# Sistema de Música

**Karolayne da Silva Alexandre, Marcelo Andrejov Junior**

Sistemas de Informação

Universidade da Região de Joinville (UNIVILLE) – Joinville, SC – Brazil

karolayne.alexandre@univille.br , marcelo.andrejov@univille.br

## 1. Introdução

Este trabalho visa desenvolver um sistema de gerenciamento musical, inspirado no Spotify, usando Java, Spring e banco de dados relacional. O sistema permite que usuários gerenciem playlists, músicas, álbuns e artistas. As principais entidades são Artista, Gênero, Album, Musica, Playlist e Usuário, interligadas por relacionamentos como muitos para muitos e um para muitos. A arquitetura usa JPA para gerenciar operações no banco de dados, garantindo integridade e facilidade de manutenção, com foco em escalabilidade e flexibilidade futuras.

## 2. Requisitos Funcionais

**Busca de Artistas**: O sistema deve permitir que os usuários busquem artistas cadastrados, com nome e gêneros musicais associados.

**Busca de Gêneros**: O sistema deve permitir que os usuários pesquisem e visualizem gêneros musicais, que podem estar associados a artistas.

**Visualização de Álbuns**: O sistema deve permitir que os usuários visualizem álbuns, com detalhes como nome, ano de lançamento e a lista de artistas participantes.

**Visualização de Músicas**: O sistema deve permitir que os usuários visualizem músicas, incluindo título, duração e o álbum ao qual pertencem.

**Criação de Playlists**: O sistema deve permitir que os usuários criem playlists, definindo nome, descrição e a lista de músicas associadas.

**Gerenciamento de Playlists**: O sistema deve permitir que os usuários adicionem ou removam músicas em suas playlists.

**Cadastro de Usuário**: O sistema deve permitir que novos usuários se registrem, fornecendo nome e e-mail, para que possam criar e gerenciar suas playlists.

**Visualização de Playlists**: O sistema deve permitir que os usuários visualizem suas playlists com detalhes das músicas adicionadas.

**Gerenciamento de Músicas em Playlists**: O sistema deve permitir que os usuários editem, adicionem ou removam músicas em suas playlists para organizá-las conforme suas preferências.

**Busca de Músicas e Artistas**: O sistema deve permitir que os usuários busquem por músicas e artistas cadastrados.

### 2.1 História de Usuário 01: Procurar Artista

Como usuário, eu quero procurar artistas no sistema, para que eles possam ser associados a álbuns e músicas.

A Figura 01 é o diagrama de classe da história de usuário 01. A entidade Artista está associada a uma lista de Álbuns, o que indica que um artista pode ter vários álbuns. O relacionamento entre Artista e Álbum também é do tipo Muitos para Muitos, uma vez que um álbum pode conter múltiplos artistas.

