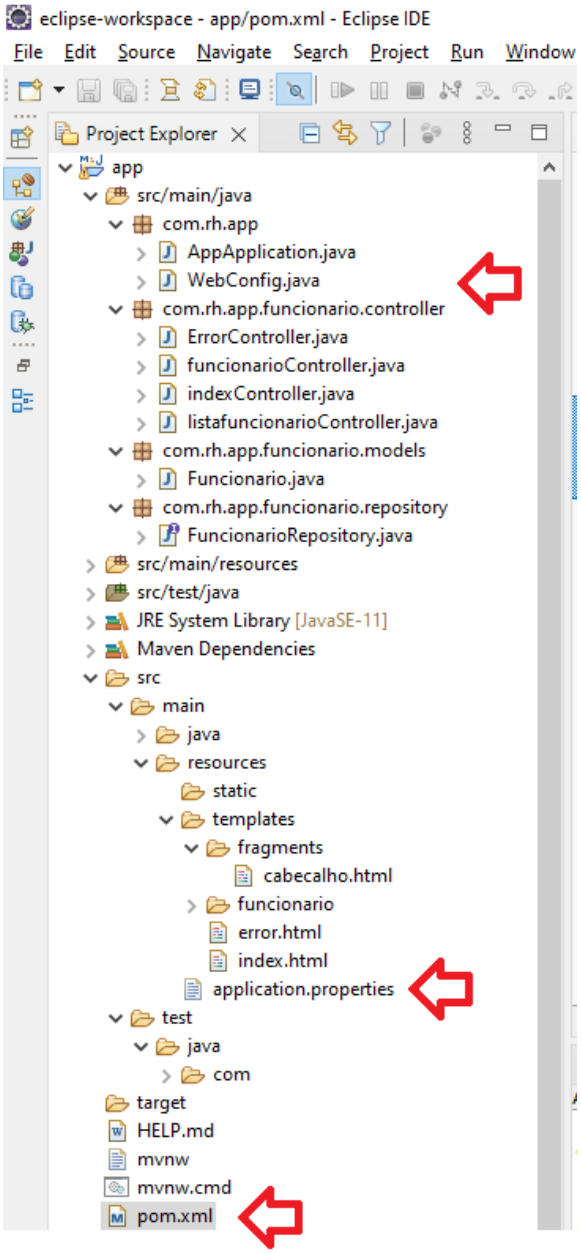
Segurança

Considere a estrutura de pasta do projeto para aplicação do Spring Security, observe as setas com os arquivos **pom.xml** para gerenciar as dependências; **aplication.properties** para configuração inicial da dependência adicionada no pom.xml; e **WebConfig** para configuração personalizada pelo programador.



1. O arquivo **pom.xml** deve conter o seguinte conteúdo, considere as linhas em realce como adição das dependências do Spring Security.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.5.6</version>
                <relativePath /> <!-- lookup parent from repository -->
        </parent>
        <groupId>com.rh</groupId>
        <artifactId>app</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>app</name>
        <description>Aplicação Spring para Sistema de RH</description>
        <properties>
                <java.version>11</java.version>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-security</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-thymeleaf</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.thymeleaf.extras</groupId>
                        <artifactId>thymeleaf-extras-springsecurity5</artifactId>
                </dependency>

                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-devtools</artifactId>
                        <scope>runtime</scope>
                        <optional>true</optional>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>org.springframework.security</groupId>
                        <artifactId>spring-security-test</artifactId>
                        <scope>test</scope>
                </dependency>

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

                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-java</artifactId>
```

```
                <scope>runtime</scope>
            </dependency>

            <dependency>
                <groupId>org.hibernate.validator</groupId>
                <artifactId>hibernate-validator</artifactId>
            </dependency>

        </dependencies>

        <build>
            <plugins>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
            </plugins>
        </build>

</project>
```

2. O arquivo **application.properties** recebe a primeira configuração do Spring Security, caso não seja encontrada uma classe que aplique a anotação **@Configuration**, a qual implementa as regras para configurações. Caso seja encontrada essa classe, ela terá prioridade sobre a configuração do Spring Security. Observe as linhas em realce, em que constam o padrão de login da aplicação que será **admin** e a senha, que também será **admin**.

