Sumário

Relatório de implementação de interface gráfica e conexão com o banco de dados MySQL do sistema MDFlix

Explicação da arquitetura

Minha estrutura de projeto não recebeu um nome específico, sendo criada por mim para tornar o código mais compreensível. Essa decisão foi tomada visando facilitar a navegação e compreensão do projeto.

Todos os elementos gráficos e o arquivo principal, responsável por iniciar esta seção específica, estão agrupados no pacote padrão (`default`). Essa escolha busca criar uma estrutura coesa, reunindo todas as classes relevantes em um único local. Isso simplifica a identificação e localização desses elementos, promovendo uma organização intuitiva que facilita a compreensão do projeto como um todo.

A lógica de programação foi mantida no próprio pacote `default`. Essa prática permite que as classes relacionadas à manipulação de dados, como conexões de banco de dados, cadastros e funcionalidades similares, estejam em um único pacote. Essa abordagem facilita a localização dessas classes e mantém uma separação clara entre a parte gráfica do projeto e as classes responsáveis pela lógica, evitando a mistura desnecessária de elementos visuais com códigos lógicos.

E possui um pacote chamado images, onde estão localizadas as imagens utilizadas no sistema, essas preferi colocar em um pacote separado para que não houvesse mistura entre códigos e imagens, assim mantendo uma certa organização entre esses elemtos.

Relatório Geral

Para interface gráfica foi utilizado a IDE NetBeans, onde toda a estrutura de interação com usuário foi projetada, e como teria que conectar tudo com o banco de dados, utilizei o HeidiSQL, que na mminha opinião é mais fácil de utilizar e junto com ele utilizei o XAMP que faz o SQL se conectar sem problemas.

A seguir um pouco de cada classe:

Login.java:

Descrição Geral: A classe `Login` é a parte da nossa aplicação que cuida do login dos usuários. Ela é responsável por criar a interface gráfica para que os usuários possam inserir suas informações e acessar o sistema.

Componentes Visuais: Nessa classe, a gente usa vários componentes visuais do Java Swing para criar a interface:

1. `JFrame`: Isso é como a janela principal do nosso programa.
2. `JPanel`: Usamos para organizar e agrupar outros componentes visuais.
3. `JLabel`: São os rótulos que indicam o que cada campo de entrada significa.
4. `JTextField`: É a caixa de texto onde o usuário insere o nome.
5. `JPasswordField`: Aqui é onde a senha é inserida de forma segura.
6. `JButton`: São os botões para iniciar o login e acessar o cadastro.

Funcionalidades:

1. **Login:** Quando o usuário clica no botão "Entrar", a classe verifica as credenciais no banco de dados. Se estiverem corretas, uma mensagem de sucesso é exibida, e a interface de usuários é aberta. Caso contrário, aparece uma mensagem informando que as credenciais estão incorretas.
2. **Cadastro:** Se o usuário quiser se cadastrar, basta clicar no botão "Cadastrar", e uma nova interface é aberta para inserção dos dados.

Integração com Banco de Dados:

1. A classe usa um objeto `java.sql.Connection` para se conectar ao banco de dados.
2. Faz uma consulta SQL para verificar as credenciais do usuário.
3. Utiliza `PreparedStatement` para evitar problemas de segurança.
4. Se ocorrerem erros no banco de dados, o sistema registra mensagens de erro usando o sistema de log.

Gestão de Eventos:

1. Eventos de ação são usados para capturar as interações do usuário, como clicar em botões.

2. O método `actionPerformed` é responsável por lidar com esses eventos, incluindo ações de login e cadastro.

Aspectos Visuais:

1. Escolhemos um esquema de cores preto e vermelho.
2. A fonte "Segoe UI" foi utilizada nos componentes visuais.

Fluxo de Execução Principal:

1. No método `main`, a interface gráfica é inicializada.
2. O usuário preenche nome de usuário e senha.
3. Ao clicar em "Entrar", as credenciais são verificadas no banco de dados e mensagens apropriadas são exibidas.
4. Ao clicar em "Cadastrar", uma nova interface para o cadastro é aberta.

