Informações tela 01

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
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.PasswordEncoder;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@SpringBootApplication
public class DemoApplication {

  public static void main(String[] args) {
    SpringApplication.run(DemoApplication.class, args);
  }

  @Controller
  public static class WebController {

    @GetMapping("/login")
    public String login() {
      return "login";
    }

    @GetMapping("/cadastro")
    public String cadastro() {
      return "cadastro";
    }

    @GetMapping("/confirmacao")
    public String confirmacao() {
      return "confirmacao";
    }
  }

  @EnableWebSecurity
  public static class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
      http
        .authorizeRequests()
          .antMatchers("/cadastro").permitAll()
          .anyRequest().authenticated()
          .and()
```

```
            .formLogin()
                .loginPage("/login")
                .permitAll()
                .and()
            .logout()
                .permitAll();
    }

    @Bean
    @Override
    public UserDetailsService userDetailsService() {
        UserDetails user = User.withUsername("usuario")
            .password(passwordEncoder().encode("senha"))
            .roles("USER")
            .build();

        return new InMemoryUserDetailsManager(user);
    }

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

Informações tela 02

```
import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

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.PasswordEncoder;

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

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

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

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;


@SpringBootApplication
```

```
public class DemoApplication {

  public static void main(String[] args) {
    SpringApplication.run(DemoApplication.class, args);
  }

  @Controller
  public static class WebController {

    @GetMapping("/login")
    public String login() {
      return "login";
    }

    @GetMapping("/cadastro")
    public String cadastro() {
      return "cadastro";
    }
  }

  @EnableWebSecurity
  public static class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
      http
        .authorizeRequests()
          .antMatchers("/cadastro").permitAll()
          .anyRequest().authenticated()
          .and()
        .formLogin()
          .loginPage("/login")
          .permitAll()
```

```
            .and()
        .logout()
            .permitAll();
    }

    @Bean
    @Override
    public UserDetailsService userDetailsService() {
      UserDetails user = User.withUsername("usuario")
          .password(passwordEncoder().encode("senha"))
          .roles("USER")
          .build();

      return new InMemoryUserDetailsManager(user);
    }

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

Informações telas 3 a 9

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.ui.Model;

@Controller
public static class WebController {

  // Código de login, cadastro, principal, e métodos anteriores...

  @GetMapping("/alterar-senha")
  public String alterarSenha() {
    return "alterar-senha";
  }

  @PostMapping("/alterar-senha")
```

```
    public String salvarNovaSenha(@RequestParam  String senhaAtual, @RequestParam String novaSenha) {
        // Lógica para alterar a senha
        return "redirect:/principal";
    }

    @GetMapping("/listar-usuarios")
    public String listarUsuarios(Model  model) {
        // Lógica para obter lista de usuários
        model.addAttribute("usuarios", /* lista de usuários */);
        return "listar-usuarios";
    }

    @GetMapping("/bloquear-desbloquear")
    public String bloquearDesbloquear(Model  model) {
        // Lógica para obter lista de usuários
        model.addAttribute("usuarios", /* lista de usuários */);
        return "bloquear-desbloquear";
    }

    @PostMapping("/bloquear-usuario")
    public String bloquearUsuario(@RequestParam  Long id) {
        // Lógica para bloquear o usuário pelo ID
        return "redirect:/bloquear-desbloquear";
    }

    @PostMapping("/desbloquear-usuario")
    public String desbloquearUsuario(@RequestParam  Long id) {
        // Lógica para desbloquear o usuário pelo ID
        return "redirect:/bloquear-desbloquear";
    }

    @GetMapping("/grupos")
    public String gerenciarGrupos(Model  model) {
        // Lógica para obter lista de grupos
        model.addAttribute("grupos", /* lista de grupos */);
        return "gerenciar-grupos";
    }

    @PostMapping("/incluir-grupo")
    public String incluirGrupo(@RequestParam  String novoGrupo) {
        // Lógica para incluir  novo grupo
        return "redirect:/grupos";
    }

    @PostMapping("/alterar-grupo")
    public String alterarNomeGrupo(@RequestParam  String grupoExistente, @RequestParam String
novoNomeGrupo) {
        // Lógica para alterar nome do grupo
        return "redirect:/grupos";
    }
}
```