Sistema de Música Spotify

José Willian da Silva Alves, Lucas Zultanski de Almeida

Engenharia de Software/Sistemas de Informação
Universidade da Região de Joinville (UNIVILLE) – Joinville, SC – Brazil
Jose.alves23@univille.br, lucas.zultanski23@univille.br

1. Introdução

O projeto em questão visa o desenvolvimento de um sistema de música semelhante ao Spotify, utilizando o framework Spring Boot e o banco de dados MySQL. Este sistema permitirá aos usuários se cadastrarem, fazer login, criar playlists personalizadas, buscar músicas, e gerenciar álbuns e músicas. Artistas também poderão cadastrar novos álbuns e faixas, que ficarão disponíveis para outros usuários.

Este projeto é ideal para estudantes que estão em estágio inicial de aprendizado em Java, Spring Boot, e bancos de dados relacionais, fornecendo uma compreensão prática de como desenvolver APIs RESTful, gerenciar dados usando JPA e configurar um banco de dados MySQL em conjunto com a aplicação Spring.

2. Requisitos Funcionais

 1. O sistema deve permitir o cadastro de novos usuários com nome, e-mail e senha.

 2. O sistema deve permitir que usuários cadastrados façam login com e-mail e senha.

 3. O usuário autenticado deve poder criar playlists e nomeá-las.

 4. O sistema deve permitir ao usuário buscar músicas, álbuns e artistas pelo nome.

 5. O usuário deve poder adicionar músicas às suas playlists.

 6. O usuário deve poder remover músicas de suas playlists.

 7. O sistema deve permitir que artistas autenticados cadastrarem álbuns e músicas.

 8. O usuário deve poder seguir artistas e receber atualizações sobre novos lançamentos.

 9. O usuário deve poder excluir playlists de sua conta.

2.1. Cadastro de Usuário

Como um usuário, quero poder me cadastrar no sistema, fornecendo meu nome, e-mail e senha, para que eu possa acessar minha conta e criar playlists personalizadas.

2.2. Login de Usuário

Como um usuário cadastrado, quero fazer login no sistema utilizando meu e-mail e senha, para acessar minhas playlists e músicas favoritas.

2.3. Criação de Playlists

Como um usuário autenticado, quero criar e nomear playlists para organizar minhas músicas favoritas.

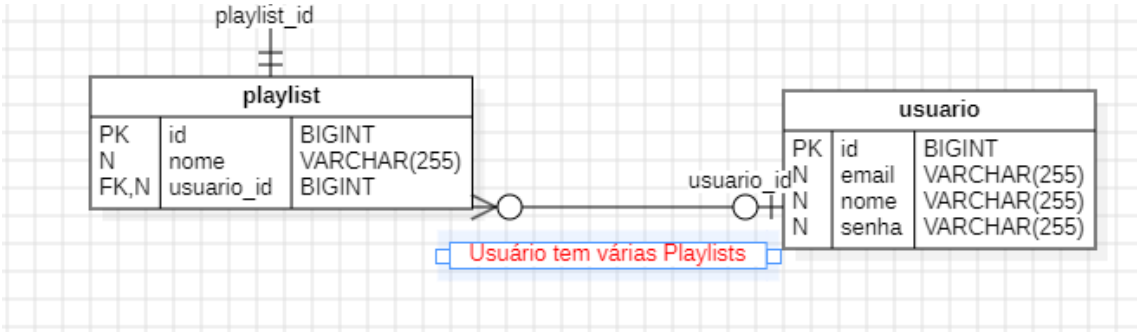

Figura 1. Diagrama de classe das entidades da Criação de Playlists.

2.4. Busca por Músicas e Álbuns

Como um usuário, quero buscar músicas e álbuns no sistema para encontrar faixas e artistas de meu interesse.

2.5. Adição de Músicas à Playlist

Como um usuário autenticado, quero adicionar músicas às minhas playlists para personalizar minha experiência musical.

