CENTRO UNIVERSITARIO NEWTON PAIVA

DAYANE BATISTA DO ROSÁRIO

JADE BRANDAO ALVES DE PAULA

LARISSA GUIMARÃES DA SILVA

RODRIGO FERREIRA CAMARGOS

Belo Horizonte

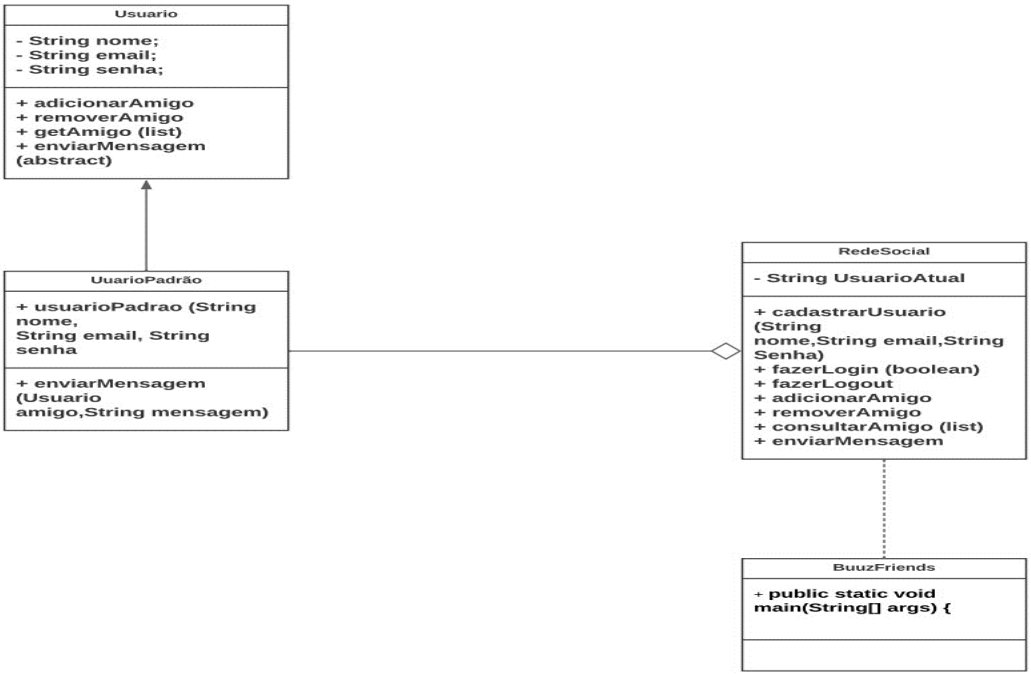
2023

INTRODUÇÃO

A escolha correta das estruturas de dados é um fator importante no desenvolvimento de programas eficientes. Ao desenvolver o projeto Mini Simulador de Rede Social enfrentamos o desafio de lidar com um número indefinido de dados e a necessidade de garantir a escalabilidade e flexibilidade do programa. Neste projeto examinaremos a escolha de estruturas de dados compatíveis que incluíram a manipulação eficiente de um grande volume de informações, garantindo um desempenho otimizado e a capacidade de lidar com diferentes tipos de operações.

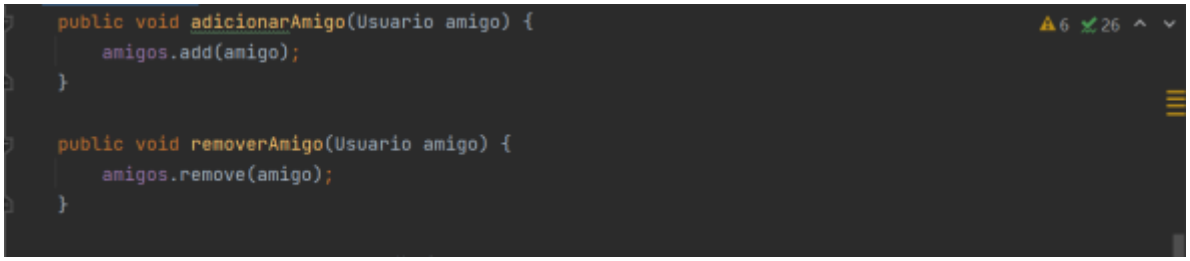
A solução encontrada para o problema apresentado foi a inclusão das estruturas de dados HashMap e List, que possibilitam a criação de uma estrutura flexível e com um número indefinido de entradas.

O uso do HashMap permite associar cada usuário a um e-mail único, possibilitando uma recuperação eficiente dos usuários na rede social e a utilização da List permite armazenar uma lista de amigos de cada usuário.