Usuarios.java:

Essa classe Java representa a tela de seleção de usuários no sistema. Aqui estão algumas informações relevantes sobre o que esta classe realiza:

Layout e Componentes Visuais: A classe `Usuarios` é uma extensão de `javax.swing.JFrame`, indicando que representa uma janela gráfica. O layout inclui quatro botões representando diferentes usuários (User 1, User 2, User 3, User 4).

Escolha do Usuário: Quando um dos botões de usuário é clicado, o método de evento correspondente (`jButton1ActionPerformed`, `jButton2ActionPerformed`, `jButton3ActionPerformed`, `jButton4ActionPerformed`) é acionado. Cada um desses métodos cria uma instância da classe `TelaPrincipal` (tela principal do sistema) e a torna visível, enquanto fecha a tela atual.

Estilo Visual: Cada botão tem uma cor de fundo diferente para distinguir visualmente os usuários.

Encerramento e Abertura de Telas: Para abrir a tela principal, a classe atual é fechada (`this.dispose()`) e uma nova instância de `TelaPrincipal` é criada e tornada visível.

Em resumo, a classe `Usuarios` é responsável por fornecer uma interface para os usuários selecionarem um perfil antes de entrar na tela principal do sistema. Atualmente, todos os usuários têm o mesmo comportamento, mas essa lógica pode ser personalizada para cada usuário no futuro, conforme necessário.

CadastroUser1.java:

Descrição Geral: A classe `CadastroUser1` representa a interface gráfica para cadastrar novos usuários em nossa aplicação. Ela inclui campos para informações como nome, e-mail, CPF, senha e data de nascimento, e também implementa funcionalidades de validação dessas informações antes de efetuar o cadastro no banco de dados.

Componentes Visuais:

1. `JFrame`: A janela principal da aplicação.
2. `JPanel`: Utilizado para organizar componentes visuais.
3. `JLabel`: Rótulos que indicam a finalidade de cada campo.
4. `JTextField`: Campos de texto para entrada de dados.
5. `JButton`: Botões para realizar o cadastro e voltar à tela de login.

Funcionalidades:

1. **Cadastro:** Ao clicar no botão "Cadastrar", o sistema verifica e valida as informações fornecidas antes de inserir os dados no banco de dados. Isso inclui a verificação de CPF válido, unicidade do CPF no banco e outras validações específicas.
2. **Validação de Dados:** Implementa funções para validar o formato do CPF e realiza verificações adicionais, como a idade do usuário, a presença de uma letra maiúscula e um número na senha, e o comprimento máximo do nome.
3. **Voltar:** O botão "Voltar" permite retornar à tela de login.

Integração com Banco de Dados:

1. Utiliza um objeto `java.sql.Connection` para estabelecer uma conexão com o banco de dados.
2. Realiza consultas para verificar a unicidade do CPF no banco antes de efetuar o cadastro.
3. Inclui código para inserir os dados no banco de dados, tratando exceções e registrando mensagens de erro.

Gestão de Eventos:

1. Usa eventos de ação para capturar interações do usuário, como clicar em botões.
2. Os métodos `jButton3ActionPerformed` e `jButton1ActionPerformed` são responsáveis por lidar com os eventos de voltar e cadastrar, respectivamente.

Validação Específica:

1. **CPF:** Utiliza um algoritmo de validação para verificar se o CPF é válido e se já está cadastrado no banco de dados.

2. **Senha:** Verifica se a senha atende aos critérios mínimos estabelecidos, como conter pelo menos uma letra maiúscula e um número.
3. **Nome:** Limita o comprimento máximo do nome para 100 caracteres.

Fluxo de Execução Principal:

1. O usuário preenche as informações nos campos da interface.
2. Ao clicar em "Cadastrar", as informações são validadas.
3. Se as informações são válidas, o sistema tenta inserir os dados no banco de dados.
4. Mensagens de sucesso ou erro são exibidas conforme a situação.
5. A conexão com o banco de dados é aberta no método de cadastro e fechada no bloco `finally` para garantir a liberação dos recursos, mesmo em caso de exceção.