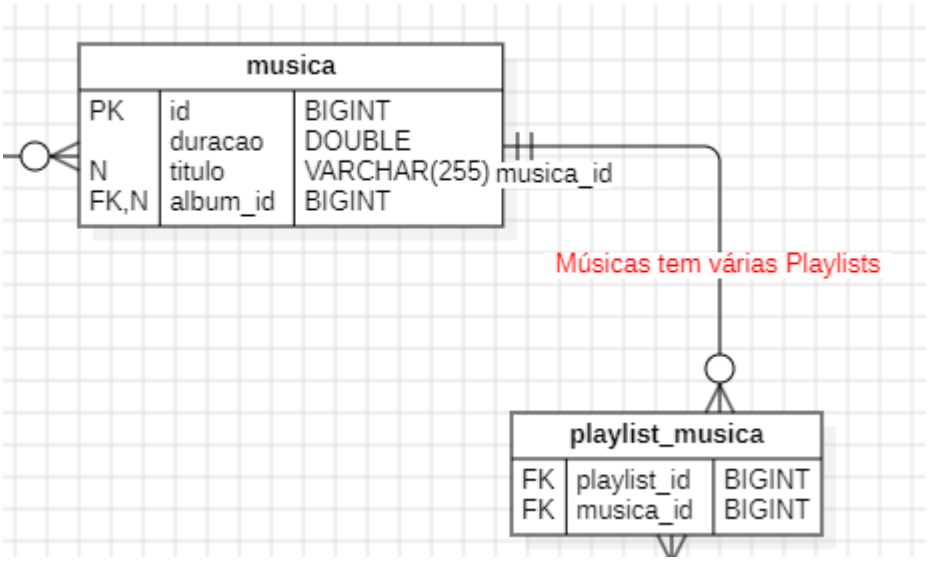

Figura 1. Diagrama de classe das entidades da Adição de Músicas à Playlist.

2.6. Remoção de Músicas da Playlist

Como um usuário autenticado, quero remover músicas das minhas playlists para organizar minha lista de faixas.

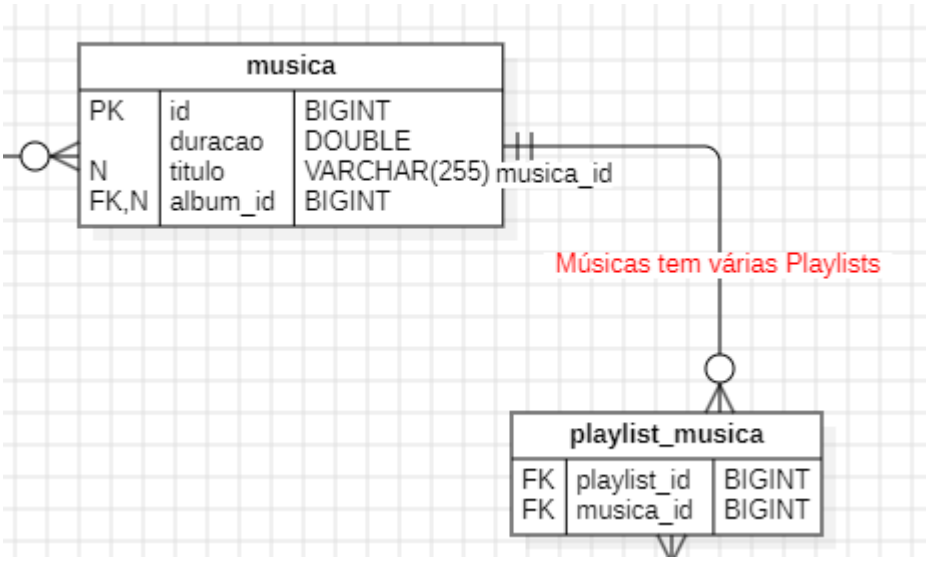

Figura 1. Diagrama de classe das entidades da Remoção de Músicas à Playlist.

2.7. Cadastro de Álbuns e Músicas (Artista)

Como um artista autenticado, quero cadastrar álbuns e músicas para que eles estejam disponíveis para outros usuários do sistema.

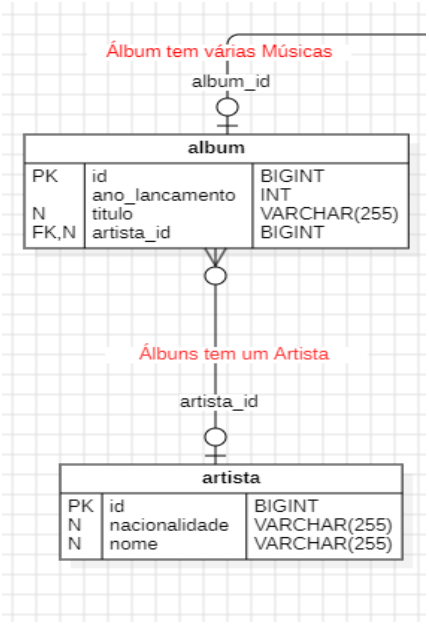

Figura 1. Diagrama de classe das entidades do Cadastro de Álbuns e Músicas (Artista).

