### Solução Passo-a-Passo para o Exercício 13.x – Segurança em Spring Boot e Vaadin

Este exercício foca-se na implementação de segurança em aplicações Spring Boot, incluindo autenticação básica, proteção de endpoints por roles, integração com LDAP, e segurança em vistas Vaadin.

### 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:** seguranca.
   * **Name:** seguranca.
   * **Package Name:** com.example.seguranca.
   * **Packaging:** Jar.
   * **Java Version:** 21.
3. **Adicionar dependências:**
   * **Spring Web:** Para criar APIs REST.
   * **Spring Boot Starter Security:** Para integrar segurança.
   * **Spring Data LDAP:** Para integração com LDAP.
   * **Spring Security LDAP:** Para autenticação LDAP.
   * **Vaadin Spring Boot Starter:** Para suporte ao Vaadin (se necessário).
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: Configuração Básica de Segurança

1. **Adicionar a dependência no pom.xml:**

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-security</artifactId>  
</dependency>

1. **Criar a classe SecurityConfig em src/main/java/com/example/seguranca/security:**

package com.example.seguranca.security;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.crypto.password.NoOpPasswordEncoder;  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
  
 @Override  
 protected void configure(HttpSecurity http) throws Exception {  
 http  
 .authorizeRequests()  
 .anyRequest().authenticated()  
 .and()  
 .formLogin()  
 .permitAll()  
 .and()  
 .logout()  
 .permitAll();  
 }  
  
 @Override  
 protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth.inMemoryAuthentication()  
 .passwordEncoder(NoOpPasswordEncoder.getInstance())  
 .withUser("user").password("password").roles("USER")  
 .and()  
 .withUser("admin").password("admin").roles("ADMIN");  
 }  
}

### Parte 3: Proteção de Rotas com Roles

1. **Atualizar o método configure em SecurityConfig:**

@Override  
protected void configure(HttpSecurity http) throws Exception {  
 http  
 .authorizeRequests()  
 .antMatchers("/admin/\*\*").hasRole("ADMIN")  
 .antMatchers("/user/\*\*").hasRole("USER")  
 .anyRequest().authenticated()  
 .and()  
 .formLogin()  
 .permitAll()  
 .and()  
 .logout()  
 .permitAll();  
}

1. **Criar o controlador MainController em src/main/java/com/example/seguranca/controller:**

package com.example.seguranca.controller;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
@RequestMapping  
public class MainController {  
  
 @GetMapping("/admin")  
 public String adminEndpoint() {  
 return "Acesso para Administradores!";  
 }  
  
 @GetMapping("/user")  
 public String userEndpoint() {  
 return "Acesso para Utilizadores!";  
 }  
}

### Parte 4: Integração com LDAP

1. **Adicionar dependências no pom.xml:**

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-ldap</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-ldap</artifactId>  
</dependency>  
<dependency>  
 <groupId>com.unboundid</groupId>  
 <artifactId>unboundid-ldapsdk</artifactId>  
</dependency>

1. **Configurar o application.properties:**

spring.ldap.embedded.base-dn=dc=example,dc=com  
spring.ldap.embedded.ldif=classpath:users.ldif  
spring.ldap.embedded.port=8389

1. **Criar o ficheiro users.ldif em src/main/resources:**

dn: dc=example,dc=com  
objectclass: top  
objectclass: domain  
dc: example  
  
dn: ou=users,dc=example,dc=com  
objectclass: top  
objectclass: organizationalUnit  
ou: users  
  
dn: uid=user,ou=users,dc=example,dc=com  
objectclass: top  
objectclass: person  
objectclass: organizationalPerson  
objectclass: inetOrgPerson  
cn: User  
sn: User  
uid: user  
userPassword: password

1. **Atualizar SecurityConfig para autenticação LDAP:**

@Override  
protected void configure(AuthenticationManagerBuilder auth) throws Exception {  
 auth  
 .ldapAuthentication()  
 .userDnPatterns("uid={0},ou=users")  
 .contextSource()  
 .url("ldap://localhost:8389/dc=example,dc=com")  
 .and()  
 .passwordCompare()  
 .passwordEncoder(new BCryptPasswordEncoder())  
 .passwordAttribute("userPassword");  
}

### Parte 5: Testar o Projeto

1. **Executar a aplicação:**
   * Use o comando mvn spring-boot:run.
2. **Testar endpoints protegidos:**
   * Acesse /admin com as credenciais de admin/admin.
   * Acesse /user com as credenciais de user/password.

Se precisar de uma pasta zip com a estrutura do projeto, avise! 😊