Conteudos.java:

Descrição Geral: A classe `Conteudos` é uma representação da interface gráfica da aplicação. Ela parece ser uma tela principal ou um painel que exibe conteúdos, como filmes e séries, com botões de navegação e opções de pesquisa. A interface é construída usando o framework Swing do Java.

Componentes Principais:

1. **JLabel e Título:** Apresenta o título "MD Flix" e rótulos para categorias como "Filmes" e "Séries".
2. **Painéis (jPanel):** Divide a tela em seções para diferentes categorias de conteúdos (Filmes e Séries).
3. **Botões (JButton):** Permite a seleção de diferentes conteúdos. Cada botão parece estar associado a um filme ou série específica.
4. **Imagens (JLabel com ImageIcon):** Exibe imagens associadas a cada conteúdo (filmes e séries).
5. **Barra de Menu (JMenuBar):** Fornece opções de menu, como "Opções", que contém itens como "Suporte" e "Sair".
6. **Botão de Pesquisa:** Abre uma nova tela ou janela para pesquisa ao ser clicado.
7. **Ações de Botões:** Cada botão tem um evento associado que abre uma nova janela ou tela específica quando clicado.

Métodos Principais:

1. **initComponents():** Método gerado automaticamente que inicializa todos os componentes da interface, definindo propriedades, layouts e comportamentos.
2. Métodos de ação associados aos botões, como `jButton1ActionPerformed`, `jButton2ActionPerformed`, etc., que respondem aos cliques dos botões e iniciam outras telas.

Fluxo de Execução:

1. A classe `Conteudos` é instanciada, criando a interface gráfica.
2. O método `initComponents()` é chamado automaticamente para inicializar todos os componentes.
3. A aplicação aguarda a interação do usuário.
4. Quando um botão é clicado, o método de ação correspondente é chamado, iniciando uma nova tela associada ao conteúdo selecionado.

Autenticações:

Autenticacao.java: Nessa classe, foi adicionada uma tela de autenticação que solicita uma senha para desbloquear o acesso a determinado conteúdo. Abaixo estão algumas explicações sobre o código:

Campo de Senha: Há um campo de senha (`txtBloqueio`) onde o usuário deve inserir a senha para desbloquear o conteúdo.

Botões: Existem dois botões na tela:

Acessar: Verifica se a senha inserida é correta. Se sim, abre a tela `Tons50`, que contém o conteúdo bloqueado. Se não, exibe uma mensagem de aviso.

Voltar: Retorna à tela principal (`TelaPrincipal`).

Lógica de Autenticação: A lógica de autenticação é simples. Se a senha inserida (`new String(txtBloqueio.getPassword())`) for igual a "123", então o acesso é permitido, e a tela `Tons50` é exibida. Caso contrário, uma caixa de diálogo com a mensagem "Você não pode acessar esse conteúdo!" é exibida.

Fluxo Principal: O método `main` inicia a aplicação, exibindo a tela de autenticação (`Autenticacao`).

AutenticacaoSerie.java: Nesta classe, foi implementada uma tela de autenticação para desbloquear o acesso a um conteúdo específico de série. Aqui estão algumas explicações sobre o código:

Campo de Senha: Existe um campo de senha (`SerieLiberar`) onde o usuário deve inserir a senha para liberar o conteúdo da série.

Botões: Há dois botões na tela:

Acessar: Verifica se a senha inserida é correta. Se sim, abre a tela `SexLife`, que contém o conteúdo da série. Se não, exibe uma mensagem de aviso.

Voltar: Retorna à tela principal (`TelaPrincipal`).

Lógica de Autenticação: A lógica de autenticação é semelhante à anterior. Se a senha inserida (`new String(SerieLiberar.getPassword())`) for igual a "123", então o acesso é permitido, e a tela `SexLife` é exibida. Caso contrário, uma caixa de diálogo com a mensagem "Você não pode acessar esse conteúdo!" é exibida.

Fluxo Principal: O método `main` inicia a aplicação, exibindo a tela de autenticação para séries (`AutenticacaoSerie`).

