



Curso Superior de Desenvolvimento de Software Multiplataforma

Alinne Martins Cardoso, 3011392323003

Atividade TDD
Técnica de Programação II

Prof^o Cláudio Roberto Corredato

Votorantim
Agosto, 2024

Índice de figuras

Figura 1: Model do Usuário.....	3
Figura 2: Controller do Usuário.....	4
Figura 3: Service do Usuário.....	5
Figura 4: Service do Usuário.....	6
Figura 5: View - Tela de Cadastro.....	7
Figura 6: Tela de Cadastro.....	9
Figura 7: Tela de Cadastro - Preenchida.....	10
Figura 8: Terminal - Exibe as informações após o cadastro.....	11
Figura 9: View - Tela de Login.....	12
Figura 10: Tela de Login - Senha Incorreta.....	15
Figura 11: Tela de Login - Exibição da mensagem devido a senha incorreta.....	16
Figura 12: Tela de Login - Senha Correta.....	17
Figura 13: Tela de Login - Exibição da mensagem devido a senha correta.....	18
Figura 14: Teste - Model Usuário.....	19
Figura 15: Teste - Controller Usuário.....	20

Figura 1: Model do Usuário

```
public class UserModel {  
    //Atributos  
    private Long id;  
    private String usuario;  
    private String senha;  
    private String email;  
  
    public UserModel(Long id, String usuario, String senha, String email) {  
        this.id = id;  
        this.usuario = usuario;  
        this.senha = senha;  
        this.email = email;  
    }  
  
    public Long getId() {  
        return id;  
    }  
  
    public String getUsuario() {  
        return usuario;  
    }  
  
    public String getSenha() {  
        return senha;  
    }  
  
    public String getEmail() {  
        return email;  
    }  
  
    public void setId(Long id) {  
        this.id = id;  
    }  
  
    public void setUsuario(String usuario) {  
        this.usuario = usuario;  
    }  
  
    public void setSenha(String senha) {  
        this.senha = senha;  
    }  
  
    public void setEmail(String email) {  
        this.email = email;  
    }  
}
```

Figura 2: Controller do Usuário

```
public class UserController {  
    //Atributos  
    private UserService userService = new UserService();  
  
    //Métodos  
  
    //Criar Usuário  
    public UserModel createUser(String usuario, String senha, String email){  
        UserModel user = new UserModel(null, usuario, senha, email);  
        UserModel createdUser = userService.saveUser(user);  
        System.out.println("Usuário criado(ID): " + createdUser.getId());  
        return createdUser;  
    }  
  
    //Listar Usuário List<UserModel>  
    public void listUsers(){  
        List<UserModel> listaUsers = userService.getAllUsers();  
        System.out.println("Usuários:");  
        listaUsers.forEach(System.out::println);  
    }  
  
    // Verifica se o Usuário existe pelo ID  
    public boolean checkUserExists(Long userId) {  
        UserModel user = userService.getUserById(userId);  
        if(user !=null) return true;  
        boolean exists = user != null;  
        System.out.println("Usuário com ID " + userId + (exists ? " existe." : " não existe."));  
        return exists;  
    }  
  
    // Autenticar Usuário  
    public boolean authenticateUser(String email, String senha) {  
        boolean isAutentincate = userService.authenticateUser(email, senha);  
        return isAutentincate;  
    }  
}
```

Figura 3: Service do Usuário

```
public class UserService {
    private List<UserModel> listaUsers;
    private long idCounter = 1;

    public UserService(){
        listaUsers = new ArrayList<>();
    }

    //Métodos
    //salvar Usuário
    public UserModel saveUser(UserModel user) {
        if (user.getId() == null) {
            user.setId(idCounter++);
            listaUsers.add(user);
        } else {
            for (int i = 0; i < listaUsers.size(); i++) {
                if (listaUsers.get(i).getId().equals(user.getId())) {
                    listaUsers.set(i, user);
                }
            }
        }
        return user;
    }

    //Listar Usuários
    public List<UserModel> getAllUsers() {
        return listaUsers;
    }

    // Pega o Usuário pelo ID
    public UserModel getUserById(Long userId) {
        for (UserModel user : listaUsers) {
            if (user.getId().equals(userId)) {
                return user;
            }
        }
        return null; // Retorna null se o usuário não for encontrado
    }
}
```