```
spring.datasource.url=jdbc:mysql://localhost:3306/appfunc
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database-platform = org.hibernate.dialect.MariaDBDialect
spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto = update
spring.security.user.name=admin
spring.security.user.password=admin
```

3. O arquivo **WebConfig.java**, na linha 10, utiliza a anotação **@Configuration**, que é a anotação para configuração do projeto que o Spring busca quando inicia o projeto. Na linha 11, o adaptador WebSecurityConfigurerAdapter, que é estendido em **WebConfig**, configura o Spring Security, conforme as instruções do programador, com as regras de negócio ou requisitos do projeto.

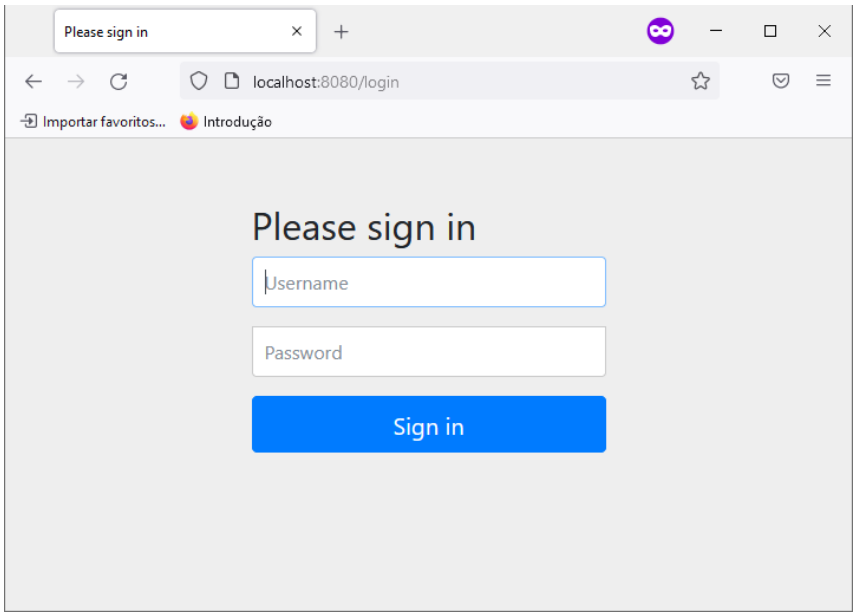
```
1 package com.rh.app;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.context.annotation.Configuration;
5 import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
6 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
7 import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
8 /** @author Rolfi Luz - Senai * */
9
10 @Configuration
11 public class WebConfig extends WebSecurityConfigurerAdapter {
12
```

```
13      @Autowired
14      public void configureGlobal(AuthenticationManagerBuilder builder) throws Exception {
15              builder
16              .inMemoryAuthentication()
17              .withUser("rolfi").password("{noop}rolfi").roles("USER")
18              .and().withUser("root").password("{noop}root").roles("ADMIN");
19      }
20
21  //@Override
22      protected void configure(HttpSecurity http) throws Exception {
23              http
24              .authorizeRequests().antMatchers("/").permitAll()
25              .antMatchers("/home**").permitAll().anyRequest()
26              .authenticated().and().formLogin().permitAll()
27              .and().logout().permitAll()
28              .and().csrf().disable();
29      }
30  }
```

A **configureGlobal** é uma função que configura as regras de entrada do projeto e cria automaticamente a tela de login com os usuários e as senhas do sistema.

Na forma de login, como **inMemoryAuthentication**, na linha 16, são configurados o usuário e a senha do projeto de forma estática. Nas senhas, a palavra reservada **{noop}** (NoopPasswordEncoder) significa que a senha não terá um encoder (criptografia) e que da forma que estiver escrita no **password**, será a senha que o usuário deverá digitar.

A função **configure**, na linha 22, possui as restrições do Spring Security. Nas linhas 24, 25, 26 e 27, permite-se que qualquer usuário navegue nessas páginas, e as demais somente serão acessadas com login. Na linha 26, por meio da função **formLogin()**, é criada automaticamente a tela de login e senha pelo Spring Security, configurada com as credenciais de autenticação da função **configureGlobal**. Já a função **logout()** gera as programações para retirar as credenciais do sistema.



Existem outras formas de autenticação como a **aplication.properties**, a qual tem o mesmo funcionamento da **inMemoryAuthentication**. Há também a forma de login com o **JPA**, por meio do qual são criados usuários dentro do banco de dados e são implementadas as classes **CustomUserDetailService** e **UserDetails**.