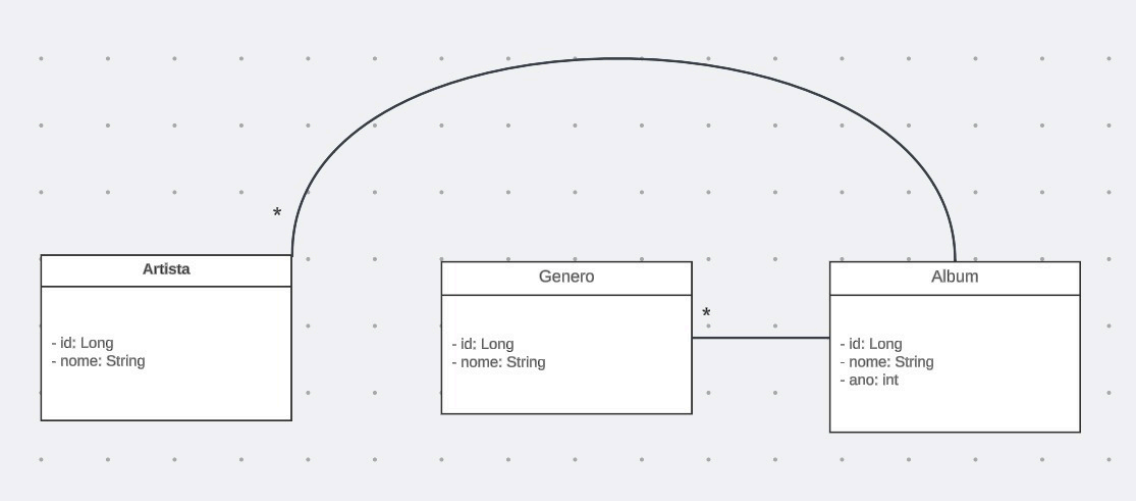


**Figura 1. Diagrama de classes da entidades da História de Usuário 01.**

A Figura 2  mostra a tabela Artista, que armazena as informações dos artistas no sistema. O campo id é a chave primária (PK) e é do tipo BIGINT, identificando exclusivamente cada artista. O campo nome contém o nome do artista. A Chave Primária (PK): id — garante a unicidade de cada artista. Relacionamentos: Um artista pode estar associado a vários gêneros musicais, estabelecendo um relacionamento Muitos para Muitos (N) entre Artista e Genero. Um artista pode ter participado de vários álbuns, o que define um relacionamento Muitos para Muitos (N) entre Artista e Album.

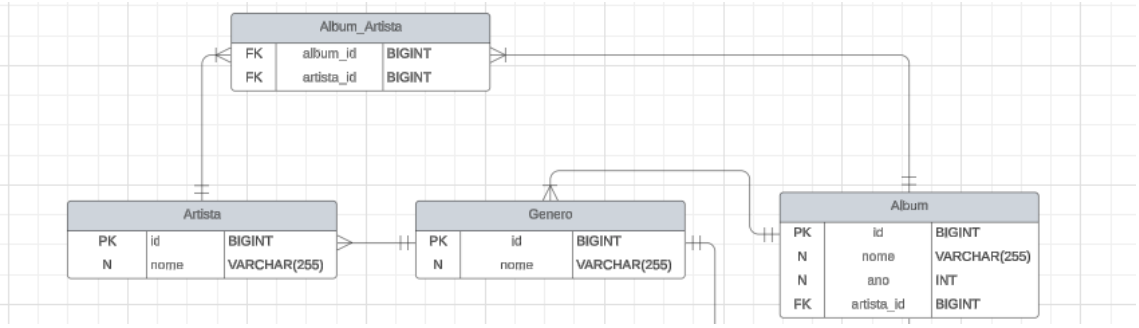


**Figura 2. Modelo Entidade Relacionamento da História de Usuário 01.**

## 2.2 História de Usuário 02: Cadastrar novos gêneros musicais

Como Usuário, eu quero buscar gêneros musicais, para que os artistas possam ser corretamente categorizados

A Figura 03 ilustra o diagrama de classe da história de usuário 02. A Entidade Usuário pode criar várias Playlists, representando um relacionamento do tipo Um para Muitos. Cada Playlist pode conter várias Músicas, e uma música pode estar presente em várias playlists, configurando um relacionamento do tipo Muitos para Muitos. Portanto, a relação entre Playlist e Música permite que os usuários organizem suas músicas favoritas de forma flexível.

