JPA Relacionamento entre Entidades

Alícia Schiochet Souza[1], Thiago Schulz da Rosa[2].

Universidade da Região de Joinville (UNIVILLE)

89.219-710–Joinville– SC – Brazil.

alicia.souza@univille.br [1]; thiago.rosa23@univille.b[2].

1. Introdução

Com base no renomado software de música Spotify, criamos o MusicStream, uma plataforma de streaming de música inovadora e acessível. O MusicStream permite aos usuários ouvirem músicas sob demanda, criar e gerenciar playlists personalizadas, e explorar uma vasta biblioteca musical. O sistema foi projetado para oferecer uma experiência de usuário intuitiva e eficiente, garantindo que todos possam desfrutar de suas músicas favoritas de maneira simples e prática.

O software MusicStream oferece uma gama completa de funcionalidades, incluindo:

- Cadastro e Login: Os usuários podem facilmente se cadastrar e fazer login para acessar suas contas.
- Criação de Playlists: Permite que os usuários criem, editem e organizem suas playlists de acordo com suas preferências musicais.
- Reprodução de Músicas: Os usuários podem reproduzir músicas sob demanda, com controles de reprodução intuitivos como play, pause e skip.

2. Principais Requisitos Funcionais

RF_01: Cadastro e Login (Autenticação de usuários para acesso ao sistema.).

RF_02: Pesquisa de músicas (Pesquisa de músicas, álbuns e artistas.).

RF_03: Criação e gerenciamento de playlists (Criação de playlists personalizadas. Adição e remoção de músicas nas playlists.).

RF_04: Reproduzir músicas (Reprodução de músicas sob demanda.).

2.1. Histórias dos Usuários Ana e Lucas

Ana, uma nova usuária do sistema de música, eu quero me cadastrar e personalizar a minha foto com meu e-mail e senha para criar uma conta e acessar o sistema.

Lucas, um usuário existente, eu quero fazer login com meu e-mail e senha para acessar minha conta e minhas playlists.

RF_01: Cadastro e Login (Autenticação de usuários para acesso ao sistema.)

O sistema deve permitir que usuários se cadastrem e façam login para acessar suas contas e recursos.

A Figura Campos do usuário: Nome e Sobrenome. Email e senha para autenticação. ID de usuário (gerado automaticamente).

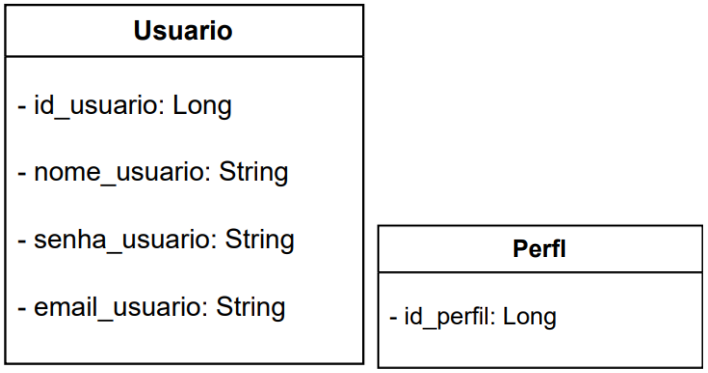

Figura 01 diagrama de classe da história dos usuários 01.

2.2 História de Usuário João

João, um amante de música, eu quero pesquisar por músicas, álbuns e artistas, para encontrar rapidamente minhas canções favoritas.

RF_02: Pesquisa de músicas (Pesquisa de músicas, álbuns e artistas.)

O sistema deve permitir que o usuário pesquise músicas, álbuns e artistas, proporcionando uma experiência de busca eficiente.

Campos de pesquisa: Título da música, Nome do álbum ou Nome do artista.

Relações com outras tabelas:

Músicas e Álbuns: A pesquisa do usuário se relaciona com as tabelas de Músicas e Álbuns, permitindo a busca por ID da música ou álbum.

Artistas: A busca pelo nome do artista deve permitir a listagem de todos os conteúdos relacionados ao artista.

