Visão Geral

O sistema implementa uma aplicação simples de rede social com funcionalidades de login, exibição de mensagens e postagem de novas mensagens. A interface gráfica é construída usando **Java Swing**, e os dados são armazenados em um banco de dados MySQL. O sistema possui as seguintes telas principais:

1. **Login**: Tela onde o usuário informa seu nome para fazer login no sistema.
2. **Home**: Tela principal onde o usuário visualiza mensagens de outros usuários e pode visualizar seu próprio nome.
3. **Postagem**: Tela onde o usuário pode escrever uma nova mensagem e publicá-la no sistema.

A aplicação segue um padrão de arquitetura **MVC** (Model-View-Controller), onde a lógica de controle e manipulação de dados é separada da interface gráfica.

Estrutura de Pacotes

Pacote controller

Este pacote contém as classes responsáveis pela lógica de controle da aplicação, manipulando os dados de mensagens e usuários.

UserDAO

Interface responsável por definir operações para manipulação de usuários.

```
public interface UserDAO{

  public User adicionaUser(User user);

}
```

- **Método**:
  - adicionaUser(User user): Método para adicionar um usuário no banco de dados.

MessageActions

Classe responsável por realizar ações relacionadas às mensagens, como salvar e recuperar mensagens do banco de dados.

```
public class MessageActions {

  // Exemplo de operação relacionada a mensagens

  public static ArrayList<Message> getMessagesFromDatabase() {

    // Recupera mensagens do banco de dados

  }

}
```

- **Métodos**:
- getMessagesFromDatabase(): Recupera todas as mensagens do banco de dados.

UserActions

Classe responsável por realizar ações relacionadas aos usuários, como salvar e recuperar dados dos usuários no banco de dados.

```
public class UserActions {

  public static User getUserByName(String name) {

    // Busca usuário no banco de dados

  }
  public static User addUserToDatabase(String name) {

    // Adiciona um usuário ao banco de dados
}}
```

- **Métodos**:
- getUserByName(String name): Recupera um usuário pelo nome.
- addUserToDatabase(String name): Adiciona um novo usuário ao banco de dados.

Pacote model

Este pacote contém as classes de modelo de dados que representam as entidades do sistema (usuários e mensagens).

Message

Classe que representa uma mensagem enviada por um usuário.

```
public class Message {

  private String nick;

  private String createdAt;

  private String message;

  // Getters e setters

}
```

- **Atributos**:
  - nick: Nome do usuário que enviou a mensagem.
  - createdAt: Data de criação da mensagem.
  - message: Texto da mensagem.

User

Classe que representa um usuário no sistema.

```
public class User {

  private int id;

  private String nick;

  // Getters e setters

}
```

- **Atributos**:
    - id: ID único do usuário no banco de dados.
    - nick: Nome do usuário.

Pacote view

Este pacote contém as classes responsáveis pela interface gráfica do usuário (GUI). As interfaces são construídas com JPanel, JButton, JLabel, entre outros componentes do Swing.

Login

Classe responsável pela tela de login.

java

Copiar código

```
public class Login {
   private JTextField name;
   private JButton continueButton;


   // Métodos para criar a interface
}
```

- **Atributos**:
    - name: Campo de texto para o nome do usuário.
    - continueButton: Botão para continuar após o login.
- **Métodos**:
    - makePanel(): Configura o layout da tela de login.
    - getNameInputText(): Retorna o nome digitado pelo usuário.
    - getContinueButton(): Retorna o botão de login.

Home

Classe responsável pela tela inicial, onde o usuário visualiza as mensagens.

```
public class Home {

   private User currentUser;
```

```
    private ArrayList<Message> mensagens;

    // Métodos para construir a interface

}
```

- **Atributos**:
    - currentUser: Representa o usuário logado.
    - mensagens: Lista de mensagens exibidas na tela.
- **Métodos**:
    - makePanel(): Constrói a tela inicial.
    - setMessages(ArrayList<Message> messages): Atualiza a lista de mensagens exibidas.

Post

Classe responsável pela tela onde o usuário pode criar uma nova postagem.

```
public class Post {

    private User currentUser;

    private JTextArea messageText;

    // Métodos para criar a interface

}
```

- **Atributos**:
    - currentUser: Representa o usuário logado.
    - messageText: Campo de texto onde o usuário digita sua mensagem.
- **Métodos**:
    - makePanel(): Constrói a interface de postagem.
    - getNewMessage(): Retorna o conteúdo da nova mensagem para ser salva.

Pacote view.components

Este pacote (não fornecido) pode conter componentes gráficos personalizados, como o Avatar, que provavelmente é um componente que desenha a imagem ou ícone do usuário.

Avatar

Classe para representar o avatar do usuário.

```
public class Avatar extends JPanel {

  // Lógica para desenhar o avatar

}
```

Pacote database

Este pacote contém a lógica para conexão e manipulação do banco de dados.

ConnectionBD

Classe responsável por conectar a aplicação ao banco de dados MySQL.

```
public class ConnectionBD {

  public static Connection getConnection() {

    // Estabelece a conexão com o banco de dados

  }

}
```

- **Métodos**:
    - getConnection(): Estabelece uma conexão com o banco de dados MySQL.

Fluxo de Execução

1. **Login**: O usuário entra com seu nome na tela de login. Quando clica no botão "Entrar", o nome é enviado para o banco de dados e um novo usuário é registrado (se não existir). O sistema navega para a tela de **Home**.
2. **Home**: A tela inicial exibe o nome do usuário e todas as mensagens disponíveis no banco de dados. O usuário pode clicar em "Novo" para criar uma postagem.
3. **Postagem**: O usuário digita sua mensagem e clica no botão "Publicar". A nova mensagem é salva no banco de dados e a tela de **Home** é atualizada para exibir a nova mensagem.

Considerações Finais

- **Banco de Dados**: A implementação de persistência de dados utiliza JDBC para conectar-se ao banco de dados MySQL e realizar operações de CRUD (Create, Read, Update, Delete).
- **Segurança**: O sistema atual não implementa mecanismos de segurança, como autenticação de senha. Apenas o nome do usuário é utilizado para login.
- **Interface Gráfica**: A interface é simples e funcional, construída com **Java Swing**. Embora a interface seja básica, ela pode ser facilmente personalizada ou expandida, como a adição de ícones e melhorias na disposição dos componentes.