**Конфигурация безопасности (SecurityConfig):**

import org.springframework.context.annotation.Bean;

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.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration

@EnableWebSecurity public class SecurityConfig extends WebSecurityConfigurerAdapter { @Override protected void configure(HttpSecurity http) throws Exception { http.authorizeRequests() .antMatchers("/public/\*\*").permitAll() // Публичные ресурсы .antMatchers("/admin/\*\*").hasRole("ADMIN") // Доступ только для администраторов .anyRequest().authenticated() .and() .formLogin().permitAll() // Включаем форму логина .and() .logout().permitAll(); // Включаем логаут }

@Override protected void configure(AuthenticationManagerBuilder auth) throws Exception { auth.inMemoryAuthentication() .withUser("admin").password(passwordEncoder().encode("adminPass")).roles("ADMIN") .and() .withUser("user").password(passwordEncoder().encode("userPass")).roles("USER"); } @Bean @Override public UserDetailsService userDetailsService() { UserDetails user = User.builder().username("user").password(passwordEncoder().encode("userPass")).roles("USER").build(); UserDetails admin = User.builder().username("admin").password(passwordEncoder().encode("adminPass")).roles("ADMIN").build(); return new InMemoryUserDetailsManager(user, admin); }

@Bean public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); } }

Этот класс конфигурации безопасности определяет правила доступа к различным конечным точкам вашего приложения. В данном случае, есть публичные ресурсы, доступные всем пользователям, а также ресурсы, доступные только администраторам. Логин и пароли определены в памяти, но в реальном проекте они были бы получены из базы данных или другого источника.

Эта конфигурация использует кодировщик паролей BCrypt для безопасного хранения паролей.