Figura 4: Service do Usuário

```
public boolean authenticateUser(String email, String senha) {  
    // Obtém a lista de todos os usuários  
    listaUsers = getAllUsers();  
  
    // Variável para verificar se o usuário foi encontrado  
    boolean userFound = false;  
  
    // Itera sobre a lista de usuários  
    for (int i = 0; i < listaUsers.size(); i++) {  
        UserModel user = listaUsers.get(i);  
  
        // Verifica se o e-mail e a senha correspondem a um usuário na lista  
        if (user.getEmail().equals(email) && user.getSenha().equals(senha)) {  
            userFound = true;  
            break; // Encerra o loop assim que o usuário é encontrado  
        }  
    }  
  
    // Mensagem de sucesso ou erro dependendo da autenticação  
    if (userFound) {  
        System.out.println("Login efetuado com sucesso.");  
        return true;  
    } else {  
        System.out.println("E-mail ou senha incorretos.");  
    }  
    return false;  
}  
  
}
```

Figura 5: View - Tela de Cadastro

```

public CadastroView() {
    initComponents();
    userController = new UserController();
}

/** This method is called from within the constructor to initialize the form ...5 lines */
@SuppressWarnings("unchecked")
Generated Code

private void txtEmailActionPerformed(java.awt.event.ActionEvent evt) {...3 lines }

private void txtSenhaActionPerformed(java.awt.event.ActionEvent evt) {...3 lines }

private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {...4 lines }

private void btnLoginMouseClicked(java.awt.event.MouseEvent evt) {
    String usuario = txtUsuario.getText();
    String email = txtEmail.getText();
    String senha = txtSenha.getText();
    System.out.println(userController.createUser(usuario, senha, email));
}

private void btnFazerLoginMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    new LoginView().setVisible(true);
    new CadastroView().setVisible(false);
}

private void btnFazerLoginActionPerformed(java.awt.event.ActionEvent evt) {...3 lines }

/**...3 lines */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new CadastroView().setVisible(true);
        }
    });
}
}

```

1. Tela de Cadastro

1.1 Descrição: A tela de cadastro tem como objetivo permitir que novos usuários se registrem no sistema.

1.2 Campos:

- **Usuário:** Campo de texto obrigatório para o nome completo do usuário.
- **Email:** Campo de texto obrigatório para o email do usuário.
- **Senha:** Campo de texto obrigatório para a senha do usuário, com visualização mascarada.

1.3 Botões:

- **Cadastrar:** Botão para enviar os dados do formulário para o sistema, efetuando o cadastro do usuário.
- **Fazer Login:** Botão para ir para a tela de login e finalizar a operação de cadastro.

