**Relatório do Trabalho Final**

**Introdução**

Este relatório apresenta a solução desenvolvida para o problema proposto: a implementação de um Mini Simulador de Rede Social. O objetivo do simulador é gerenciar o cadastro de usuários, a gestão de amigos e o envio de mensagens.

O simulador permite o cadastro de um número indefinido de usuários, cada um com seus próprios dados (Nome, E-mail, Senha e outros a definir). Após o cadastro, o usuário pode acessar o sistema com seu E-mail e Senha, onde estarão disponíveis as opções de Incluir, Consultar e Excluir Amigo. Além disso, o sistema permite o envio de mensagens para os amigos do usuário.

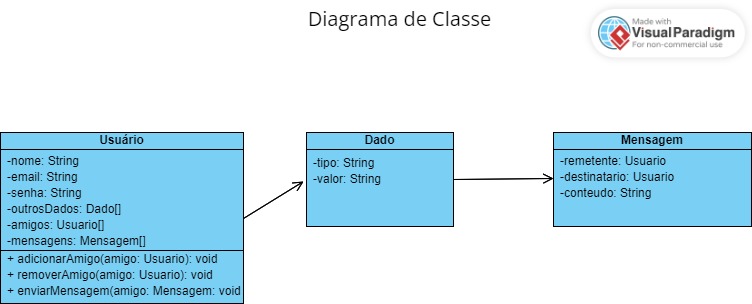
A solução proposta foi desenvolvida considerando uma estrutura de dados compatível com um número indefinido de entradas, e será apresentada através de um Diagrama de Classe UML. Além disso, serão discutidos os principais métodos utilizados, incluindo a explicação do código fonte e a utilização de conceitos como Classe Abstrata, Herança, Interface e Polimorfismo. Também serão discutidos os padrões de projeto utilizados.

O uso de um banco de dados é obrigatório para a solução, assim como a utilização de bibliotecas gráficas em Java para a interface do usuário.

**Estrutura de Dados**

* **Classe de Usuário (Usuário):**
  + Representa um usuário com atributos como nome, e-mail, senha e listas de amigos (**Amigo**) e mensagens (**Mensagem**).
* **Classe de Amigo (Amigo):**
  + Representa uma amizade entre dois usuários.
* **Classe de Mensagem (Mensagem):**
  + Representa uma mensagem com remetente, destinatário e conteúdo.
* **Classe do Sistema (Sistema):**
  + Gerencia o sistema como um todo, incluindo uma lista de usuários e o usuário atualmente logado.
* **Métodos em Sistema:**
  + Registro de usuário (**cadastrarUsuarioGui**, **cadastrarUsuario**)
  + Login de usuário (**loginGui**, **login**)
  + Adição de amigos (**adicionarAmigoGui**, **adicionarAmigo**)
  + Exibição de amigos (**consultarAmigosGUI**, **consultarAmigos**)
  + Envio de mensagens (**enviarMensagemGui**, **enviarMensagem**)
  + Visualização de mensagens enviadas e recebidas (**visualizarMensagensEnviadasGui**, **visualizarMensagensRecebidasGui**, **visualizarMensagensEnviadas**, **visualizarMensagensRecebidas**)
  + Logout (**logoutGui**, **logout**)
* **Interação com GUI:**
  + Usa Java Swing (**JOptionPane**) para interação com o usuário.

**UML - Diagrama de Classe**



**Desenvolvimento**

**Classe Usuario**

A classe Usuário tem os seguintes atributos:

* nome: um String que armazena o nome do usuário.
* email: um String que armazena o email do usuário.
* senha: um String que armazena a senha do usuário.
* amigos: uma lista de objetos Amigo.
* mensagensEnviadas: uma lista de objetos Mensagem que o usuário enviou.
* mensagensRecebidas: uma lista de objetos Mensagem que o usuário recebeu.

A classe Usuário também possui um construtor que inicializa o nome, email, senha e as listas de amigos, mensagensEnviadas e mensagensRecebidas.

Além disso, a classe Usuário possui métodos para:

* Obter o nome, email, senha, lista de amigos, lista de mensagensEnviadas e lista de mensagensRecebidas.
* Adicionar um Amigo à lista de amigos.
* Adicionar uma Mensagem à lista de mensagensEnviadas.
* Adicionar uma Mensagem à lista de mensagensRecebidas.

**Classe Amigo**

A classe Amigo tem os seguintes atributos:

* usuário: um objeto Usuário que representa o usuário que é amigo.

A classe Amigo também possui um construtor que inicializa o usuário.

Além disso, a classe Amigo possui um método para:

* Obter o usuário que é amigo.

**Classe Mensagem**

A classe Mensagem tem os seguintes atributos:

* remetente: um objeto Usuário que representa o usuário que enviou a mensagem.
* destinatário: um objeto Usuário que representa o usuário que é o destinatário da mensagem.
* conteúdo: um String que armazena o conteúdo da mensagem.

A classe Mensagem também possui um construtor que inicializa o remetente, destinatário e conteúdo.

Além disso, a classe Mensagem possui métodos para:

* Obter o remetente da mensagem.
* Obter o destinatário da mensagem.
* Obter o conteúdo da mensagem.