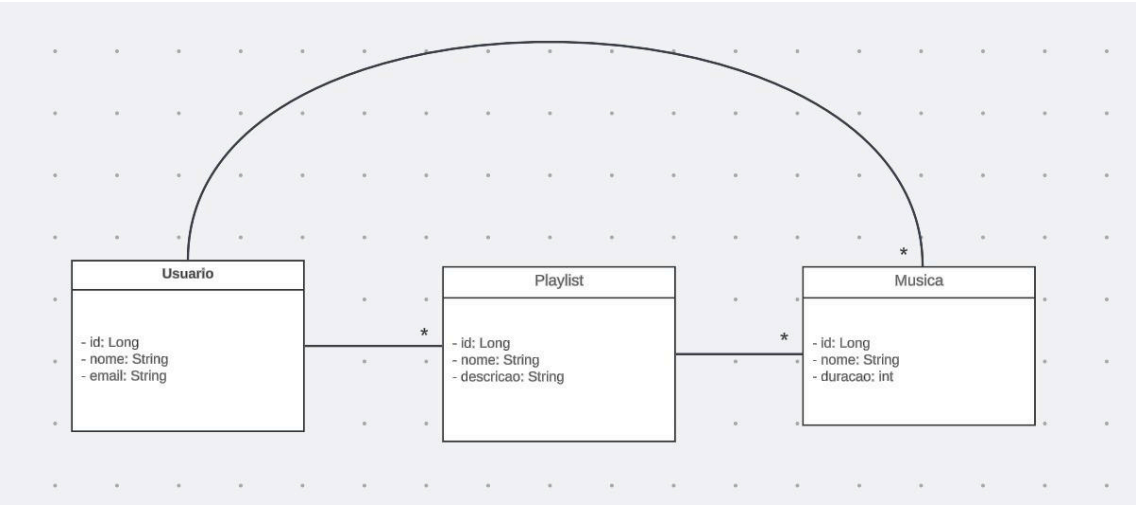


**Figura 3. Diagrama de classes da entidades da História de Usuário 02.**

A Figura 04 mostra a tabela Musica, que armazena as músicas no sistema. O campo id é a chave primária (PK) e é do tipo BIGINT, identificando exclusivamente cada música. Os campos titulo e duracao armazenam, respectivamente, o título da música e sua duração. A Chave Primária (PK): id — garante a unicidade de cada música.

Relacionamentos: Uma música pertence a um único álbum, definindo um relacionamento Muitos para Um (N:1) entre Musica e Album. Uma música pode estar em várias playlists, estabelecendo um relacionamento Muitos para Muitos (N) entre Musica e Playlist.

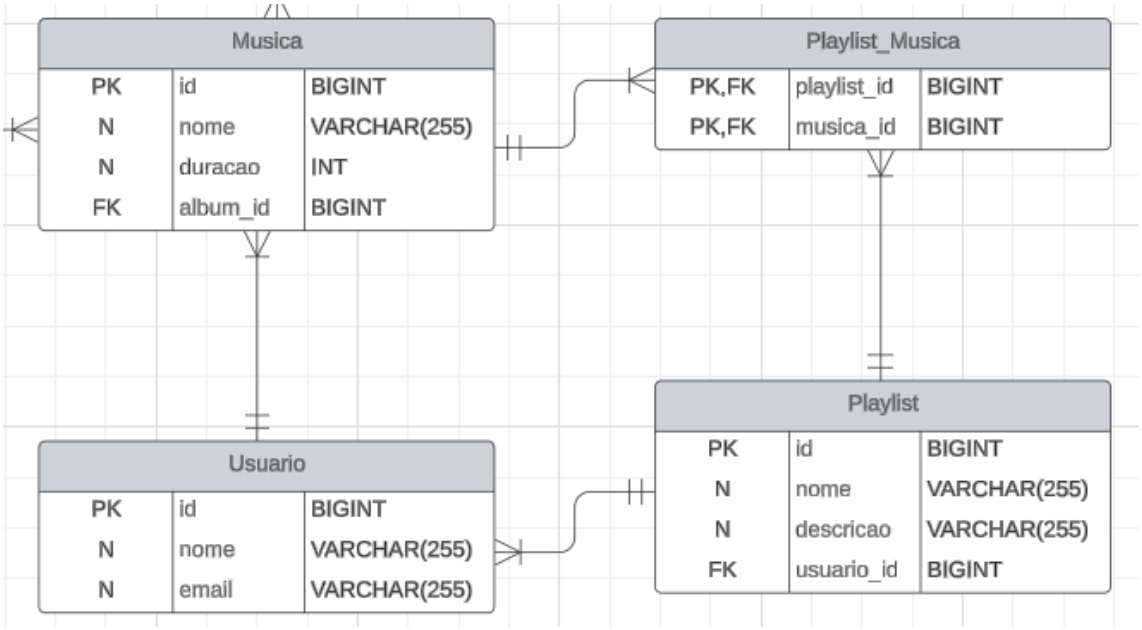


**Figura 4. Modelo Entidade Relacionamento da História de Usuário 02.**

### 2.3 História de Usuário 03: Adicionar Álbum

Como usuário, eu quero editar as informações de um álbum existente, para que os dados estejam sempre atualizados.