Pesquisa.java:

Essa classe Java representa a tela de pesquisa no sistema. Aqui estão algumas informações relevantes sobre o que esta classe realiza:

Layout e Componentes Visuais: A classe `Pesquisa` é uma extensão de `javax.swing.JFrame`, indicando que representa uma janela gráfica. O layout inclui um campo de texto para inserir a pesquisa, um botão para iniciar a pesquisa, e um botão para voltar à tela principal.

Método de Pesquisa: O método `jButton1ActionPerformed` é acionado quando o botão "Procurar" é clicado. Ele verifica o conteúdo do campo de pesquisa (`pesquisar`) e, com base na entrada, decide qual tela relacionada abrir. Se a entrada corresponder a um filme ou série conhecido (por exemplo, "Homem de Ferro" ou "Jogos Vorazes"), a aplicação abrirá a tela correspondente. Se não houver correspondência, exibirá uma mensagem de aviso.

Voltar à Tela Principal: O botão "Voltar" (`jButton2`) permite que o usuário retorne à tela principal do sistema "MD FLIX".

Aviso de Filme Não Encontrado: Se a pesquisa não corresponder a nenhum filme ou série conhecido, será exibida uma caixa de diálogo (`JOptionPane`) informando ao usuário que o filme não foi encontrado.

Entrada e Comparação: A entrada do campo de pesquisa é convertida para maiúsculas (`pesquisar.getText().toUpperCase()`) antes da comparação, garantindo que a comparação não seja sensível a maiúsculas e minúsculas.

Encerramento e Abertura de Telas: Para abrir uma nova tela, a classe atual é fechada (`this.dispose()`) e a nova tela é instanciada e tornada visível.

Em resumo, a classe `Pesquisa` é responsável por fornecer uma interface para os usuários pesquisarem filmes e séries no sistema e navegar para as páginas correspondentes com base nos resultados da pesquisa.

Suporte.java:

Atributos:

- `private Connection conec`: Representa uma conexão com o banco de dados. No entanto, a conexão é redundante, pois a classe `ConexaoSQL` já possui métodos para obter e fechar conexões. Além disso, esse atributo não é usado.

2. **Método Construtor:**

- `Suporte()`: O construtor da classe. Inicializa a interface gráfica (componentes visuais) definida no método `initComponents()`.

3. **Métodos:**

- `initComponents()`: Este método é gerado automaticamente pelo Editor de Formulários do NetBeans. Ele inicializa e organiza os componentes visuais na interface gráfica.
- `jButton2ActionPerformed(ActionEvent evt)`: Este método é chamado quando o botão "Voltar" é pressionado. Ele cria uma instância da classe `Conteudos` e a torna visível, enquanto fecha a janela atual.
- `jButton1ActionPerformed(ActionEvent evt)`: Este método é chamado quando o botão "Enviar" é pressionado. Ele coleta o assunto e a mensagem do relatório, estabelece uma conexão com o banco de dados, e insere os dados na tabela "suporte".

4. **Método Main:**

- `main(String args[])`: O método principal que cria uma instância da classe `Suporte` e a torna visível.

Funcionamento Geral:

- O usuário preenche o assunto e a mensagem do relatório.
- Ao clicar em "Enviar", os dados são inseridos na tabela "suporte" no banco de dados.
- O código inclui manipulação de exceções para lidar com erros durante a inserção no banco de dados.
- Após o envio bem-sucedido, uma mensagem de sucesso é exibida, e os campos são limpos.

Cursed.java:

Essa classe Java representa a interface gráfica para a série "Cursed - A Lenda do Lago" no sistema. Aqui estão algumas informações relevantes sobre o código:
Layout e Componentes Visuais: A classe `Cursed` é uma extensão de `javax.swing.JFrame`, indicando que representa uma janela gráfica. Ela exibe informações sobre a série, como título, imagem, ano de lançamento, gênero, número de temporadas, diretores e sinopse.
Informações da Série:

- **Título:** Cursed - A Lenda do Lago
- **Ano de Lançamento:** 2020
- **Gênero:** Fantasia, Aventura, Drama
- **Temporadas:** 1 temporada
- **Diretores:** Frank Miller, Tom Wheeler
- **Sinopse:** A história se passa em um mundo medieval reimaginado e segue Nimue, uma jovem com um misterioso dom, destinada a se tornar a Dama do Lago.

