### **Instruções Passo-a-Passo para Configurar e Executar o Projeto Completo**

Este guia fornece todos os passos necessários para configurar e executar o **Inventário com Vaadin Security** no seu computador.

### **1. Criar o Projeto com Spring Initializr**

#### 1.1. Configurar o Projeto

1. Acesse [Spring Initializr](https://start.spring.io/).
2. Configure o projeto com as seguintes opções:
   * **Project:** Maven.
   * **Language:** Java.
   * **Spring Boot:** Última versão estável.
   * **Group:** com.example.
   * **Artifact:** inventario.
   * **Name:** inventario.
   * **Dependencies:**
     + Spring Web
     + Spring Data JPA
     + Spring Security
     + H2 Database
     + Vaadin Spring Boot Starter
     + Vaadin Spring Security.
3. Clique em **Generate** para descarregar o projeto.

#### 1.2. Abrir o Projeto no VS Code

1. Extraia o ficheiro ZIP gerado pelo Spring Initializr.
2. No **VS Code**, clique em **File > Open Folder** e selecione a pasta do projeto.

### **2. Configurar o Backend**

#### 2.1. Configurar a Base de Dados

No ficheiro src/main/resources/application.properties, adicione as seguintes configurações:

spring.datasource.url=jdbc:h2:file:./data/inventariodb  
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

### **3. Implementar o Código Backend**

#### 3.1. Criar a Entidade Livro

Crie a classe Livro em com.example.inventario.model:

package com.example.inventario.model;  
  
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 String titulo;  
 private String descricao;  
 private double preco;  
 private int quantidade;  
  
 public Livro() {}  
  
 public Livro(String referencia, String titulo, String descricao, double preco, int quantidade) {  
 this.referencia = referencia;  
 this.titulo = titulo;  
 this.descricao = descricao;  
 this.preco = preco;  
 this.quantidade = quantidade;  
 }  
  
 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 String getTitulo() { return titulo; }  
 public void setTitulo(String titulo) { this.titulo = titulo; }  
 public String getDescricao() { return descricao; }  
 public void setDescricao(String descricao) { this.descricao = descricao; }  
 public double getPreco() { return preco; }  
 public void setPreco(double preco) { this.preco = preco; }  
 public int getQuantidade() { return quantidade; }  
 public void setQuantidade(int quantidade) { this.quantidade = quantidade; }  
}

#### 3.2. Criar o Repositório LivroRepository

Crie a interface LivroRepository em com.example.inventario.repository:

package com.example.inventario.repository;  
  
import com.example.inventario.model.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> findByReferenciaContainingIgnoreCaseOrTituloContainingIgnoreCase(String referencia, String titulo);  
}

#### 3.3. Criar o Serviço LivroService

Implemente a classe LivroService em com.example.inventario.service:

package com.example.inventario.service;  
  
import com.example.inventario.model.Livro;  
import com.example.inventario.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> listarTodos() {  
 return livroRepository.findAll();  
 }  
  
 public Livro salvarOuAtualizar(Livro livro) {  
 return livroRepository.save(livro);  
 }  
  
 public void remover(Long id) {  
 livroRepository.deleteById(id);  
 }  
  
 public List<Livro> filtrarLivros(String filtro) {  
 return livroRepository.findByReferenciaContainingIgnoreCaseOrTituloContainingIgnoreCase(filtro, filtro);  
 }  
}

#### 3.4. Configurar Segurança com Vaadin Security

Implemente a classe SecurityConfig em com.example.inventario.security:

package com.example.inventario.security;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.web.SecurityFilterChain;  
  
@Configuration  
public class SecurityConfig {  
  
 @Bean  
 public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  
 http  
 .authorizeRequests()  
 .antMatchers("/login").permitAll()  
 .anyRequest().authenticated()  
 .and()  
 .formLogin()  
 .loginPage("/login")  
 .and()  
 .logout()  
 .logoutSuccessUrl("/login?logout")  
 .and()  
 .csrf().disable();  
 return http.build();  
 }  
  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
}

### **4. Criar o Frontend com Vaadin**

#### 4.1. Criar a Classe LoginView

Implemente a página de login:

package com.example.inventario.views;  
  
import com.vaadin.flow.component.login.LoginForm;  
import com.vaadin.flow.component.orderedlayout.VerticalLayout;  
import com.vaadin.flow.router.Route;  
import jakarta.annotation.security.PermitAll;  
  
@PermitAll  
@Route("login")  
public class LoginView extends VerticalLayout {  
  
 public LoginView() {  
 LoginForm loginForm = new LoginForm();  
 loginForm.setAction("login");  
 add(loginForm);  
 }  
}

#### 4.2. Criar a Classe InventoryView

Implemente a vista principal do inventário:

package com.example.inventario.views;  
  
import com.example.inventario.model.Livro;  
import com.example.inventario.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.VerticalLayout;  
import com.vaadin.flow.component.textfield.TextField;  
import com.vaadin.flow.router.Route;  
import jakarta.annotation.security.RolesAllowed;  
  
import java.util.List;  
  
@RolesAllowed("USER")  
@Route("")  
public class InventoryView extends VerticalLayout {  
  
 private final LivroService livroService;  
 private final Grid<Livro> grid = new Grid<>(Livro.class);  
 private final TextField filtro = new TextField("Filtrar por Referência ou Título");  
  
 public InventoryView(LivroService livroService) {  
 this.livroService = livroService;  
  
 configurarGrid();  
 configurarFiltro();  
  
 add(filtro, grid);  
 atualizarGrid();  
 }  
  
 private void configurarGrid() {  
 grid.addColumn(Livro::getReferencia).setHeader("Referência");  
 grid.addColumn(Livro::getTitulo).setHeader("Título");  
 grid.addColumn(Livro::getDescricao).setHeader("Descrição");  
 grid.addColumn(Livro::getPreco).setHeader("Preço (€)");  
 grid.addColumn(Livro::getQuantidade).setHeader("Quantidade");  
 grid.addComponentColumn(livro -> {  
 Button remover = new Button("Remover", click -> {  
 livroService.remover(livro.getId());  
 atualizarGrid();  
 Notification.show("Livro removido.");  
 });  
 return remover;  
 });  
 }  
  
 private void configurarFiltro() {  
 filtro.setPlaceholder("Digite uma referência ou título...");  
 filtro.addValueChangeListener(event -> atualizarGrid());  
 }  
  
 private void atualizarGrid() {  
 String filtroTexto = filtro.getValue();  
 List<Livro> livros = filtroTexto == null || filtroTexto.isEmpty()  
 ? livroService.listarTodos()  
 : livroService.filtrarLivros(filtroTexto);  
 grid.setItems(livros);  
 }  
}

### **5. Executar o Projeto**

1. No terminal, execute:

* mvn spring-boot:run

1. Aceda ao sistema em <http://localhost:8080>.
2. Credenciais:
   * **Admin:** admin/admin
   * **User:** user/password.

Se precisar de mais detalhes, avise! 😊