A Figura 05 apresenta o diagrama de classe da história de usuário 03. A entidade Álbum está associada à entidade Artista, onde um álbum pode ter vários artistas (Muitos para Muitos) e pode conter várias Músicas (Um para Muitos). Isso demonstra que cada álbum pode representar o trabalho de múltiplos artistas e incluir diversas músicas, permitindo a organização do conteúdo musical de forma estruturada.

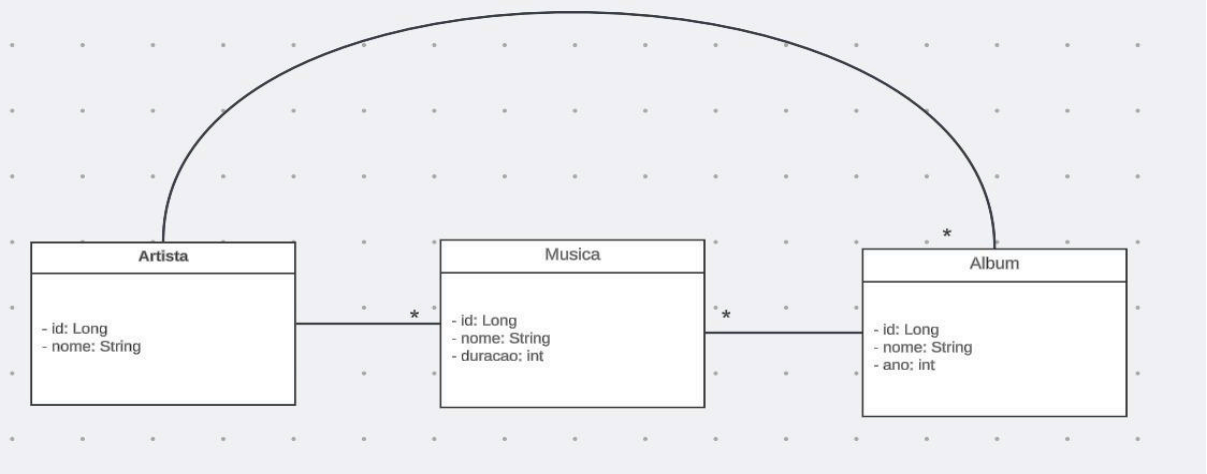


**Figura 5. Diagrama de classes da entidades da História de Usuário 03.**

A Figura 06 apresenta o diagrama de classe da história de usuário 03, onde a entidade Música está associada a outras entidades no sistema de gerenciamento de músicas. A entidade Música está associada à entidade Álbum, em um relacionamento Muitos para Um (N:1), indicando que uma música pertence a apenas um álbum, mas um álbum pode conter várias músicas (Um para Muitos). Isso permite organizar as músicas de acordo com o álbum ao qual pertencem.Além disso, a entidade Música está associada à entidade Playlist, em um relacionamento Muitos para Muitos (N), onde uma música pode estar presente em várias playlists, e uma playlist pode conter várias músicas. Essa estrutura flexível possibilita que usuários criem playlists personalizadas com uma seleção diversa de músicas.