Componentes Visuais:

Botão "Assistir": Ao clicar neste botão (`jButton1`), o código instancia a classe `CursedTemp` (possivelmente uma classe que representa a reprodução de episódios) e a torna visível.
Botão "Voltar": Ao clicar neste botão (`jButton2`), o código instancia a classe `TelaPrincipal` e a torna visível, enquanto fecha a janela atual.
CheckBox "Favoritar": Permite ao usuário marcar a série como favorita.
Imagem da Série: A imagem da série é exibida usando um componente `jLabel2` com um ícone.

CursedTemp.java:

Esta classe Java, chamada `CursedTemp`, representa a interface gráfica para os episódios da série "Cursed - A Lenda do Lago". Aqui estão algumas informações relevantes sobre o código:

Layout e Componentes Visuais: A classe `CursedTemp` é uma extensão de `javax.swing.JFrame`, indicando que representa uma janela gráfica.
A janela exibe uma lista de episódios da "Temporada 1" da série "Cursed - A Lenda do Lago".

Episódios da Série: A lista de episódios é exibida em um painel (`jPanel2`), com cada episódio representado por um botão. Os episódios são numerados de 1 a 10.

Botão "Voltar": Ao clicar no botão "Voltar" (`jButton11`), o código instancia a classe `Cursed` e a torna visível, enquanto fecha a janela atual.

Título da Janela: O título da janela é "CURSED - A LENDA DO LAGO" (`jLabel1`).

HomemFerro.java:

Esta classe Java, chamada `HomemFerro`, representa a interface gráfica para o filme "Homem de Ferro". Aqui estão algumas informações relevantes sobre o código:

Layout e Componentes Visuais: A classe `HomemFerro` é uma extensão de `javax.swing.JFrame`, indicando que representa uma janela gráfica.
A janela exibe uma imagem (`jLabel1`) correspondente ao filme "Homem de Ferro". Botões e rótulos são utilizados para mostrar informações sobre o filme, como título, ano de lançamento, diretor, gênero e sinopse.

Botões e Ações Associadas: Um botão "Assistir" (`jButton1`) é apresentado, mas a ação associada ainda não foi implementada (`jButton1ActionPerformed`).
Um botão "Voltar" (`jButton2`) é fornecido, e ao ser clicado, ele instancia a classe `TelaPrincipal` e a torna visível, enquanto fecha a janela atual.

Favoritar: Um `JCheckBox` (`jCheckBox1`) permite ao usuário marcar ou desmarcar a opção de "Favoritar".

JogosVorazes.java:

Esta classe Java, chamada `JogosVorazes`, representa a interface gráfica para o filme "Jogos Vorazes". Aqui estão algumas informações relevantes sobre o código:

Layout e Componentes Visuais: A classe `JogosVorazes` é uma extensão de `javax.swing.JFrame`, indicando que representa uma janela gráfica. A janela exibe uma imagem (`jLabel2`) correspondente ao filme "Jogos Vorazes". Botões e rótulos são utilizados para mostrar informações sobre o filme, como título, ano de lançamento, diretor, gênero, sinopse e duração.

Botões e Ações Associadas: Um botão "Assistir" (`jButton1`) é apresentado, mas a ação associada ainda não foi implementada (`jButton1ActionPerformed`). Um botão "Voltar" (`jButton2`) é fornecido, e ao ser clicado, ele instancia a classe `TelaPrincipal` e a torna visível, enquanto fecha a janela atual.

Informações do Filme: Rótulos (`jLabel3` a `jLabel9`) são utilizados para exibir informações como título, ano de lançamento, diretor, gênero, sinopse e duração do filme "Jogos Vorazes".

Favoritar: Um `JCheckBox` (`jCheckBox1`) permite ao usuário marcar ou desmarcar a opção de "Favoritar". No entanto, não há lógica associada a essa opção no código fornecido.