2.8. Seguir Artistas

Como um usuário, quero seguir artistas para receber atualizações sobre novos lançamentos de álbuns e músicas.

2.9. Excluir Playlists

Como um usuário autenticado, quero poder excluir playlists que não desejo mais manter

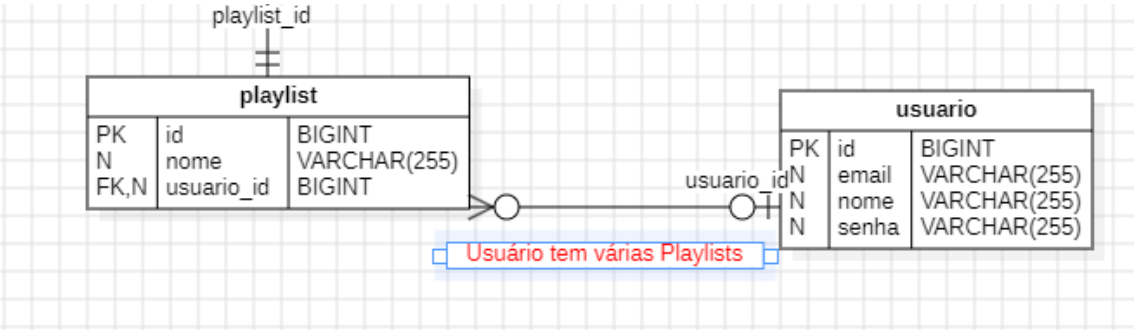

Figura 1. Diagrama de classe das entidades da Exclusão de Playlists.

3. Codificação

Apresentar as entidades e como realizou os relacionamentos. Apresentar o Diagrama complete em forma de figura XY.

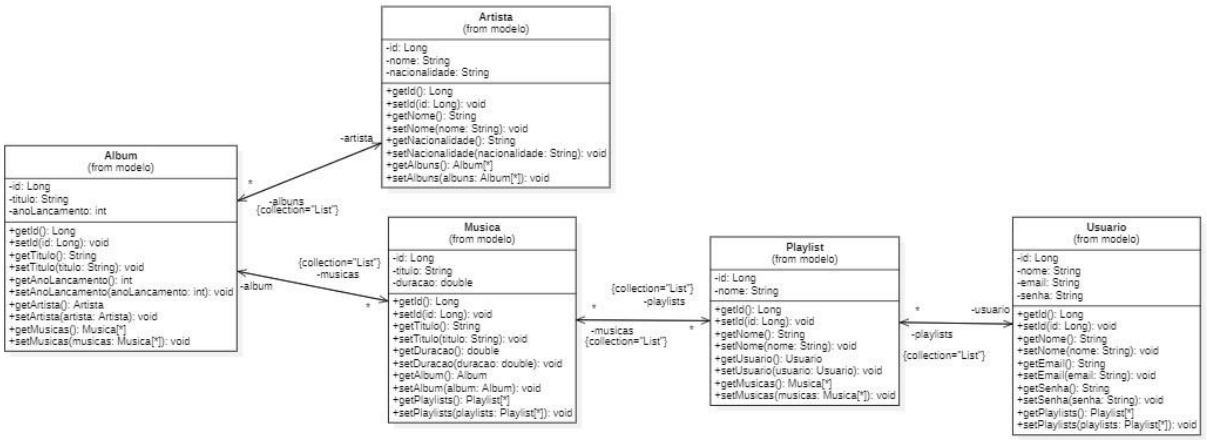

Figura XY. Diagrama de classe do Sistema de música Spotify.

3.1. Entidade Usuário

A entidade Usuário representa os usuários da plataforma, sejam eles ouvintes ou artistas. Sendo que um usuário pode **criar várias playlists** (relação um-para-muitos

com a entidade Playlist). E um usuário pode **seguir artistas** (relação muitos-para-muitos com a entidade Artista).

```java
@Entity
public class Usuario {
    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    2 usages
    private String nome;
    2 usages
    private String email;
    2 usages
    private String senha;

    2 usages
    @OneToMany(mappedBy = "usuario", cascade = CascadeType.ALL)
    private List<Playlist> playlists;
```

Figura 3.1. Código da entidade Usuário.