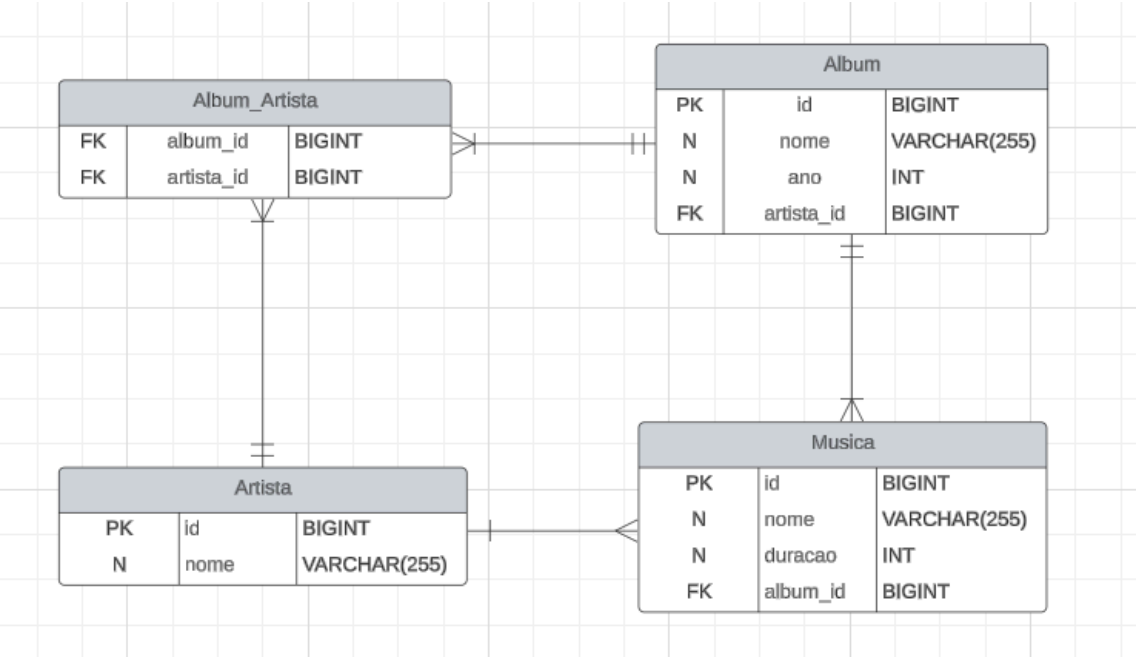

**Figura 6. Modelo Entidade Relacionamento da História de Usuário 03.**

## 2.4 História de Usuário 04: Gerenciar Músicas

Como usuário, eu quero editar as informações de uma música existente, para garantir que os dados estejam corretos.

A Figura 07 mostra o diagrama de classe da história de usuário 04. A entidade Álbum contém várias Músicas, caracterizando um relacionamento do tipo Um para Muitos. Além disso, a entidade Música pode estar associada a várias Playlists, representando um relacionamento do tipo Muitos para Muitos. Isso permite que o administrador gerencie as músicas em diferentes contextos, assegurando que a biblioteca musical permaneça organizada e acessível.

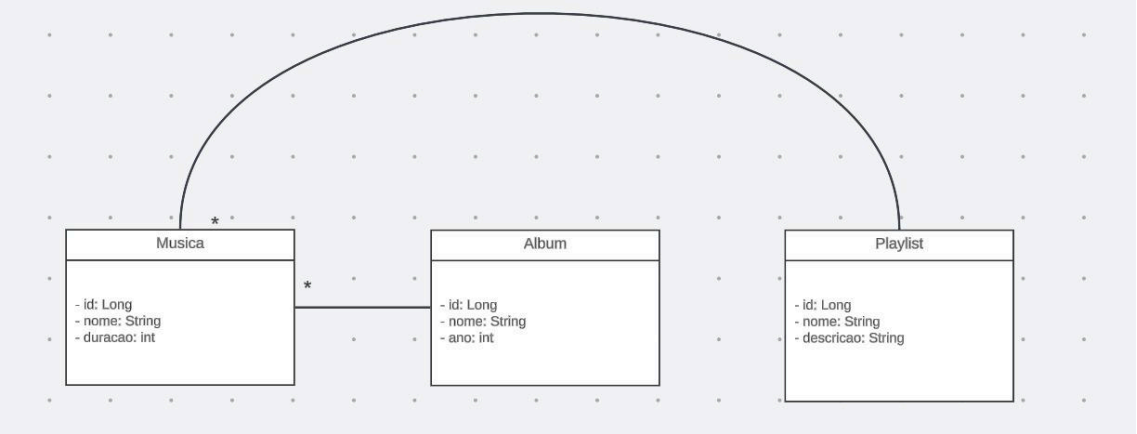


**Figura 7. Diagrama de classes da entidades da História de Usuário 04.**

A Figura 08 mostra o diagrama de classe da história de usuário 04, onde a entidade Playlist está associada a outras entidades do sistema de gerenciamento de músicas.A entidade Playlist está associada à entidade Música em um relacionamento Muitos para Muitos (N). Isso significa que uma playlist pode conter várias músicas, e, ao mesmo tempo, uma música pode estar associada a várias playlists. Essa associação facilita a criação de playlists personalizadas por parte dos usuários, que podem reunir suas músicas favoritas em diversas coleções. Além disso, a entidade Playlist está relacionada à entidade Usuário em um relacionamento Muitos para Um (N:1). Isso implica que cada playlist pertence a um único usuário, mas um usuário pode criar várias playlists, permitindo uma organização musical personalizada. No diagrama apresentado, a classe de associação entre Playlist e Música representa o relacionamento Muitos para Muitos entre essas duas entidades. Essa classe de associação é implementada através de uma tabela intermediária na base de dados, onde cada registro conecta uma playlist a uma música específica.

