

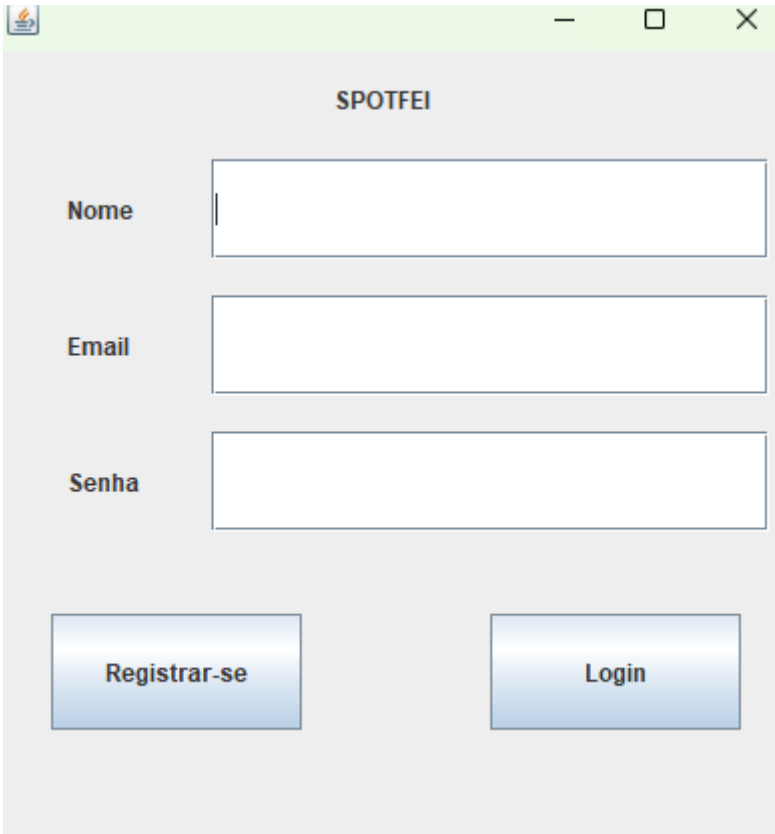
Relatório Técnico - SpotiFei

Objetivo: O objetivo do projeto Spotifei é construir uma plataforma de informações de áudios digitais, como músicas e podcasts

- Você pode se inspirar na Plataforma de Streaming de Áudios Digitais Spotify, pois a lógica de compartilhamento de informações é similar.
 - Observação: o projeto Spotifei é apenas para compartilhamento de informações sobre músicas, não é preciso reproduzir as músicas.
 - Tecnologias que devem ser usadas no desenvolvimento do projeto: Swing (Java) + JDBC PostgreSQL + MVC (Model, View, Controller)
-

Funcionalidades do Usuário

- Cadastrar novo usuário
- Login de usuário



A imagem mostra uma janela de aplicativo com o título "SPOTFEI". Dentro da janela, há três campos de entrada de texto rotulados "Nome", "Email" e "Senha". Abaixo dos campos, há dois botões: "Registrar-se" e "Login".

No arquivo UsuarioDAO é onde contém a maior parte dos meus códigos, pois eu ao criar meu projeto tive na cabeça o seguinte: cada música curtida, playlist, discutida e etc é para apenas UM usuário, logo tudo tem que ser salvo dentro desse usuário e não em um contexto geral, por isso tudo menos as músicas que aí sim são iguais para todos os usuários são cadastrados e salvos dentro do UsuarioDAO.

- Buscar músicas por nome, artista ou gênero

- A função de buscar musica por nome artista ou gênero nada mais é do que uma busca em SQL dentro do java, não é algo nem um pouco complicado se tiver os conhecimentos certos.

“””

```
public List<Musica> buscarMusicas(String termoBusca) {  
    List<Musica> musicas = new ArrayList<>();  
  
    String sql = "SELECT * FROM musica WHERE LOWER(nome) LIKE ? OR LOWER(genero)  
    LIKE ? OR artista_id = ?";  
  
    try (Connection conn = Conexao.getConexao();  
        PreparedStatement stmt = conn.prepareStatement(sql)) {  
  
        String busca = "%" + termoBusca.toLowerCase() + "%";  
        stmt.setString(1, busca);  
        stmt.setString(2, busca);  
  
        try {  
            int artistald = Integer.parseInt(termoBusca);  
            stmt.setInt(3, artistald);  
        } catch (NumberFormatException e) {  
            stmt.setInt(3, -1); // Nenhum artista terá ID -1, evita erros na consulta  
        }  
  
        ResultSet rs = stmt.executeQuery();  
  
        while (rs.next()) {  
            int id = rs.getInt("id");  
            String nome = rs.getString("nome");  
            String genero = rs.getString("genero");  
            int artista_id = rs.getInt("artista_id");
```

```

        Musica musica = new Musica(id, nome, genero, artista_id);

        musicas.add(musica);
    }

} catch (SQLException e) {

    System.out.println("Erro ao buscar músicas: " + e.getMessage());

}

return musicas;

}

"""