3.2. Entidade Playlist

A entidade Playlist representa uma coleção de músicas personalizada por um usuário. Sendo que uma playlist pertence a **um único usuário** (relação muitos-para-um com a entidade Usuário). E uma playlist pode **conter várias músicas** (relação muitos-para-muitos com a entidade Música).

```java
@Entity
public class Playlist {
    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    2 usages
    private String nome;

    2 usages
    @ManyToOne
    @JoinColumn(name = "usuario_id")
    private Usuario usuario;

    2 usages
    @ManyToMany
    @JoinTable(
            name = "playlist_musica",
            joinColumns = @JoinColumn(name = "playlist_id"),
            inverseJoinColumns = @JoinColumn(name = "musica_id"))
    private List<Musica> musicas;
```

Figura 3.2. Código da entidade Playlist.

3.3. Entidade Album

A entidade Album representa um álbum musical criado por um artista. Sendo que um álbum é **criado por um artista** (relação muitos-para-um com a entidade Artista). E um álbum pode conter **várias músicas** (relação um-para-muitos com a entidade Música).

```
@Entity
public class Album {
    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    2 usages
    private String titulo;
    2 usages
    private int anoLancamento;

    2 usages
    @ManyToOne
    @JoinColumn(name = "artista_id")
    private Artista artista;

    2 usages
    @OneToMany(mappedBy = "album", cascade = CascadeType.ALL)
    private List<Musica> musicas;
```

Figura 3.3. Código da entidade Album.

3.4. Entidade Música

A entidade música representa uma faixa musical dentro do sistema. Sendo que **uma música pertence a um álbum** (relação muitos-para-um com a entidade Álbum).E uma música pode estar em várias playlists (relação muitos-para-muitos com a entidade Playlist).

```java
@Entity
public class Musica {
    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    2 usages
    private String titulo;
    2 usages
    private double duracao;

    2 usages
    @ManyToOne
    @JoinColumn(name = "album_id")
    private Album album;

    2 usages
    @ManyToMany(mappedBy = "musicas")
    private List<Playlist> playlists;
```

Figura 3.4. Código da entidade Música.

3.5. Entidade Artista

A entidade Artista representa os artistas que disponibilizam álbuns e músicas na plataforma. Sendo que um artista pode **lançar vários álbuns** (relação um-para-muitos com a entidade Álbum). E um artista pode ser **seguido por vários usuários** (relação muitos-para-muitos com a entidade Usuário).

```java
@Entity
public class Artista {
    2 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    2 usages
    private String nome;
    2 usages
    private String nacionalidade;

    2 usages
    @OneToMany(mappedBy = "artista", cascade = CascadeType.ALL)
    private List<Album> albuns;
```

Figura 3.5. Código da entidade Artista.

4. Banco de dados

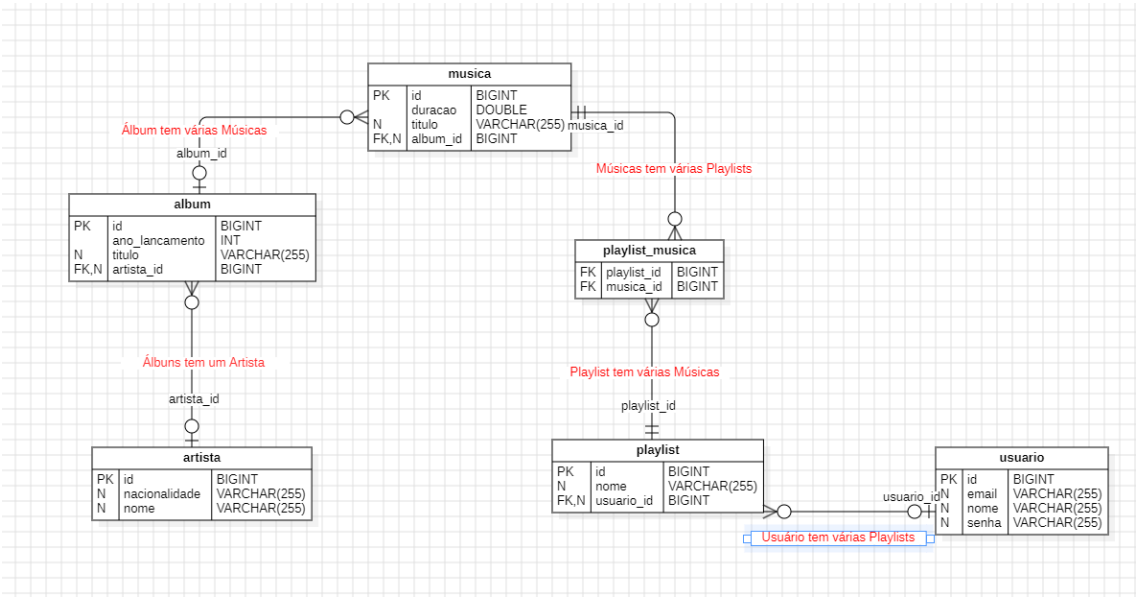

Figura 4. Modelo Diagrama MER do Sistema de Música Spotify.