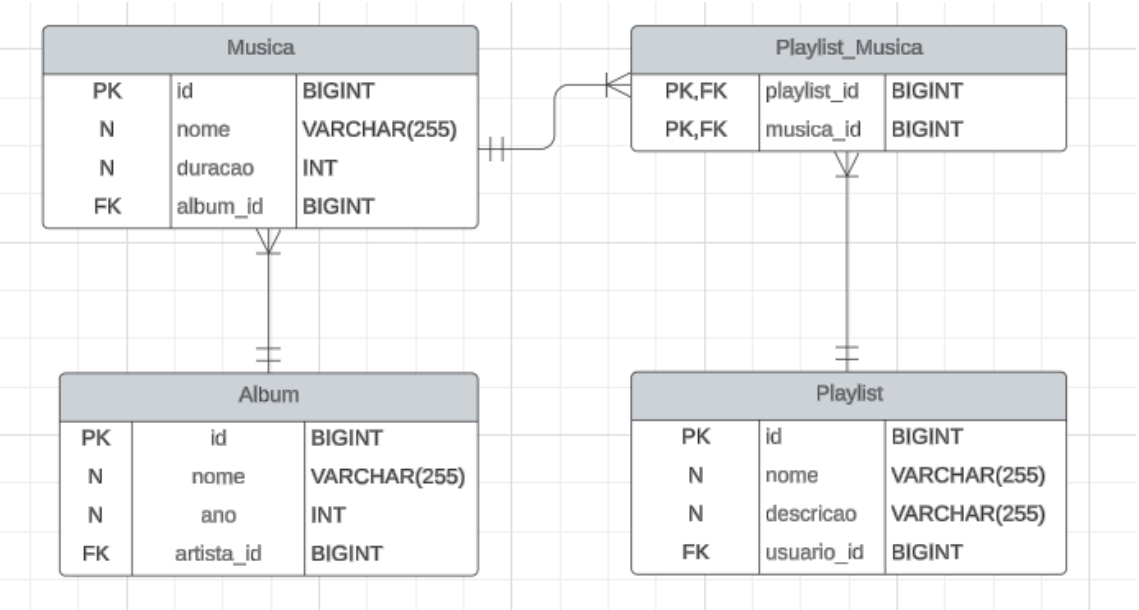


**Figura 8. Modelo Entidade Relacionamento da História de Usuário 04.**

## 2.5 História de Usuário 05: Gerenciar Usuários

Como um usuário, eu quero editar a descrição de uma playlist, para que ela reflita melhor meu gosto musical

A Figura 09 descreve o diagrama de classe da história de usuário 05. A entidade Usuário pode criar várias Playlists, estabelecendo um relacionamento Um para Muitos. Isso significa que um usuário pode ter várias playlists, cada uma podendo conter múltiplas músicas, organizando suas preferências musicais de forma personalizada. Essa estrutura permite que as informações do usuário sejam geridas de maneira eficiente.