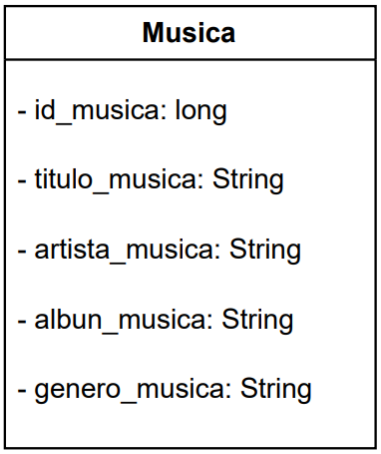

Figura 02 diagrama de classe da história do usuário 03.

2.3 Histórias dos Usuários Pedro e Beatriz

Pedro, um usuário frequente, eu quero criar uma playlist personalizada, para organizar minhas músicas preferidas em categorias.

Beatriz, eu quero adicionar e remover músicas de uma playlist existente, para atualizar as playlists conforme meu gosto musical muda.

RF_03: Criação e gerenciamento de playlists (Criação de playlists personalizadas. Adição e remoção de músicas nas playlists.)

O sistema deve permitir que o usuário crie, edite, visualize e exclua playlists, além de adicionar e remover músicas dessas playlists.

Campos da Playlist: ID da playlist (gerado automaticamente). Nome da playlist. Lista de músicas adicionadas à playlist.

Relações com outras tabelas:

Músicas: O usuário poderá adicionar músicas à playlist, criando uma relação de muitos-para-muitos entre a Playlist e as Músicas.

Usuário: Cada playlist será vinculada a um único usuário, formando uma relação de um-para-muitos entre o Usuário e a Playlist.

Histórico de Modificação da Playlist: O sistema deve registrar cada modificação feita na playlist (adição ou remoção de músicas), criando uma relação de um-para-muitos entre o Usuário e o Histórico de Modificações

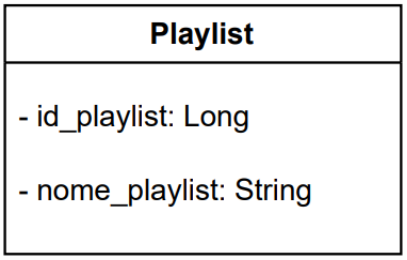

Figura 03 diagrama de classe da história do usuário 04.

3. Codificação

```
1.  @Data
2.  @NoArgsConstructor
3.  @Entity
4.
5.  public class Perfil{
6.      @Id
7.      @GeneratedValue(strategy = GenerationType.IDENTITY)
8.      private long id_perfil;
9.      @OneToOne
10.         @JoinColumn(name = "usuario_id")
11.         private Usuario usuario;
12.     }
```

Figura 1: Código da entidade “Perfil”.

```
1.   @Data
2.   @NoArgsConstructor
3.   @Entity
4.
5.   public class Usuario{
6.       @Id
7.       @GeneratedValue(strategy = GenerationType.IDENTITY)
8.       private long id_usuario;
9.
10.      private String nome_usuario;
11.      private String email_usuario;
12.      private String senha_usuario;
13.
14.      @OneToMany(mappedBy = "usuario", cascade = CascadeType.ALL)
15.      private List<Playlist> playlists;
16.  }
```

Figura 2: Código da entidade “Usuario”.

```
1.  @Data
2.  @NoArgsConstructor
3.  @Entity
4.
5.  public class Musica{
6.      @Id
7.      @GeneratedValue(strategy = GenerationType.IDENTITY)
8.      private long id_musica;
9.
10.     private String titulo_musica;
11.     private String artista_musica;
12.     private String genero_musica;
13.     private String album_musica;
14.
15.     @ManyToMany(mappedBy = "musicas")
16.     private List<Playlist> playlists;
17. }
```

Figura 3: Código da entidade “Musica”.

```
1.  @Data
2.  @NoArgsConstructor
3.  @Entity
4.
5.  public class Playlist {
6.      @Id
7.      @GeneratedValue(strategy = GenerationType.IDENTITY)
8.      private long idPlaylist;
9.      private String nomePlaylist;
10.
11.     @ManyToOne
12.     @JoinColumn(name = "usuario_id")
13.     private Usuario usuario;
14.
15.     @ManyToMany
16.     @JoinTable(
17.             name = "playlist_songs",
18.             joinColumns = @JoinColumn(name = "playlist_id"),
19.             inverseJoinColumns = @JoinColumn(name = "song_id"))
20.     private List<Musica> musicas;
21. }
```