4.1. Usuário

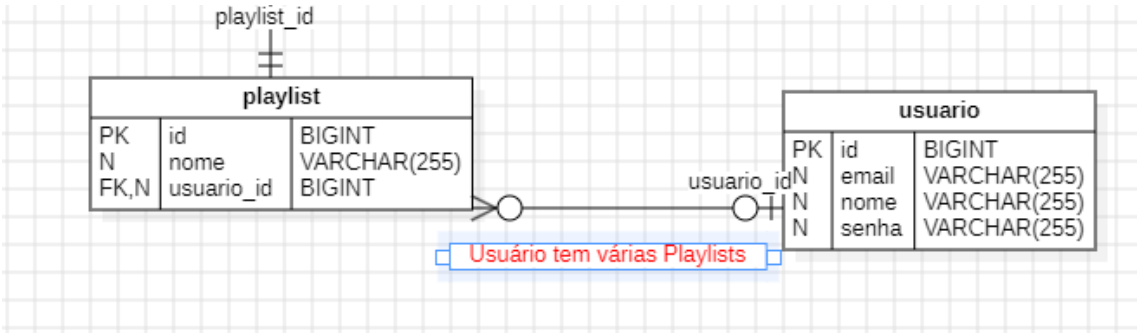

Figura 4.1. Modelo Entidade Usuário do Sistema de Música Spotify.

4.2. Playlist

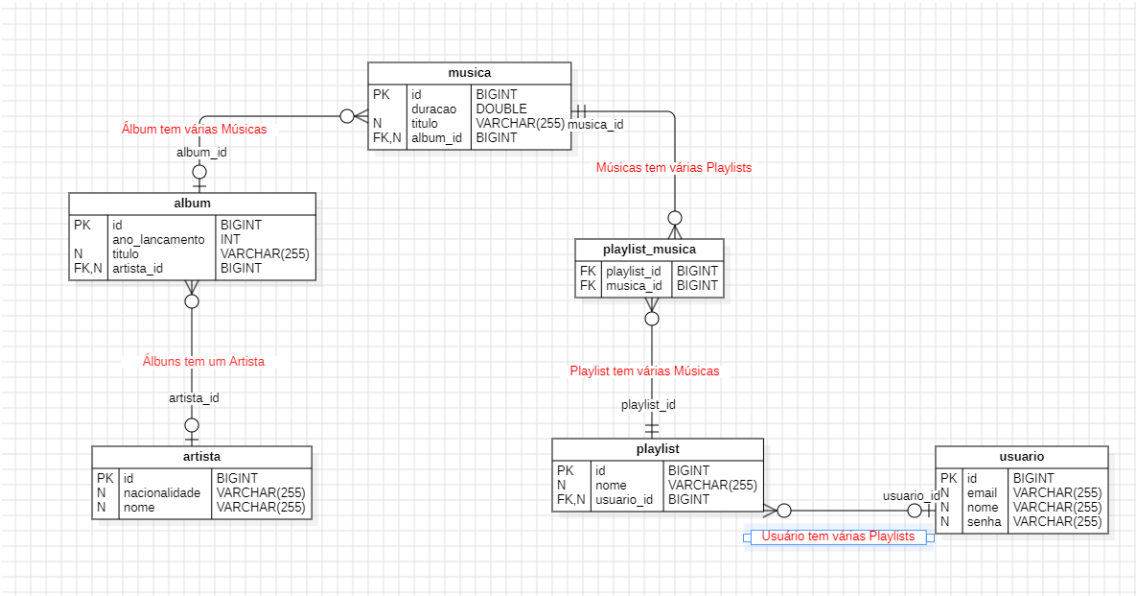

Figura 4.2.1. Modelo Entidade Playlist do Sistema de Música Spotify.