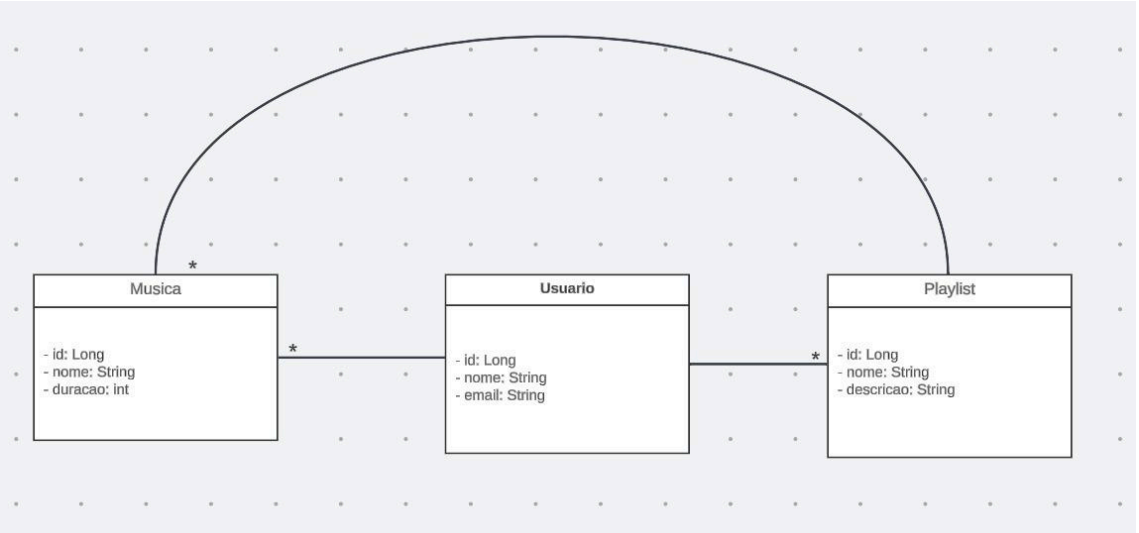


**Figura 9. Diagrama de classes da entidades da História de Usuário 05.**

Na Figura 10, além do relacionamento Um para Muitos entre a entidade Usuário e

Playlist, também existe uma classe associativa importante que aparece entre Playlist e Música. Esta classe associativa é essencial para representar o relacionamento Muitos para Muitos, pois uma Playlist pode conter várias músicas, e uma mesma Música pode estar presente em várias playlists. Esse relacionamento é implementado por meio de uma classe associativa, que define as regras de mapeamento entre Playlist e Música. No diagrama de classe, isso é feito com uma tabela intermediária na base de dados que une as chaves primárias das duas entidades, permitindo essa multiplicidade. Essa classe associativa organiza a relação de maneira estruturada, garantindo que os dados de playlists e músicas sejam geridos corretamente.