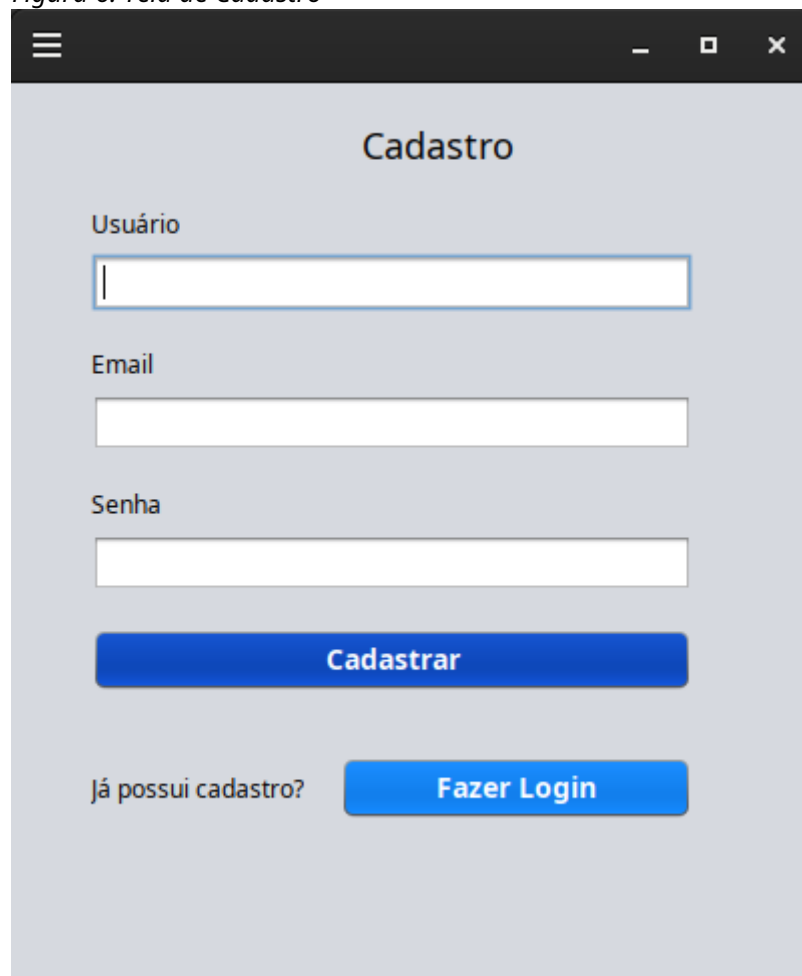
1.4 Validações: (Não Implementado)

- **Nome Completo:** Verifica se o campo está preenchido e se contém apenas letras e espaços.
- **Email:** Verifica se o campo está preenchido e se o formato do email é válido (ex: nome@dominio.com).
- **Senha:** Verifica se o campo está preenchido, se possui no mínimo 8 caracteres e se contém ao menos 1 letra maiúscula, 1 letra minúscula e 1 número.
- **Confirmar Senha:** Verifica se o campo está preenchido e se a senha digitada é igual à primeira.

1.5 Mensagens: (Não Implementado)

- **Erro:** Exibe mensagens de erro caso os dados do formulário não atendam às validações.
- **Sucesso:** Exibe mensagens de sucesso após o cadastro realizado com sucesso.

Figura 6: Tela de Cadastro



The image shows a web application window titled "Cadastro". It features a dark header bar with a menu icon (three horizontal lines) on the left and standard window control buttons (minimize, maximize, close) on the right. The main content area has a light gray background. At the top center, the word "Cadastro" is displayed in a bold, black font. Below this, there are three input fields, each preceded by a label: "Usuário", "Email", and "Senha". The "Usuário" field contains a single vertical line cursor. Below the input fields is a prominent blue button with the text "Cadastrar" in white. At the bottom left, the text "Já possui cadastro?" is followed by a blue button labeled "Fazer Login" in white.

Cadastro

Usuário

Email

Senha

Cadastrar

Já possui cadastro? **Fazer Login**

Figura 7: Tela de Cadastro - Preenchida

The image shows a web application window with a dark header bar containing a menu icon (three horizontal lines) on the left and standard window control icons (minimize, maximize, close) on the right. The main content area has a light gray background and is titled "Cadastro" in a large, bold, black font. Below the title, there are three input fields, each preceded by a label: "Usuário" (Username) with the value "john_doe", "Email" with the value "john@example.com", and "Senha" (Password) with the value "password123". The "Senha" field has a blue border, indicating it is the active field. Below the input fields is a large blue button with the text "Cadastrar" in white. At the bottom left, there is a link "Já possui cadastro?" (Already have an account?). To its right is another blue button with the text "Fazer Login" (Login) in white.