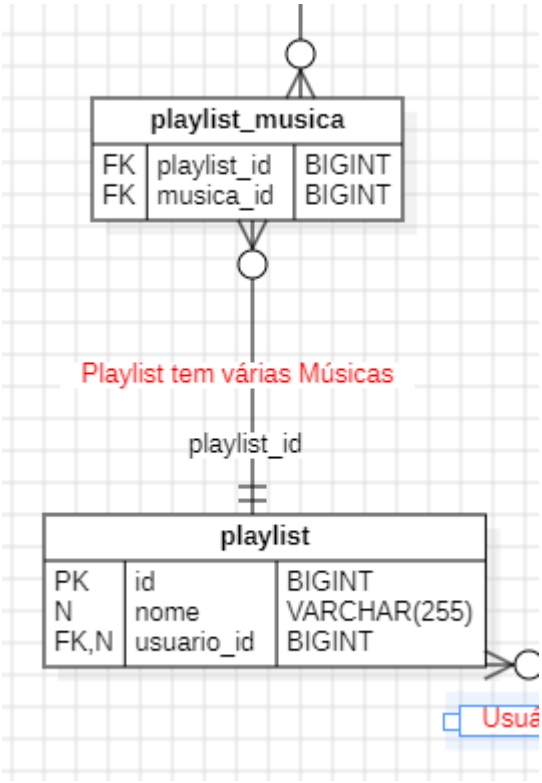

Figura 4.2.2. Modelo Entidade Playlist do Sistema de Música Spotify.

4.3. Album

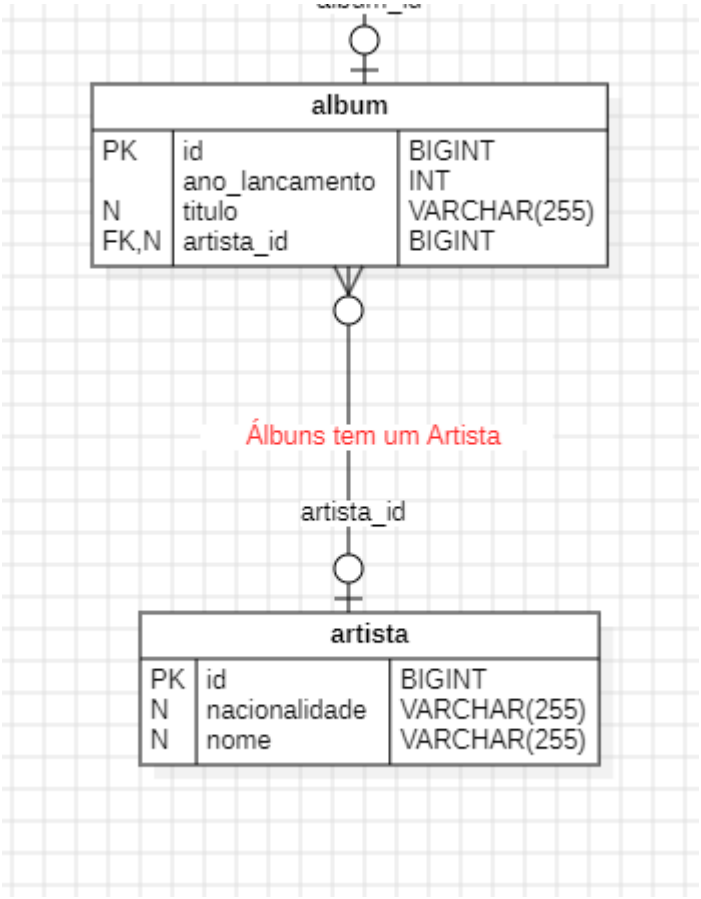

Figura 4.3.1. Modelo Entidade Album do Sistema de Música Spotify.

