração de Segurança (SecurityConfig):**

package com.example.inventory.config;  
  
import com.vaadin.flow.spring.security.VaadinWebSecurity;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
  
@Configuration  
public class SecurityConfig extends VaadinWebSecurity {  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 // Permitir acesso à página de login sem autenticação  
 super.configure(http);  
 setLoginView(http, LoginView.class);  
 }  
}

**Vista de Login (LoginView):**

package com.example.inventory.views;  
  
import com.vaadin.flow.component.html.H1;  
import com.vaadin.flow.component.html.Label;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.component.textfield.PasswordField;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.component.button.Button;  
import com.vaadin.flow.router.Route;  
  
@Route("login")  
public class LoginView extends VerticalLayout {  
  
 public LoginView() {  
 TextField usernameField = new TextField("Username");  
 PasswordField passwordField = new PasswordField("Password");  
 Button loginButton = new Button("Login", e -> {  
 getUI().ifPresent(ui -> ui.getPage().setLocation("")); // Redirecionar após login  
 });  
  
 add(new H1("Login"), usernameField, passwordField, loginButton, new Label("Introduza as suas credenciais."));  
 }  
}

**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 com.vaadin.flow.data.value.ValueChangeMode;  
import jakarta.annotation.security.PermitAll;  
  
import java.util.List;  
import java.util.stream.Collectors;  
  
@PermitAll  
@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");  
  
 private Item selectedItem = null;  
  
 public InventoryView(ItemService itemService) {  
 this.itemService = itemService;  
  
 configureGrid();  
 configureForm();  
 configureFilter();  
  
 HorizontalLayout filterLayout = new HorizontalLayout(nameFilter);  
 HorizontalLayout buttonLayout = new HorizontalLayout(addButton, clearButton);  
 buttonLayout.setAlignItems(Alignment.END);  
  
 add(filterLayout, itemGrid, new HorizontalLayout(itemNameField, itemQuantityField), buttonLayout);  
  
 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.asSingleSelect().addValueChangeListener(event -> {  
 selectedItem = event.getValue();  
 if (selectedItem != null) {  
 itemNameField.setValue(selectedItem.getName());  
 itemQuantityField.setValue(String.valueOf(selectedItem.getQuantity()));  
 addButton.setText("Atualizar");  
 } else {  
 clearForm();  
 }  
 });  
  
 itemGrid.setItems(itemService.getAllItems());  
 }  
  
 private void configureForm() {  
 addButton.addClickListener(click -> {  
 if (selectedItem == null) {  
 Item newItem = new Item();  
 newItem.setName(itemNameField.getValue());  
 newItem.setQuantity(Integer.parseInt(itemQuantityField.getValue()));  
 itemService.saveItem(newItem);  
 Notification.show("Item adicionado!");  
 } else {  
 selectedItem.setName(itemNameField.getValue());  
 selectedItem.setQuantity(Integer.parseInt(itemQuantityField.getValue()));  
 itemService.saveItem(selectedItem);  
 Notification.show("Item atualizado!");  
 }  
 updateGrid();  
 clearForm();  
 });  
  
 clearButton.addClickListener(click -> clearForm());  
 }  
  
 private void configureFilter() {  
 nameFilter.setValueChangeMode(ValueChangeMode.LAZY);  
 nameFilter.setValueChangeTimeout(300);  
  
 nameFilter.addValueChangeListener(event -> {  
 String filterText = event.getValue().toLowerCase();  
 applyFilter(filterText);  
 });  
 }  
  
 private void applyFilter(String filterText) {  
 List<Item> filteredItems = itemService.getAllItems().stream()  
 .filter(item -> matchesFilter(item.getName(), filterText))  
 .collect(Collectors.toList());  
 itemGrid.setItems(filteredItems);  
 }  
  
 private boolean matchesFilter(String fullName, String searchText) {  
 String normalizedFullName = fullName.toLowerCase();  
 String[] searchWords = searchText.toLowerCase().split(" ");  
 return List.of(searchWords).stream().allMatch(word -> normalizedFullName.contains(word));  
 }  
  
 private void updateGrid() {  
 itemGrid.setItems(itemService.getAllItems());  
 }  
  
 private void clearForm() {  
 selectedItem = null;  
 itemNameField.clear();  
 itemQuantityField.clear();  
 addButton.setText("Adicionar");  
 }  
}

**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  
  
spring.security.user.name=user  
spring.security.user.password=password

#### Parte 4: Testar a Aplicação

1. Inicie a aplicação.
2. Navegue para a página inicial (/).
3. Será redirecionado para a página de login.
4. Use as credenciais padrão:
   * **Username:** user
   * **Password:** password
5. Após autenticação, aceda ao inventário.