Cadastro

Usuário
john_doe

Email
john@example.com

Senha
password123

Cadastrar

Já possui cadastro? **Fazer Login**

Figura 8: Terminal - Exibe as informações após o cadastro

```
--- exec:3.1.0:exec (default-cli) @ TPI-II ---  
Usuário criado(ID): 1  
model.UserModel@7a71a642
```

Figura 9: View - Tela de Login

```
public class LoginView extends javax.swing.JFrame {
    UserController userController;
    /** Creates new form LoginView ...3 lines */
    public LoginView() {
        initComponents();
        userController = new UserController();
    }

    /** This method is called from within the constructor to initialize the form ...5 lines */
    @SuppressWarnings("unchecked")
    Generated Code

    private void txtEmailActionPerformed(java.awt.event.ActionEvent evt) {...3 lines }

    private void txtSenhaActionPerformed(java.awt.event.ActionEvent evt) {...3 lines }

    private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {...3 lines }

    private void btnLoginMouseClicked(java.awt.event.MouseEvent evt) {
        String email = txtEmail.getText();
        String senha = txtSenha.getText();
        boolean isAuthenticated = userController.authenticateUser(email, senha);
        if (isAuthenticated) {
            // Autenticação bem-sucedida, redirecionar para a próxima tela
            JOptionPane.showMessageDialog(this, "Login Efetuado com Sucesso!");
            this.dispose(); // Fechar a janela de login
        } else {
            // Autenticação falhou, mostrar mensagem de erro
            JOptionPane.showMessageDialog(this, "E-mail ou senha incorretos.");
        }
    }

    private void btnFazerCadastroMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        new LoginView().setVisible(false);
        new CadastroView().setVisible(true);
    }

    /**...3 lines */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new LoginView().setVisible(false);
            }
        });
    }
}
```

2. Tela de Login

2.1 Descrição: A tela de login tem como objetivo permitir que os usuários autentiquem-se no sistema com suas credenciais.

2.2 Campos:

- **Email:** Campo de texto obrigatório para o email do usuário.
- **Senha:** Campo de texto obrigatório para a senha do usuário.

2.3 Botões:

- **Entrar:** Botão para enviar os dados do formulário para o sistema, efetuando a autenticação do usuário.
- **Fazer Cadastro:** Link para a tela de cadastro.

2.4 Validações: (Não Implementado)

- **Email:** Verifica se o campo está preenchido e se o formato do email é válido (ex: [nome@dominio.com](#)).
- **Senha:** Verifica se o campo está preenchido.

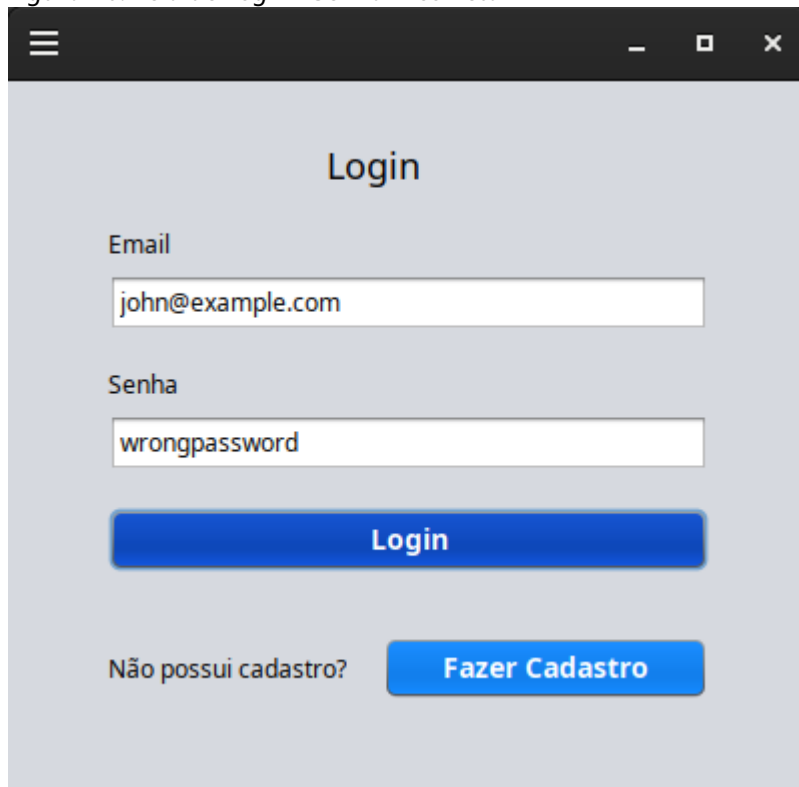
2.5 Mensagens:

- **Erro:** Exibe mensagens de erro caso as credenciais do usuário estejam incorretas.
- **Sucesso:** Após a autenticação com sucesso, o usuário é direcionado para a tela principal do sistema.

3. Fluxos:

- **Cadastro:**
 - O usuário preenche o formulário de cadastro com os dados válidos.
 - O sistema valida os dados do formulário.
 - Caso os dados sejam válidos, o usuário é cadastrado no sistema.
- **Login:**
 - O usuário preenche o formulário de login com os dados válidos.
 - O sistema valida as credenciais do usuário.
 - Caso as credenciais sejam válidas, o usuário é autenticado no sistema.
 - O usuário recebe uma mensagem de sucesso.

Figura 10: Tela de Login - Senha Incorreta



The image shows a web browser window with a dark header bar containing a menu icon (three horizontal lines) on the left and standard window control buttons (minimize, maximize, close) on the right. The main content area has a light gray background. At the top center, the word "Login" is displayed in a large, bold, black font. Below this, there are two input fields. The first is labeled "Email" and contains the text "john@example.com". The second is labeled "Senha" and contains the text "wrongpassword". Below the password field is a prominent blue button with the text "Login" in white. At the bottom left, the text "Não possui cadastro?" is displayed. To its right is another blue button with the text "Fazer Cadastro" in white.

Login

Email

john@example.com

Senha

wrongpassword

Login

Não possui cadastro?