```

- Listar músicas buscadas

- Uma tabela de músicas feitas pelo ChatGPT com 20 músicas aleatórias de diferentes gêneros foi inserida dentro do banco de dados para permitir o uso do projeto, as músicas ficam sempre aparentes na tabela do projeto, mas podem ser filtradas por nome ou gênero. Todas as buscas feitas são salvas do banco de dados para poder depois haver a visualização do histórico das últimas 10 pesquisas.

```

"""

public void registrarBusca(int idUsuario, String termo) {

    String sql = "INSERT INTO historico_buscas (id_usuario, texto_busca) VALUES (?, ?)";

    try (Connection conn = Conexao.getConexao();

        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setInt(1, idUsuario);

        stmt.setString(2, termo);

        stmt.executeUpdate();

    } catch (SQLException e) {

        e.printStackTrace();

    }

}

}

"""

```

- Curtir e descurtir músicas

- Nas funções de curtir e discutir música eu fiz um sistema simples de procura no banco de dados para retirar a música de um caso tenha e colocar no outro, por exemplo se eu tenho uma música discutida e curto ela logo em seguida, o sistema vai na tabela do banco de dados dos discutido e remove a musica de la, e adiciona na tabela das curtidas logo em seguida.

“””

```
public void curtirMusica(int usuariold, int musicald) {

    String deleteSql = "DELETE FROM descurtidas WHERE id_usuario = ? AND id_musica = ?";

    String insertSql = "INSERT INTO curtidas (id_usuario, id_musica) VALUES (?, ?)";

    try (Connection conn = Conexao.getConexao()) {

        conn.setAutoCommit(false);

        try (PreparedStatement deleteStmt = conn.prepareStatement(deleteSql);

            PreparedStatement insertStmt = conn.prepareStatement(insertSql)) {

            deleteStmt.setInt(1, usuariold);

            deleteStmt.setInt(2, musicald);

            deleteStmt.executeUpdate();

            insertStmt.setInt(1, usuariold);

            insertStmt.setInt(2, musicald);

            insertStmt.executeUpdate();

            conn.commit();

            System.out.println("Música curtida com sucesso!");

        } catch (SQLException e) {

            conn.rollback();

            System.out.println("Erro ao curtir música: " + e.getMessage());

        } finally {

            conn.setAutoCommit(true);

        }

    } catch (SQLException e) {

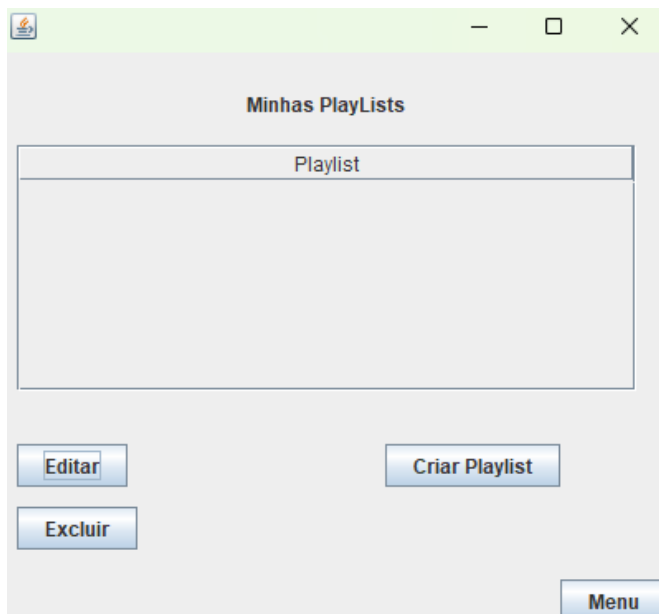
        System.out.println("Erro na conexão: " + e.getMessage());

    }
```

```
}  
  
}  
“””
```

- Gerenciar playlists

- A parte de visualização da playlist provavelmente é a parte mais difícil desse projeto, pois são muitas ações dentro e fora de uma playlist que geraram vários erros durante a execução do projeto, tem que ter as opções de ver, editar, excluir, mudar nome, adicionar música e remover música, são várias coisas relativamente simples, mas que requer muita atenção na execução.



- Visualizar histórico

- Todas as buscas são salvas no banco de dados, mas na visualização só aparecem as 10 últimas, e em relação a curtidas e discutidas aparecem todas que estão salvas no banco de dados do usuário logado.