SexLife.java:

A classe Java chamada `SexLife` representa a interface gráfica para a série de TV "SexLife". Abaixo estão algumas informações importantes sobre o código:
A classe `SexLife` é uma extensão de `javax.swing.JFrame`, indicando que ela representa uma janela gráfica.
A janela exibe uma imagem (`jLabel2`) correspondente à série "SexLife". Botões e rótulos são utilizados para mostrar informações sobre a série, como título, ano de lançamento, diretor, gênero, número de temporadas, sinopse, etc.
Botões e Ações Associadas: Um botão "Assistir" (`jButton1`) é apresentado. A ação associada a esse botão (`jButton1ActionPerformed`) instância e torna visível uma nova janela da classe `SexTemp`, que contém os episódios da série referida.
Um botão "Voltar" (`jButton3`) é fornecido. Ao ser clicado, ele instancia a classe `TelaPrincipal` e a torna visível, enquanto fecha a janela atual.
Um `JCheckBox` (`jCheckBox1`) permite ao usuário marcar ou desmarcar a opção de "Favoritar".
Fluxo Principal: O método `main` inicia a aplicação, exibindo a janela dos episódios da série (`SexLife`).

SexTemp.java:

Esta classe Java representa uma interface gráfica para os episódios de uma série chamada "Sex Life". Aqui estão algumas explicações sobre o código:

Painéis e Botões: Existem dois painéis, `jPanel2` para a Temporada 2 e `jPanel3` para a Temporada 1. Cada painel contém botões representando os episódios de cada temporada (por exemplo, `jButton1` a `jButton8` para a Temporada 1 e `jButton9` a `jButton14` para a Temporada 2).

Ação do Botão "Voltar": Existe um botão chamado `jButton15` com a ação de voltar. Quando pressionado, ele fecha a janela atual (`SexTemp`) e abre a tela principal (`SexLife`).

Fluxo Principal: O método `main` inicia a aplicação, exibindo a janela dos episódios da série (`SexTemp`).

The100.java:

Esta classe Java representa a tela de detalhes de uma série chamada "The 100". Vamos analisar algumas partes específicas do código:

Labels e Imagem: Existem várias labels (`JLabel`) que exibem informações sobre a série, como título, imagem, ano de estreia, número de temporadas, diretores, sinopse, etc. A imagem é exibida usando um `JLabel` com um ícone (`ImageIcon`) carregado de um recurso no caminho "/images/the 100.jpeg".

Botões: Há dois botões, `jButton2` e `jButton9`, que têm ações associadas. `jButton2` é para "Assistir" e, quando clicado, abre uma nova janela (`The100Temp`) e fecha a janela atual (`The100`). `jButton9` é para "Voltar" e, quando clicado, retorna à tela principal (`TelaPrincipal`).

Checkbox: Existe uma `JCheckBox` chamada `jCheckBox1` com a opção "Favoritar". Os usuários podem marcar ou desmarcar esta opção.

Ação dos Botões: As ações dos botões estão implementadas nos métodos `jButton2ActionPerformed` e `jButton9ActionPerformed`. No primeiro, uma nova janela é aberta; no segundo, a tela principal é exibida.

Fluxo Principal: O método `main` inicia a aplicação, exibindo a janela de detalhes da série (`The100`).

The100Temp.java:

Esta classe Java representa uma interface gráfica para os episódios de uma série chamada "The 100". Aqui estão algumas explicações sobre o código:

Estrutura Geral: O código está dividido em vários métodos, onde cada método realiza uma função específica. A estrutura geral inclui:

- main **Método Principal:**
    - Configura o look and feel do Swing para "Nimbus".
    - Cria uma instância da classe `The100Temp` (que representa a janela principal) e a torna visível.

Organização da Interface Gráfica:

- Há vários painéis (`jPanel2`, `jPanel3`, ..., `jPanel8`) para cada temporada, organizados horizontalmente na interface principal (`jPanel1`).
- Cada painel contém um rótulo (`jLabelX`) indicando a temporada e uma série de botões representando os episódios dessa temporada.
- O botão "Voltar" (`jButton101`) está localizado no final da interface.