Fazer Cadastro

Figura 11: Tela de Login - Exibição da mensagem devido a senha incorreta

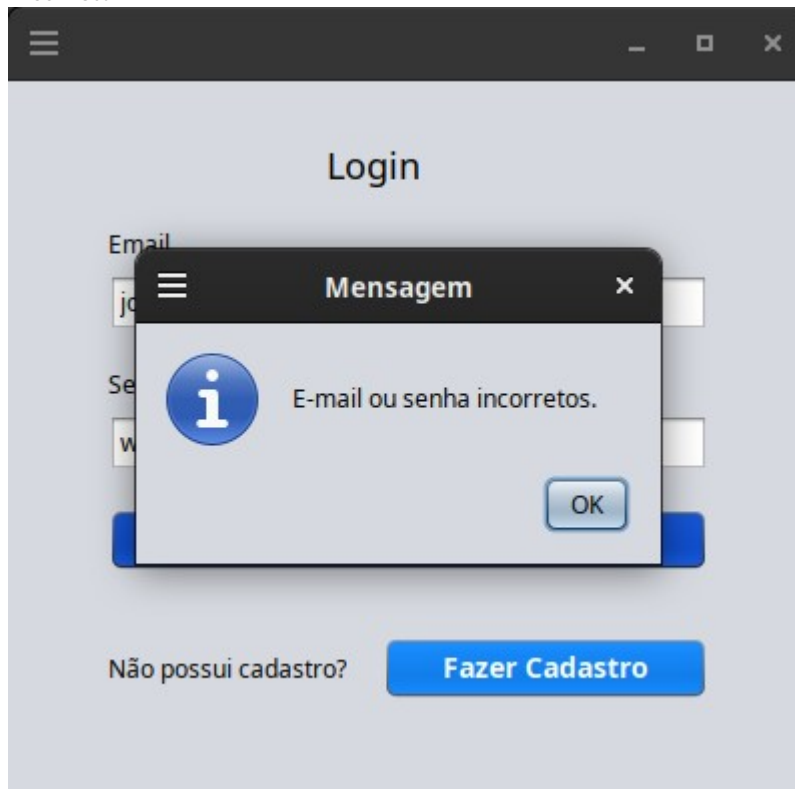
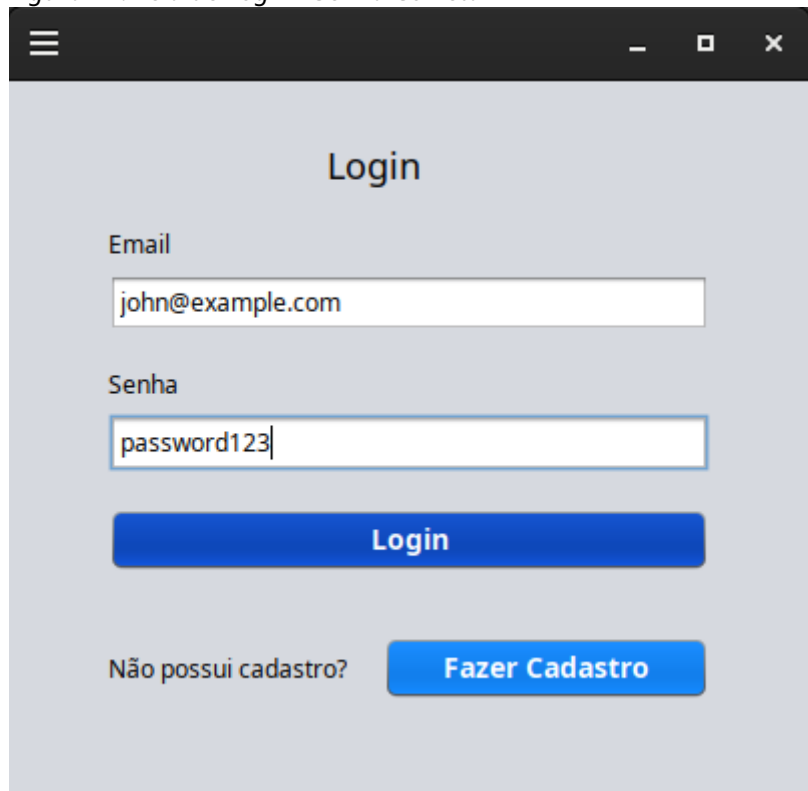


Figura 12: Tela de Login - Senha Correta



The image shows a web application window with a dark header bar containing a menu icon (three horizontal lines) on the left and standard window control icons (minimize, maximize, close) on the right. The main content area has a light gray background. At the top center of this area is the word "Login" in a large, bold, black font. Below this, there are two input fields. The first is labeled "Email" and contains the text "john@example.com". The second is labeled "Senha" and contains the text "password123". Below the password field is a prominent blue button with the text "Login" in white. At the bottom of the form, there is a link "Não possui cadastro?" followed by a blue button with the text "Fazer Cadastro" in white.