DESENVOLVIMENTO

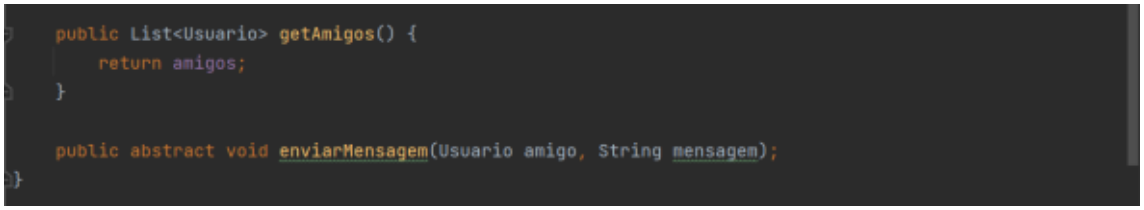
A classe Usuário é abstrata e contém os atributos comuns a todos os tipos de usuários, como nome, e-mail, senha e lista de amigos. Ela possui os métodos adicionar Amigo e remover amigos.



```
    public void adicionarAmigo(Usuario amigo) {
        amigos.add(amigo);
    }

    public void removerAmigo(Usuario amigo) {
        amigos.remove(amigo);
    }
```

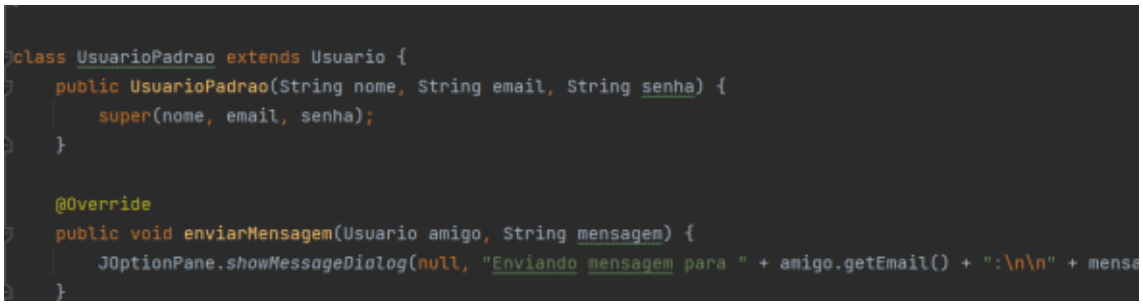
Um método public list<Usuario> que retorna uma lista de amigos do usuário e um método abstrato que sera implementado pelas classes derivadas de usuário e servira para enviar uma mensagem para um amigo.



```
    public List<Usuario> getAmigos() {
        return amigos;
    }

    public abstract void enviarMensagem(Usuario amigo, String mensagem);
}
```

Usuário Padrão é uma subclasse da classe abstrata Usuário que se baseia no polimorfismo para implementar o método enviarMensagem, que exibe uma mensagem de diálogo utilizando o JOptionPane.



```
class UsuarioPadrao extends Usuario {
    public UsuarioPadrao(String nome, String email, String senha) {
        super(nome, email, senha);
    }

    @Override
    public void enviarMensagem(Usuario amigo, String mensagem) {
        JOptionPane.showMessageDialog(null, "Enviando mensagem para " + amigo.getEmail() + ":\n\n" + mensa
    }
```

A classe RedeSocial possui o método cadastrarUsuario.Esse método cadastra um novo usuário na classe RedeSocial,cria uma instancia de "UsuarioPadrao" com o nome, e-mail e senha fornecidos e os adiciona ao mapa de usuários usando o e-mail como chave.

O método fazerLogin realiza o login na rede social verificando se o e-mail e senha fornecidos correspondem a um usuário registrado. Se as informações estiverem corretas, o usuário é definido como o usuário atual e o método retorna true. Caso contrário, retorna false.

O método fazerLogout faz o logout do usuário atual, definindo o usuário atual como null.

```
public void fazerLogout() {
    usuarioAtual = null;
}
```

Esses métodos adicionam e removem um amigo para o usuário atual.

```
public void adicionarAmigo(Usuario amigo) {
    usuarioAtual.adicionarAmigo(amigo);
}

public void removerAmigo(Usuario amigo) {
    usuarioAtual.removerAmigo(amigo);
}
```

O método list<Usuario> consultarAmigos retorna a lista de amigos do usuário atual.

```
public List<Usuario> consultarAmigos() {
    return usuarioAtual.getAmigos();
}
}
```

CONCLUSÃO

Uma das principais dificuldades ao desenvolver um Mini Simulador de Rede Social foram os erros de programação que variam desde erros simples de sintaxe até erros lógicos mais complexos como por exemplo o erro "NullPointerException"" ocorre quando você tenta invocar um método ou acessar uma propriedade de um objeto que está com o valor nulo (null). Nesse caso específico, o erro ocorreu na linha 67 do arquivo BuzzFriends.java. Identificar e corrigir esses erros demanda habilidades de apuração, compreensão do código e paciência para percorrer o código em busca das falhas. É importante considerar a experiência do usuário, garantindo que o programa seja intuitivo, e fácil de usar. Conclui-se que o código apresenta uma estrutura básica para simular uma rede social simples, com funcionalidades limitadas.