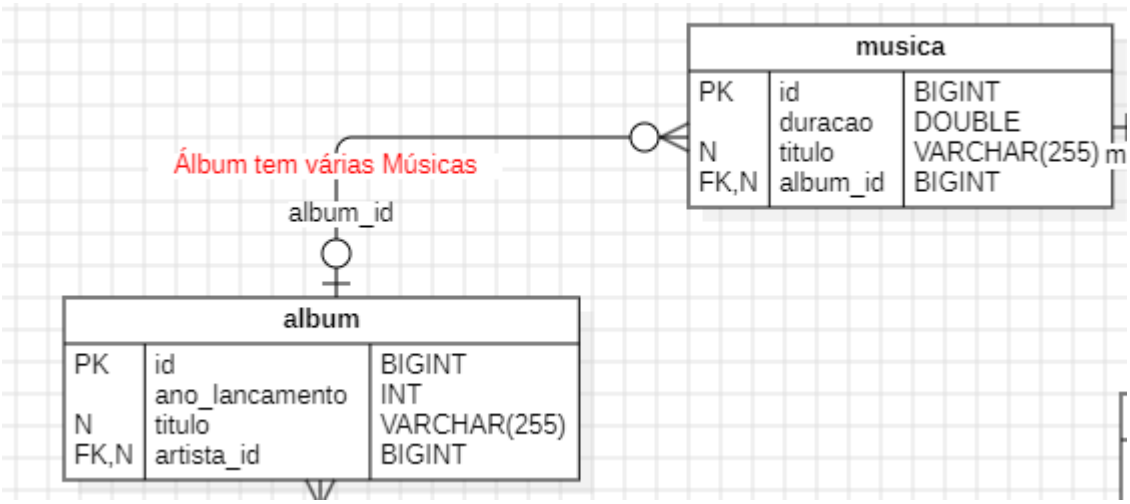

Figura 4.3.2. Modelo Entidade Album do Sistema de Música Spotify.

4.4. Música

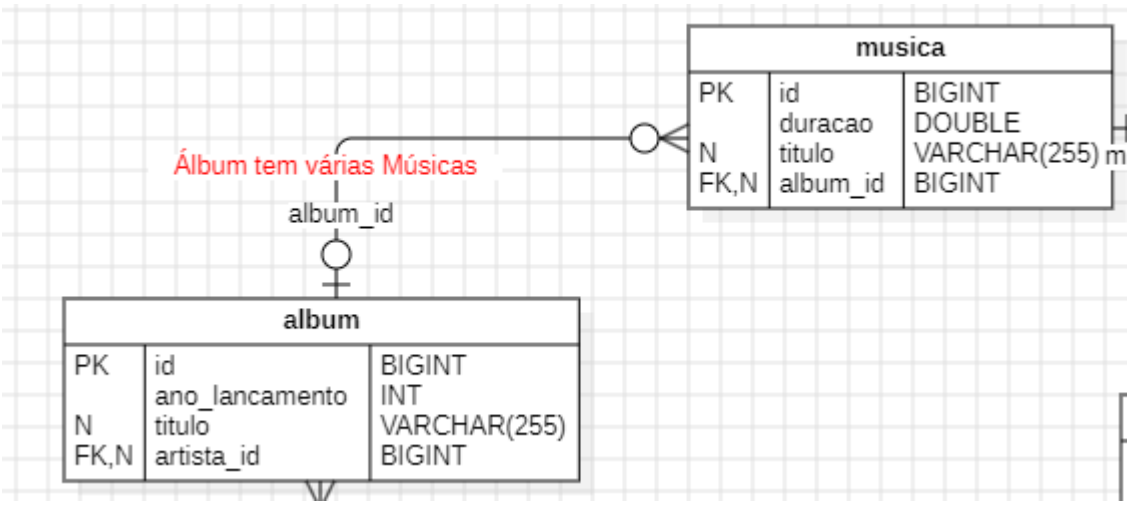

Figura 4.4.1. Modelo Entidade Musica do Sistema de Música Spotify.

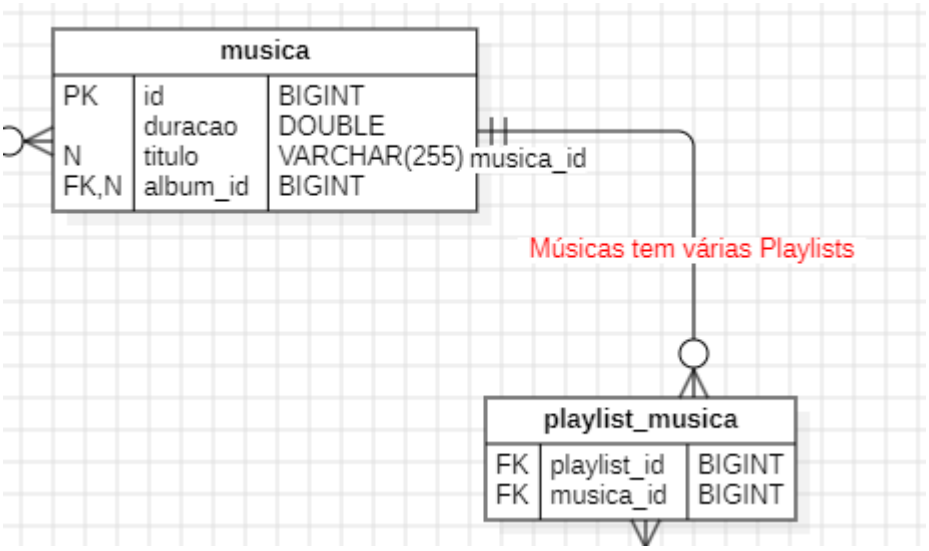

Figura 4.4.2. Modelo Entidade Musica do Sistema de Música Spotify.