Temporadas (`jPanel2`, `jPanel3`, ..., `jPanel8`):

Cada temporada é representada por um painel (`JPanel`). Cada um desses painéis contém um rótulo indicando a temporada e botões representando episódios. **Configuração dos Painéis de Temporadas (**jPanel2**,** jPanel3**, ...,** jPanel8**):**

- Define a cor de fundo de cada painel.
- Configura um rótulo indicando o número da temporada.
- Adiciona botões representando episódios para cada temporada.

Botão "Voltar" (`jButton101`):
jButton101ActionPerformed **- Ação do Botão "Voltar":**

- Cria uma nova instância da classe `The100`.
- Torna a nova instância visível.
- Fecha a janela atual (`this.dispose()`).

Em todas os painéis de temporadas, os botões são organizados verticalmente usando o layout `GroupLayout`.

Tons50.java:

Esta classe representa a tela de detalhes do filme "Cinquenta Tons de Cinza". Vamos analisar algumas partes específicas do código:

Labels e Imagem: Existem várias labels (`JLabel`) que exibem informações sobre o filme, como título, imagem, ano de lançamento, diretor, gênero, sinopse, duração, etc. A imagem é exibida usando um `JLabel` com um ícone (`ImageIcon`) carregado de um recurso no caminho "/images/50Tons.jpeg".

Botões: Há dois botões, `jButton1` e `jButton2`, que têm ações associadas. `jButton1` é para "Assistir". `jButton2` é para "Voltar" e, quando clicado, retorna à tela principal (`TelaPrincipal`).

Checkbox: Existe uma `JCheckBox` chamada `jCheckBox1` com a opção "Favoritar". Os usuários podem marcar ou desmarcar esta opção.

Ação dos Botões: As ações dos botões estão implementadas nos métodos `jButton1ActionPerformed` e `jButton2ActionPerformed`. O primeiro pode ser implementado para abrir uma nova janela ou executar a ação de assistir. O segundo retorna à tela principal.

Fluxo Principal: O método `main` inicia a aplicação, exibindo a janela de detalhes do filme (`Tons50`).

Conexão com o Banco de dados MySQL

1. **Importações:**

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

Importa as classes necessárias para manipular exceções (SQLException), conexões (Connection), e carregar drivers JDBC (DriverManager).

2. **Atributos Estáticos:**

```
private static final String URL = "jdbc:mysql://localhost:3306/streeming";
private static final String USUARIO = "root";
private static final String SENHA = "";
```

URL: A URL de conexão do banco de dados MySQL.

- **USUARIO:** Nome de usuário do banco de dados.
- **SENHA:** Senha do banco de dados.

3. **Método getCon():**

```
public static Connection getCon() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        return DriverManager.getConnection(URL, USUARIO, SENHA);
    } catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace();
        return null;
    }
}
```

Objetivo: Obter uma conexão com o banco de dados.

- **Passos:**
  - Tenta carregar o driver JDBC do MySQL usando `Class.forName`.
  - Estabelece uma conexão com o banco de dados utilizando `DriverManager.getConnection`.

- **Exceções Tratadas:**
    - `ClassNotFoundException`: Se o driver não for encontrado.
    - `SQLException`: Se houver problemas na conexão.

4. **Método closeCon():**

```
public static void closeCon(Connection conexao) {
    if (conexao != null) {
        try {
            conexao.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Objetivo: Fechar uma conexão com o banco de dados.

- **Passos:**
    - Verifica se a conexão não é nula.
    - Fecha a conexão utilizando `conexao.close()`.
- **Exceções Tratadas:**
    - `SQLException`: Se ocorrer um problema ao fechar a conexão.
    -

Informações Adicionais:

- O código utiliza o padrão Singleton, fornecendo métodos estáticos para obter e fechar conexões.
- É uma abordagem básica e, geralmente, é recomendado o uso de recursos como try-with-resources para garantir o fechamento adequado de conexões.

Demais Implementações:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import javax.swing.JoptionPane;
import javax.swing.*;
import java.awt.event.ActionEvent;
```