Login

Email

john@example.com

Senha

password123

Login

Não possui cadastro? Fazer Cadastro

Figura 13: Tela de Login - Exibição da mensagem devido a senha correta

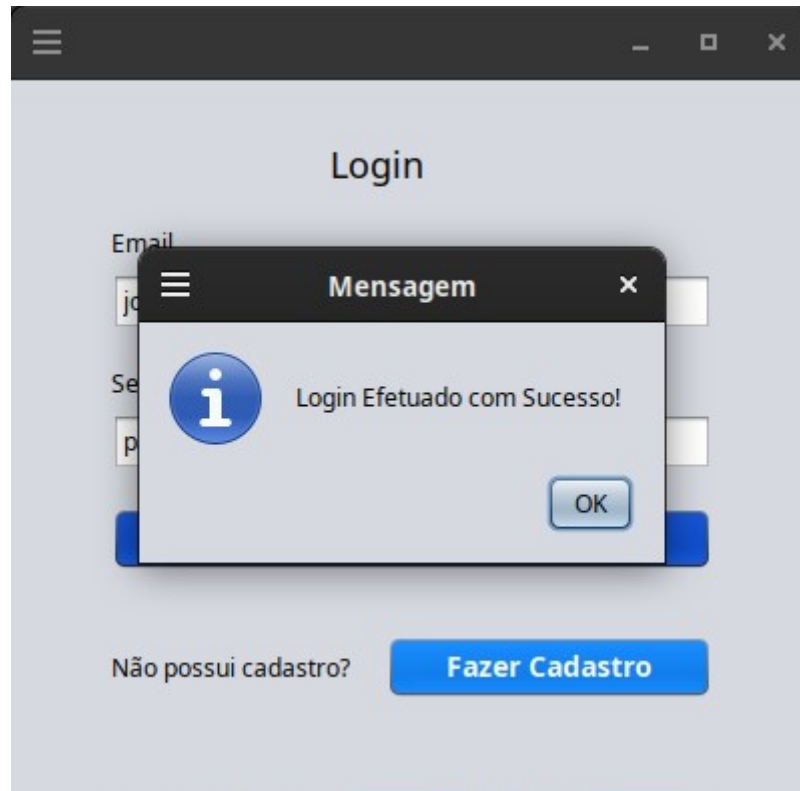


Figura 14: Teste - Model Usuário

```
public class ModelTest {

    @Test
    public void testUserConstructorAndGetters() {
        // Arrange
        String usuario = "alinne";
        String senha = "password123";
        String email = "alinne@example.com";

        // Act
        UserModel user = new UserModel(null, usuario, senha, email);

        // Assert
        assertEquals(usuario, user.getUsuario(), "O usuário deve ser 'alinne'");
        assertEquals(senha, user.getSenha(), "A senha deve ser 'password123'");
        assertEquals(email, user.getEmail(), "O e-mail deve ser 'alinne@example.com'");
    }

    @Test
    public void testSetters() {
        // Arrange
        UserModel user = new UserModel(null, "initial_user", "initial_password", "initial.email@example.com");

        // Act
        user.setUsuario("new_user");
        user.setSenha("new_password");
        user.setEmail("new.email@example.com");

        // Assert
        assertEquals("new_user", user.getUsuario(), "O usuário deve ser 'new_user'");
        assertEquals("new_password", user.getSenha(), "A senha deve ser 'new_password'");
        assertEquals("new.email@example.com", user.getEmail(), "O e-mail deve ser 'new.email@example.com'");
    }
}
```

Figura 15: Teste - Controller Usuário

```
public class ControllerTest {
    // Criação do controlador e configuração inicial
    UserController userController = new UserController();

    public ControllerTest() {
        // Criação de usuários para teste
        UserService userService = new UserService();
        userService.saveUser(new UserModel(1L, "john_doe", "password123", "john@example.com"));
        userService.saveUser(new UserModel(2L, "jane_doe", "password456", "jane@example.com"));

        // Testando a autenticação
        userController.authenticateUser("john@example.com", "password123"); // Deve imprimir "Login Efetuado"
        userController.authenticateUser("john@example.com", "wrongpassword"); // Deve imprimir "E-mail ou senha incorretos"
        userController.authenticateUser("unknown@example.com", "password123"); // Deve imprimir "E-mail ou senha incorretos"
    }
}
```