4.5. Artista

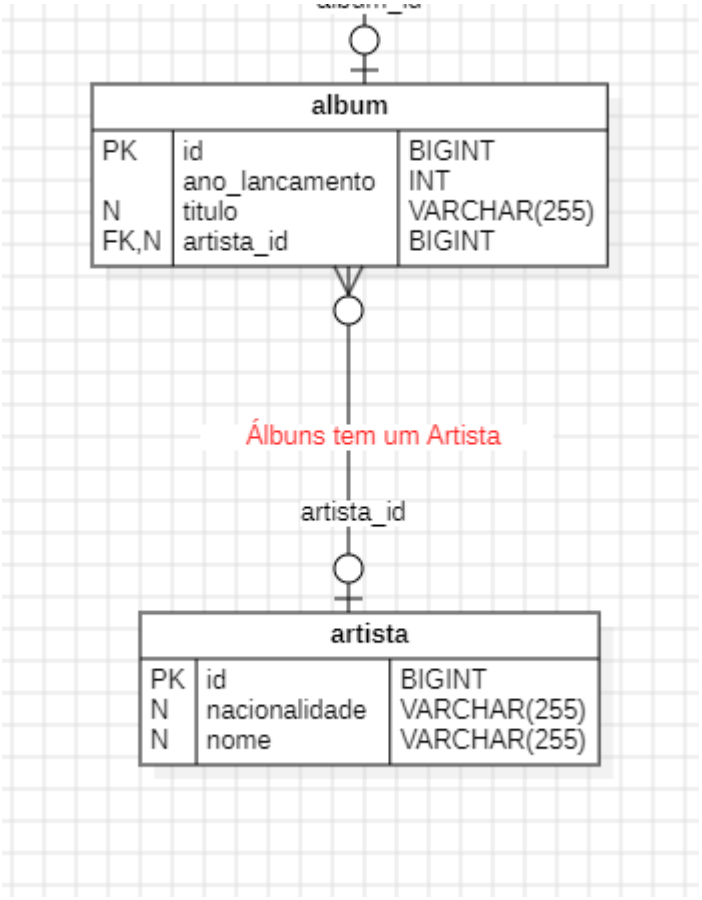

Figura 4.5. Modelo Entidade Artista do Sistema de Música Spotify.

4. Conclusão

Sobre o sistema de música, que revela uma excelente oportunidade de aprendizado, utilizando tecnologias como Java, Spring Boot e MySQL(ou outro banco de dados). O foco em histórias de usuário proporciona uma visão clara das necessidades reais, desde o cadastro até a criação de playlists, enfatizando a importância de uma abordagem centrada no usuário. O modelo de entidades, que abrange usuários, músicas, álbuns e playlists, garante uma estrutura lógica para o gerenciamento de dados. Assim, o projeto não só reforça conceitos técnicos, mas também prepara os alunos para desafios práticos no desenvolvimento de sistemas complexos e interativos.

Referências

Boulic, R. and Renault, O. (1991) “3D Hierarchies for Animation”, In: New Trends in Animation and Visualization, Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd., England.

Dyer, S., Martin, J. and Zulauf, J. (1995) “Motion Capture White Paper”, http://reality.sgi.com/employees/jam_sb/mocap/MoCapWP_v2.0.html, December.

Evans, E. (2003) “Domain-Driven Design: Tackling Complexity in the Heart of Software”, Addison-Wesley Professional, United States.

Holton, M. and Alexander, S. (1995) “Soft Cellular Modeling: A Technique for the Simulation of Non-rigid Materials”, Computer Graphics: Developments in Virtual Environments, R. A. Earnshaw and J. A. Vince, England, Academic Press Ltd., p. 449-460.

Knuth, D. E. (1984), The TeXbook, Addison Wesley, 15th edition.

Pressman, R. S., and Maxim, B. R. (2020) *Software Engineering: A Practitioner's Approach*, 9th Edition, McGraw-Hill Education, United States.

Smith, A. and Jones, B. (1999). On the complexity of computing. In *Advances in Computer Science*, pages 555–566. Publishing Press.