Figura 4: Código da entidade “Playlist”

3.1. Entidade XOP

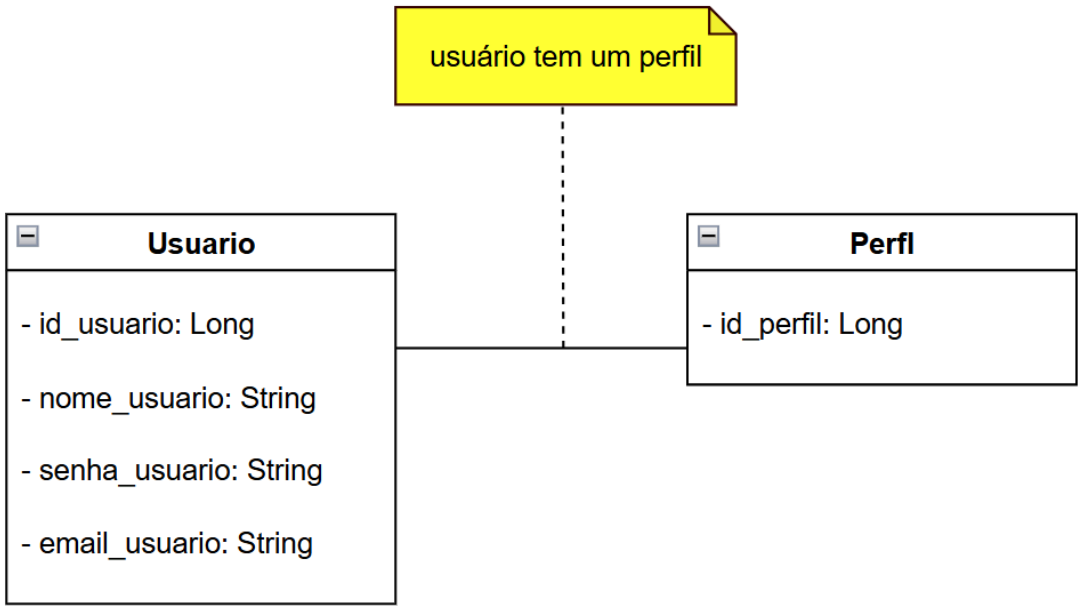

Figura 1.1- Diagrama de classe relacionamento usuário com perfil.

