# Resolução do exercício 15.6, completa, passo-a-passo

## Introdução

Neste tutorial, vamos criar um projeto Spring Boot com Vaadin, adequado para iniciantes que já tenham o ambiente configurado. O projeto demonstrará como criar uma aplicação simples de inventário com funcionalidades de gestão de livros, incluindo criação, leitura, atualização e eliminação de registos.

O tutorial segue os seguintes passos:

1. Criação do projeto no Spring Initializr.
2. Importação no Visual Studio Code.
3. Configuração da estrutura do projeto e dependências.
4. Copiar e configurar os ficheiros fornecidos.
5. Executar e testar o projeto.

## 1. Criar o Projeto no Spring Initializr

Acesse o [Spring Initializr](https://start.spring.io/) e configure o projeto com os seguintes detalhes:

* **Project**: Maven
* **Language**: Java
* **Spring Boot Version**: 3.4.1 (ou superior)
* **Group**: com.example
* **Artifact**: demo
* **Name**: demo
* **Description**: Demo project for Spring Boot
* **Package name**: com.example.demo
* **Packaging**: Jar
* **Java**: 21

### Dependências Necessárias

Adicione as seguintes dependências:

* Spring Web
* Spring Data JPA
* Spring Security
* H2 Database
* Vaadin

Clique em **Generate** para baixar o projeto gerado.

## 2. Importar o Projeto no Visual Studio Code

1. Abra o Visual Studio Code.
2. Certifique-se de ter o **Extension Pack for Java** instalado.
3. Extraia o ficheiro ZIP do Spring Initializr para uma pasta no seu computador.
4. No Visual Studio Code, selecione **File > Open Folder** e abra a pasta do projeto.
5. Confirme que a estrutura do projeto coincide com a seguinte:

src/  
 main/  
 java/  
 com/  
 example/  
 demo/  
 resources/  
 application.properties

## 

## 3. Configuração da Estrutura do Projeto

Para adaptar a estrutura ao tutorial, reestruture o projeto conforme indicado abaixo. Crie manualmente as pastas necessárias se elas não existirem:

src/  
 main/  
 frontend/  
 java/  
 com/  
 example/  
 inventory/  
 InventoryApplication.java  
 config/  
 SecurityConfig.java  
 entity/  
 Livro.java  
 repository/  
 LivroRepository.java  
 service/  
 LivroService.java  
 views/  
 InventoryView.java  
 LoginView.java  
 resources/  
 application.properties

## 4. Copiar e Configurar os Ficheiros Fornecidos

### Substituir os Ficheiros no Projeto

1. Copie os ficheiros fornecidos para os locais corretos conforme a estrutura de diretórios apresentada.
2. Confirme que o código nos ficheiros coincide com o fornecido neste tutorial.

### 

### Código dos Ficheiros Fornecidos

#### 

#### InventoryApplication.java

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);  
 }  
}

#### 

#### SecurityConfig.java

package com.example.inventory.config;  
  
import com.example.inventory.views.LoginView;  
import com.vaadin.flow.spring.security.VaadinWebSecurity;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
  
@EnableWebSecurity  
@Configuration  
public class SecurityConfig extends VaadinWebSecurity {  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http.authorizeHttpRequests()  
 .requestMatchers("/h2-console/\*\*").permitAll();  
   
 // Configuração do H2 Console  
 http.csrf().disable();  
 http.headers().frameOptions().disable();  
   
 // Configuração do Vaadin  
 super.configure(http);  
 setLoginView(http, LoginView.class);  
 }  
}

#### 

#### Livro.java

package com.example.inventory.entity;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class Livro {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Long id;  
 private String referencia;  
 private double preco;  
 private String descricao;  
 private int quantidade;  
  
 public Livro() {}  
  
 public Livro(String referencia, double preco, String descricao, int quantidade) {  
 this.referencia = referencia;  
 this.preco = preco;  
 this.descricao = descricao;  
 this.quantidade = quantidade;  
 }  
  
 // Getters e Setters  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getReferencia() {  
 return referencia;  
 }  
  
 public void setReferencia(String referencia) {  
 this.referencia = referencia;  
 }  
  
 public double getPreco() {  
 return preco;  
 }  
  
 public void setPreco(double preco) {  
 this.preco = preco;  
 }  
  
 public String getDescricao() {  
 return descricao;  
 }  
  
 public void setDescricao(String descricao) {  
 this.descricao = descricao;  
 }  
  
 public int getQuantidade() {  
 return quantidade;  
 }  
  
 public void setQuantidade(int quantidade) {  
 this.quantidade = quantidade;  
 }  
}