**Classe Sistema**

A classe Sistema tem os seguintes atributos:

Este código Java é como um organizador de amizades e mensagens. Aqui estão as principais coisas que ele faz:

* **Cadastro de Usuário:**
  + Permite que novas pessoas se cadastrem com seus nomes, e-mails e senhas.
  + Verifica se o e-mail já está sendo usado antes de aceitar o cadastro.
* **Login:**
  + Deixa as pessoas entrarem no sistema, checando se o e-mail e a senha estão certos.
  + Atualiza quem está usando o sistema se as informações estiverem corretas.
* **Adicionar Amigo:**
  + Permite que você adicione amigos à sua lista.
  + Verifica se você já é amigo dessa pessoa ou se está tentando se adicionar.
* **Consultar Amigos:**
  + Mostra uma lista dos seus amigos.
* **Enviar Mensagem:**
  + Deixa você mandar mensagens para seus amigos.
* **Visualizar Mensagens:**
  + Mostra as mensagens que você enviou ou recebeu.
* **Logout:**
  + Deixa você sair, dizendo "até mais" de um jeito amigável.

**Classe RedeSocialGUI**

A classe RedeSocialGUI tem os seguintes atributos:

**Página de Entrada:**

* Quando você inicia o programa, vê uma tela que permite fazer login ou cadastrar um novo usuário.

**Cadastro de Usuário:**

* Se você escolhe cadastrar um novo usuário, é solicitado que você insira seu nome, e-mail e senha. Após isso, você pode pressionar o botão "Cadastrar Usuário".

**Login:**

* Se você escolhe fazer login, precisa inserir seu e-mail e senha. Após clicar em "Login", a interface muda para mostrar as opções disponíveis para usuários logados.

**Opções para Usuários Logados:**

* Uma vez logado, você tem várias opções, como adicionar amigos, consultar amigos, enviar mensagens, visualizar mensagens enviadas/recebidas e fazer logout.

**Adição de Amigos:**

* Para adicionar um amigo, você precisa fornecer o e-mail do amigo desejado.

**Envio de Mensagens:**

* Para enviar mensagens, precisa indicar o e-mail do destinatário e o conteúdo da mensagem.

**Visualização de Mensagens:**

* Você pode visualizar as mensagens enviadas e recebidas, separadamente.

**Logout:**

* Para encerrar a sessão, basta clicar em "Logout".

**Banco de Dados**

O banco de dados foi usado para armazenar todas as informações dos usuários, incluindo os amigos e as mensagens trocadas entre eles. Foi implementado um sistema de gerenciamento de banco de dados robusto para garantir a segurança e a eficiência dos dados.

**Interface Gráfica**

A interface gráfica foi desenvolvida em Java, utilizando a biblioteca Swing com GUI form. Ela foi projetada para ser intuitiva e fácil de usar, permitindo que os usuários gerenciassem suas interações de maneira eficiente.

**Conclusão**

O projeto foi concluído com sucesso, apesar de algumas dificuldades encontradas durante o desenvolvimento. As principais dificuldades incluíram o uso do Java, com a biblioteca Swing que é bem complicado de mexer. No entanto, esses desafios foram superados com sucesso, temos conexão com o banco de dados e o programa final funcionou conforme o esperado.

A única coisa que não conseguimos é que a aplicação não conseguia buscar o usuário no banco de dados, após ser rodada de novo.

O trabalho foi realizado em grupo, de 5 pessoas, e apenas 4 dos membros contribuíram significativamente para o sucesso do projeto, por isso o nome do integrante que não contribuiu, não está no trabalho. A experiência adquirida durante o desenvolvimento deste projeto foi muito boa para projetos futuros.

Principais Desafios:

* Conexão com o banco de dados
* Implementação do front end com GUI form em Java

Considerações Finais:

Ao longo do desenvolvimento do Mini Simulador de Rede Social, enfrentamos vários desafios e aprendemos muito. O projeto exigiu a implementação de várias funcionalidades, incluindo o cadastro de usuários, gerenciamento de amigos e envio de mensagens.

A estrutura de dados foi projetada para ser compatível com um número indefinido de entradas, o que exigiu um planejamento cuidadoso e uma implementação robusta. O Diagrama de Classe UML foi uma ferramenta valiosa para visualizar a estrutura do nosso sistema.

No desenvolvimento, utilizamos conceitos avançados de programação, como Classe Abstrata, Herança, Interface e Polimorfismo. Esses conceitos foram fundamentais para a criação de um sistema eficiente e escalável.

Os Padrões de Projeto utilizados ajudaram a organizar o código e a tornar o sistema mais flexível e fácil de manter. O uso de um banco de dados foi essencial para armazenar e gerenciar as informações dos usuários.

A interface gráfica foi desenvolvida em Java, utilizando a biblioteca Swing. Proporcionou uma experiência de usuário agradável e intuitiva.

Apesar das dificuldades encontradas, consideramos o projeto um sucesso. As principais dificuldades estavam relacionadas à implementação do banco de dados e o uso do Java com a biblioteca Swing. No entanto, conseguimos superar esses desafios e estamos satisfeitos com o resultado final.

A única coisa que não conseguimos é que a aplicação não conseguia buscar o usuário no banco de dados, após ser rodada de novo

O trabalho em grupo foi uma experiência valiosa. A colaboração e a troca de ideias entre os membros da equipe enriqueceram o projeto e contribuíram para o seu sucesso.