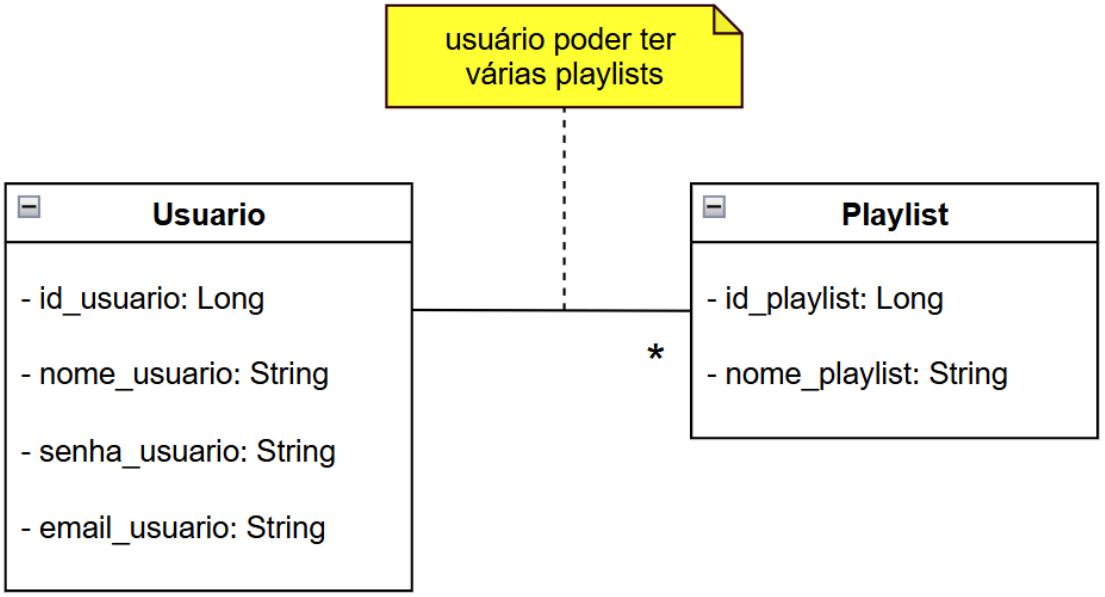

Figura 1.2- Diagrama de classe relacionamento usuário com Playlist

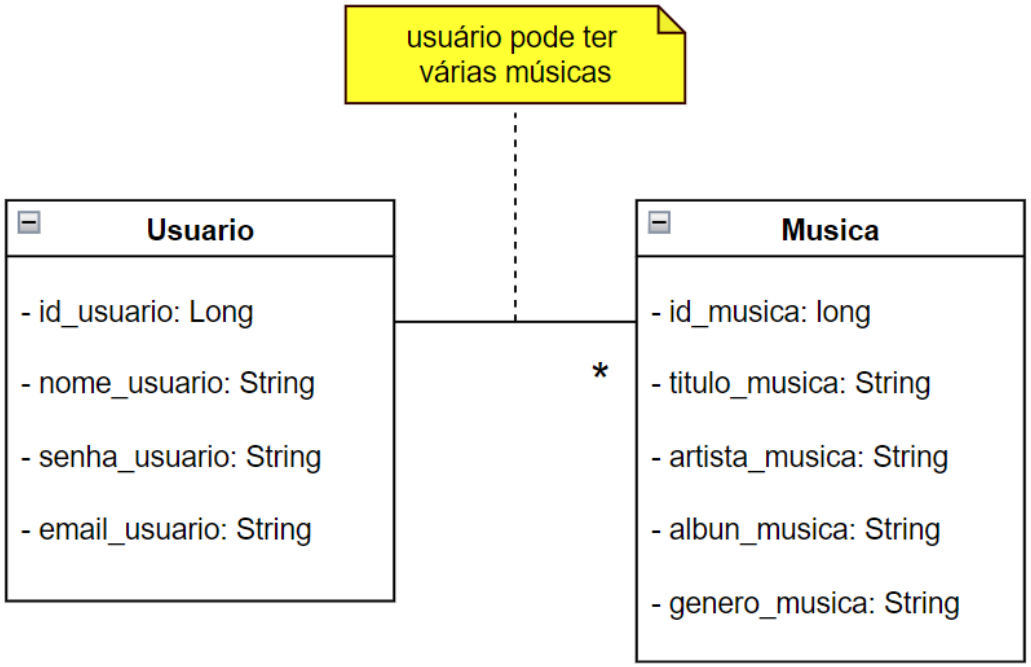

Figura 1.3- Diagrama de classe relacionamento usuário com música.