#### 

#### LivroRepository.java

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

#### 

#### LivroService.java

package com.example.inventory.service;  
  
import com.example.inventory.entity.Livro;  
import com.example.inventory.repository.LivroRepository;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class LivroService {  
  
 private final LivroRepository livroRepository;  
  
 public LivroService(LivroRepository livroRepository) {  
 this.livroRepository = livroRepository;  
 }  
  
 public List<Livro> getAllLivros() {  
 return livroRepository.findAll();  
 }  
  
 public List<Livro> filterLivrosByReferencia(String referenciaFilter) {  
 return livroRepository.findByReferenciaContainingIgnoreCase(referenciaFilter);  
 }  
  
 public Livro saveLivro(Livro livro) {  
 return livroRepository.save(livro);  
 }  
  
 public void deleteLivro(Long id) {  
 livroRepository.deleteById(id);  
 }  
}

#### 

#### InventoryView.java

package com.example.inventory.views;  
  
import com.example.inventory.entity.Livro;  
import com.example.inventory.service.LivroService;  
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.component.textfield.TextArea;  
import com.vaadin.flow.component.textfield.NumberField;  
import com.vaadin.flow.component.textfield.IntegerField;  
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 LivroService livroService;  
 private final Grid<Livro> livroGrid = new Grid<>(Livro.class, false);  
 private final TextField referenciaFilter = new TextField("Filtrar por Referência");  
 private final TextField referenciaField = new TextField("Referência");  
 private final NumberField precoField = new NumberField("Preço");  
 private final TextArea descricaoField = new TextArea("Descrição");  
 private final IntegerField quantidadeField = new IntegerField("Quantidade");  
 private final Button addButton = new Button("Adicionar");  
 private final Button clearButton = new Button("Limpar");  
  
 private Livro selectedLivro = null;  
  
 public InventoryView(LivroService livroService) {  
 this.livroService = livroService;  
  
 configureGrid();  
 configureForm();  
 configureFilter();  
  
 HorizontalLayout filterLayout = new HorizontalLayout(referenciaFilter);  
 HorizontalLayout formLayout = new HorizontalLayout(referenciaField, precoField, descricaoField, quantidadeField);  
 HorizontalLayout buttonLayout = new HorizontalLayout(addButton, clearButton);  
 buttonLayout.setAlignItems(Alignment.END);  
  
 add(filterLayout, livroGrid, formLayout, buttonLayout);  
  
 updateGrid();  
 }  
  
 private void configureGrid() {  
 livroGrid.addColumn(Livro::getReferencia).setHeader("Referência");  
 livroGrid.addColumn(Livro::getPreco).setHeader("Preço");  
 livroGrid.addColumn(Livro::getDescricao).setHeader("Descrição");  
 livroGrid.addColumn(Livro::getQuantidade).setHeader("Quantidade");  
 livroGrid.addComponentColumn(livro -> {  
 Button deleteButton = new Button("Eliminar", click -> {  
 livroService.deleteLivro(livro.getId());  
 updateGrid();  
 Notification.show("Livro eliminado!");  
 });  
 return deleteButton;  
 });  
  
 livroGrid.asSingleSelect().addValueChangeListener(event -> {  
 selectedLivro = event.getValue();  
 if (selectedLivro != null) {  
 referenciaField.setValue(selectedLivro.getReferencia());  
 precoField.setValue(selectedLivro.getPreco());  
 descricaoField.setValue(selectedLivro.getDescricao());  
 quantidadeField.setValue(selectedLivro.getQuantidade());  
 addButton.setText("Atualizar");  
 } else {  
 clearForm();  
 }  
 });  
  
 livroGrid.setItems(livroService.getAllLivros());  
 }  
  
 private void configureForm() {  
 addButton.addClickListener(click -> {  
 if (selectedLivro == null) {  
 Livro newLivro = new Livro();  
 newLivro.setReferencia(referenciaField.getValue());  
 newLivro.setPreco(precoField.getValue());  
 newLivro.setDescricao(descricaoField.getValue());  
 newLivro.setQuantidade(quantidadeField.getValue());  
 livroService.saveLivro(newLivro);  
 Notification.show("Livro adicionado!");  
 } else {  
 selectedLivro.setReferencia(referenciaField.getValue());  
 selectedLivro.setPreco(precoField.getValue());  
 selectedLivro.setDescricao(descricaoField.getValue());  
 selectedLivro.setQuantidade(quantidadeField.getValue());  
 livroService.saveLivro(selectedLivro);  
 Notification.show("Livro atualizado!");  
 }  
 updateGrid();  
 clearForm();  
 });  
  
