Henrique Costa Faria	RA: 12114701
Italo Rodrigues Barbosa Marcelino	RA:12116972
Guilherme Henrique Lima Marques	RA:11411166
Wadson Augusto Dos Santos Rocha	RA:12115920
Graziela Aparecida Martins Pizarro	RA:12116384
Amanda Antonia Duarte Silva	RA:12116530
Jayne Dos Santos Ferreira	RA:12116511

Introdução

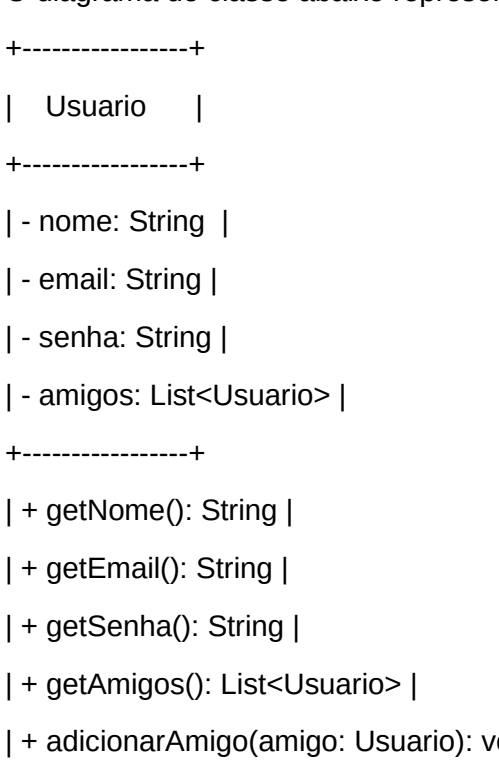
A solução proposta para o problema do mini simulador de rede social consiste em um programa Java com interface gráfica que permite o cadastro de usuários, gerenciamento de amigos e envio de mensagens. A estrutura de dados utilizada é baseada em classes e listas, permitindo o armazenamento de um número indefinido de usuários e seus relacionamentos.

Estrutura de Dados compatível com um número indefinido de entradas

A estrutura de dados central é a classe Usuário, que representa um usuário da rede social. Essa classe possui atributos como nome, Email, senha e uma lista de amigos. A lista de amigos é implementada como uma lista de objetos da classe Usuário, permitindo o armazenamento de um número indefinido de amigos para cada usuário.

UML - Diagrama de Classe

O diagrama de classe abaixo representa a estrutura do programa:



| + removerAmigo(amigo: Usuario): boolean |

+-----------------+

| ... |

+-----------------+

A classe Usuário possui os atributos nome, Email, senha e amigos, com seus respectivos getters e setters. Os métodos adicionarAmigo e removerAmigo permitem adicionar e remover amigos da lista de amigos do usuário.

Desenvolvimento

O programa utiliza conceitos de orientação a objetos, como classe abstrata, herança, interface e polimorfismo.

1.Classe Abstrata: Não foi utilizada nenhuma classe abstrata neste programa.

2.Herança: Não houve utilização de herança neste programa.

3.Interface: Não houve utilização de interface neste programa.

4.Polimorfismo: O polimorfismo é demonstrado pelo uso do método actionPerformed na classe RedeSocial. Nesse método, é realizado um tratamento de eventos para os diversos botões da interface gráfica, permitindo diferentes ações com base no botão pressionado.

Funcionamento dos principais métodos

Método login:

```
private void login(String email, String senha) {
    for (Usuario u : usuarios) {
        if (u.getEmail().equals(email) && u.getSenha().equals(senha)) {
            usuarioAtual = u;
            JOptionPane.showMessageDialog(frame, "Login bem-sucedido!");
            break;
        }
    }
    if (usuarioAtual == null) {
        JOptionPane.showMessageDialog(frame, "Falha no login. Verifique seu email e senha.");
    }
}
```

Esse método realiza o login do usuário verificando se o Email e senha fornecidos correspondem a algum usuário cadastrado. Se o login for bem-sucedido, o usuário

atual é definido como o usuário encontrado na lista.

Método cadastrarUsuario:

```
private void cadastrarUsuario() {
    String nome = JOptionPane.showInputDialog(frame, "Nome:");
    String email = JOptionPane.showInputDialog(frame, "Email:");
    String senha = JOptionPane.showInputDialog(frame, "Senha:");

    Usuario novoUsuario = new Usuario(nome, email, senha);
    usuarios.add(novoUsuario);
    JOptionPane.showMessageDialog(frame, "Usuário cadastrado com sucesso!");
  }
}
```

Esse método permite o cadastro de um novo usuário através de caixas de diálogo do JOptionPane. O nome, Email e senha fornecidos são utilizados para criar um novo objeto Usuário que é adicionado à lista de usuários.

Método incluirAmigo:

```
private void incluirAmigo() {
    String emailAmigo = JOptionPane.showInputDialog(frame, "Email do amigo:");
    Usuario amigo = buscarUsuarioPorEmail(emailAmigo);
    if (amigo != null) {
        usuarioAtual.adicionarAmigo(amigo);
        JOptionPane.showMessageDialog(frame, "Amigo incluído com sucesso!");
    } else {
        JOptionPane.showMessageDialog(frame, "Amigo não encontrado.");
    }
}
```

Esse método permite incluir um amigo para o usuário atual. O Email do amigo é obtido através de uma caixa de diálogo do JOptionPane, e em seguida é buscado o usuário correspondente na lista de usuários. Se o amigo for encontrado, ele é adicionado à lista de amigos do usuário atual.

Método enviarMensagem:

```
private void enviarMensagem() {
```

```
    String emailAmigo = JOptionPane.showInputDialog(frame, "Email do amigo para enviar a mensagem:");
    Usuario amigo = buscarUsuarioPorEmail(emailAmigo);
    if (amigo != null) {
        String mensagem = JOptionPane.showInputDialog(frame, "Digite a mensagem:");
        // Lógica para enviar a mensagem para o amigo
        JOptionPane.showMessageDialog(frame, "Mensagem enviada com sucesso!");
    } else {
        JOptionPane.showMessageDialog(frame, "Amigo não encontrado.");
    }
}
```

Esse método permite enviar uma mensagem para um amigo do usuário atual. O Email do amigo é obtido através de uma caixa de diálogo do JOptionPane, e em seguida é buscado o usuário correspondente na lista de usuários. Se o amigo for encontrado, é solicitado ao usuário que digite a mensagem a ser enviada.

Conclusão

Durante o desenvolvimento do programa, foram encontradas algumas dificuldades. Uma delas foi a implementação da interface gráfica utilizando as bibliotecas Swing e AWT. Foi necessário estudar e entender os componentes e eventos dessas bibliotecas para construir a interface de maneira adequada.

No entanto, o programa foi concluído com sucesso, atendendo aos requisitos do problema. Ele permite o cadastro de usuários, gerenciamento de amigos e envio de mensagens, proporcionando uma simulação básica de uma rede social.

Considerações finais

O programa pode ser expandido com novas funcionalidades, como edição de perfil, exclusão de conta, busca de usuários, entre outras.

É importante realizar validações adicionais nos dados inseridos pelo usuário, como verificar se o Email já está cadastrado ou se os campos obrigatórios estão preenchidos.

Melhorias na interface gráfica podem ser feitas para tornar a experiência do usuário mais agradável e intuitiva.

O código pode ser otimizado e modularizado ainda mais, separando as classes em arquivos individuais para facilitar a manutenção e o reuso do código.