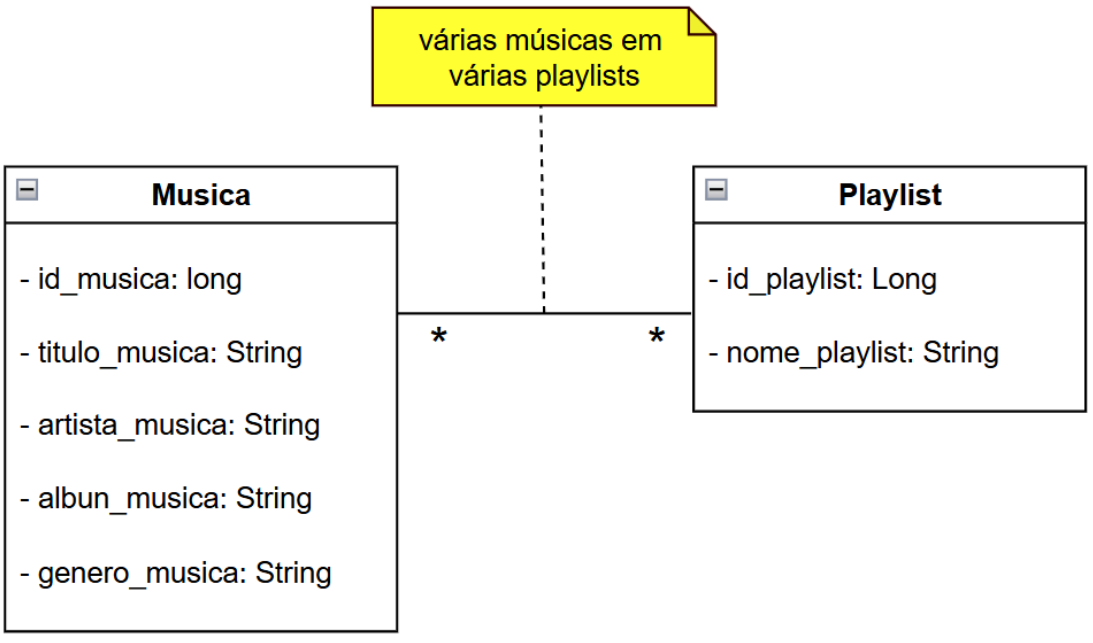

Figura 1.4- Diagrama de classe relacionamento musica com playlists.

4. Banco de dados

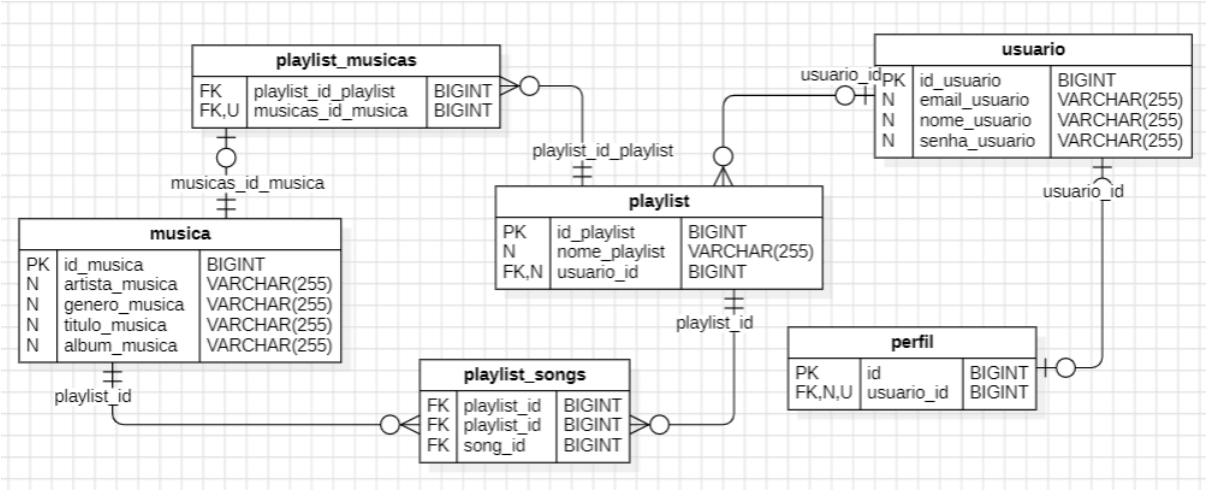

Figura 4. Modelo Entidade Relacionamento do Sistema de Música

4. Conclusão

Este sistema foi projetado para oferecer uma experiência de usuário intuitiva e eficiente, permitindo que os usuários desfrutem de suas músicas favoritas de maneira simples e prática. Com funcionalidades robustas como cadastro e login de usuários, criação e gerenciamento de playlists. O MusicStream se posiciona como uma solução completa para amantes da música. A implementação de um sistema de recomendação personalizado também garante que os usuários possam descobrir novas músicas e artistas, enriquecendo ainda mais sua experiência musical.

A estrutura do projeto, baseada em Java e interligada a um banco de dados relacional, assegura a escalabilidade e a eficiência do sistema. O uso de técnicas de segurança para armazenamento de senhas e a validação de dados garantem a proteção das informações dos usuários.

Em resumo, o MusicStream não só atende às necessidades básicas de um serviço de streaming de música, mas também proporciona uma plataforma rica e personalizada, pronta para ser expandida e melhorada conforme as demandas dos usuários evoluem. Este projeto demonstra a capacidade de integrar tecnologia e música de forma harmoniosa, criando uma experiência agradável e envolvente para todos os usuários.