 clearButton.addClickListener(click -> clearForm());  
 }  
  
 private void configureFilter() {  
 referenciaFilter.setValueChangeMode(ValueChangeMode.LAZY);  
 referenciaFilter.setValueChangeTimeout(300);  
  
 referenciaFilter.addValueChangeListener(event -> {  
 String filterText = event.getValue().toLowerCase();  
 applyFilter(filterText);  
 });  
 }  
  
 private void applyFilter(String filterText) {  
 List<Livro> filteredLivros = livroService.getAllLivros().stream()  
 .filter(livro -> matchesFilter(livro.getReferencia(), filterText))  
 .collect(Collectors.toList());  
 livroGrid.setItems(filteredLivros);  
 }  
  
 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() {  
 livroGrid.setItems(livroService.getAllLivros());  
 }  
  
 private void clearForm() {  
 selectedLivro = null;  
 referenciaField.clear();  
 precoField.clear();  
 descricaoField.clear();  
 quantidadeField.clear();  
 addButton.setText("Adicionar");  
 }  
}

#### 

#### LoginView.java

package com.example.inventory.views;  
  
import com.vaadin.flow.component.html.H1;  
import com.vaadin.flow.component.login.LoginForm;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.router.BeforeEnterEvent;  
import com.vaadin.flow.router.BeforeEnterObserver;  
import com.vaadin.flow.router.PageTitle;  
import com.vaadin.flow.router.Route;  
import com.vaadin.flow.server.auth.AnonymousAllowed;  
  
@Route("login")  
@PageTitle("Login")  
@AnonymousAllowed  
public class LoginView extends VerticalLayout implements BeforeEnterObserver {  
 private final LoginForm loginForm = new LoginForm();  
  
 public LoginView() {  
 addClassName("login-view");  
 setSizeFull();  
 setAlignItems(Alignment.CENTER);  
 setJustifyContentMode(JustifyContentMode.CENTER);  
  
 loginForm.setAction("login");  
 loginForm.setForgotPasswordButtonVisible(false);  
  
 add(  
 new H1("Login"),  
 loginForm  
 );  
 }  
  
 @Override  
 public void beforeEnter(BeforeEnterEvent beforeEnterEvent) {  
 if(beforeEnterEvent.getLocation()  
 .getQueryParameters()  
 .getParameters()  
 .containsKey("error")) {  
 loginForm.setError(true);  
 }  
 }  
}

#### 

#### 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  
  
spring.jmx.enabled=false

### 

### Verificação do ficheiro pom.xml

Certifique-se de que as dependências no ficheiro pom.xml incluem as seguintes entradas:

<dependency>  
 <groupId>com.vaadin</groupId>  
 <artifactId>vaadin-spring-boot-starter</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>  
<dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
</dependency>

## 

## 5. Executar e Testar o Projeto

### 

### Passos para Executar

1. No Visual Studio Code, abra o terminal integrado (**Terminal > New Terminal**).
2. Execute o comando:

* ./mvnw spring-boot:run

1. Aguarde até que a aplicação inicie. O console deve indicar que o servidor está ativo em http://localhost:8080.

### Testar no Navegador

1. Aceda, no browser, a http://localhost:8080 para abrir a interface de inventário.
2. Utilize as funcionalidades de adicionar, filtrar, atualizar e eliminar livros para testar o sistema.
3. Aceda ao H2 Console em http://localhost:8080/h2-console com as credenciais:
   * **JDBC URL**: jdbc:h2:file:./data/inventorydb
   * **User Name**: sa
   * **Password**: *(deixe vazio)*

## Para testar o login, acesse http://localhost:8080/login com as credenciais:

* **User**: user
* **Password**: password

### Resultado Esperado

* A aplicação deve exibir uma interface gráfica gerida pelo Vaadin, onde é possível visualizar, adicionar, editar e remover livros do inventário.
* O acesso ao sistema deve estar protegido por autenticação.
* O H2 Console deve permitir inspecionar e manipular os dados diretamente na base de dados.

## Conclusão

Neste tutorial, explorámos a criação de uma aplicação Spring Boot integrada com Vaadin para criar uma interface de gestão de inventário. A partir deste ponto, pode expandir o projeto, adicionando novas funcionalidades, como relatórios, autenticação avançada, ou integração com APIs externas.