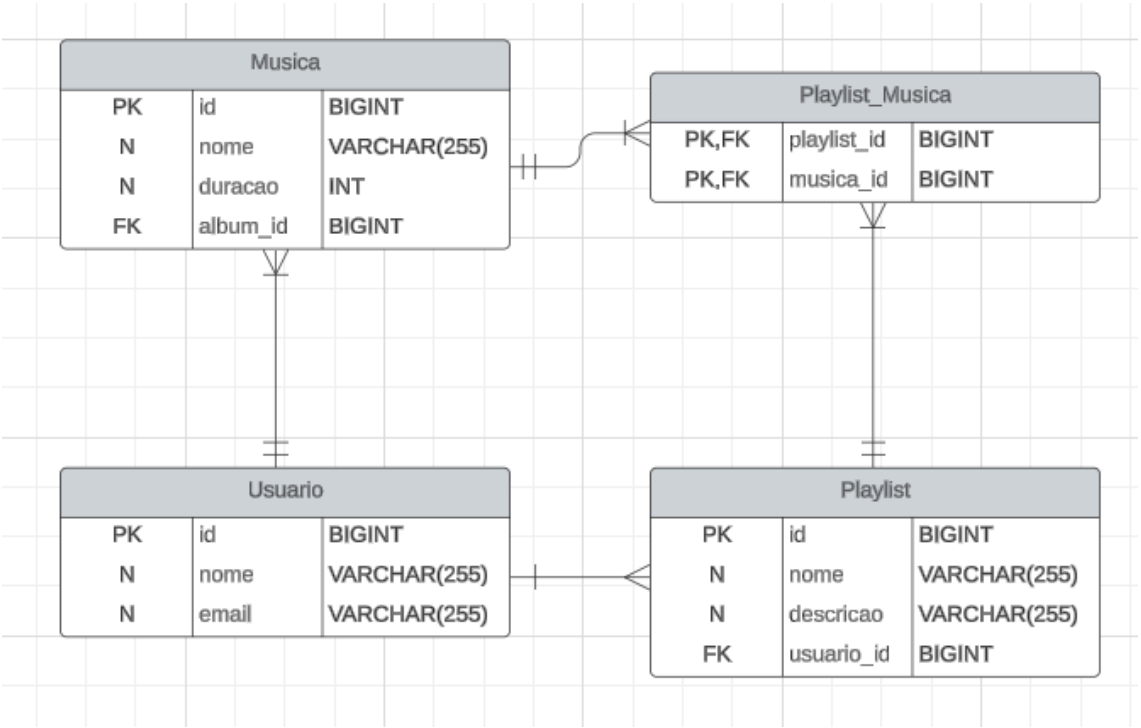


**Figura 10. Modelo Entidade Relacionamento da História de Usuário 05.**

### 3.1. Entidade Artista

A entidade Artista representa os artistas no sistema e contém os seguintes atributos: id, que é uma identificação única gerada automaticamente, e nome, que armazena o nome do artista. A entidade Artista estabelece um relacionamento muitos para muitos com a entidade Genero, indicando que um artista pode estar associado a vários gêneros e que um gênero pode ter vários artistas. Além disso, a entidade também possui um relacionamento muitos para muitos com a entidade Album, onde um artista pode participar de vários álbuns, e um álbum pode ter várias colaborações de diferentes artistas.

```
@Data

@NoArgsConstructor
```

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

    private String nome;

    @ManyToMany
    @JoinTable(name = "artista_genero",
            joinColumns = @JoinColumn(name = "artista_id"),
            inverseJoinColumns = @JoinColumn(name =
"genero_id"))
    private List<Genero> generos;

    @ManyToMany(mappedBy = "artistas")
    private List<Album> albuns;
}
```

**Figura 11. Código da entidade Artista**

### 3.2. Entidade Gênero

A entidade Artista representa os artistas no sistema e contém os seguintes atributos: id, que é uma identificação única gerada automaticamente, e nome, que armazena o nome do artista. A entidade Artista estabelece um relacionamento muitos para muitos com a entidade Gênero, indicando que um artista pode estar associado a vários gêneros e que um gênero pode ter vários artistas. Além disso, a entidade também possui um relacionamento muitos para muitos com a entidade Álbum, onde um artista pode participar de vários álbuns, e um álbum pode ter várias colaborações de diferentes artistas.

```
@Data
@NoArgsConstructor
```

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

    private String nome;

    @ManyToMany(mappedBy = "generos")
    private List<Artista> artistas;
}
```

**Figura 12. Código da entidade Genero**

### 3.3. Entidade Album

A entidade Album representa álbuns de música e possui os atributos id, nome e ano. A entidade Album tem um relacionamento muitos para muitos com a entidade Artista, permitindo que um álbum tenha várias contribuições de diferentes artistas. Também existe um relacionamento um para muitos com a entidade Musica, onde um álbum pode conter várias músicas, refletindo a organização do conteúdo musical de maneira estruturada.

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

    private String nome;
    private int ano;
```

```
    @ManyToMany
    @JoinTable(name = "album_artista",
            joinColumns = @JoinColumn(name = "album_id"),
            inverseJoinColumns = @JoinColumn(name =
"artista_id"))
    private List<Artista> artistas;


    @OneToMany(mappedBy = "album")
    private List<Musica> musicas;
}
```

**Figura 13. Código da entidade Album**

### 3.4. Entidade Musica

A entidade Musica representa as músicas no sistema e contém os atributos id, titulo e duração. A entidade Musica tem um relacionamento muitos para um com a entidade Album, indicando que cada música pertence a um único álbum. Além disso, a entidade Musica estabelece um relacionamento muitos para muitos com a entidade Playlist, o que permite que uma música esteja presente em várias playlists e que uma playlist possa conter várias músicas, facilitando a organização e a gestão de preferências musicais.

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


    private String titulo;
    private int duracao;
```

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

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

**Figura 14. Código da entidade Musica**

### 3.5. Entidade Playlist

A entidade Playlist representa as listas de reprodução criadas pelos usuários e contém os atributos id, nome e descricao. A entidade Playlist tem um relacionamento muitos para muitos com a entidade Musica, permitindo que os usuários organizem suas músicas favoritas em diferentes playlists. Além disso, existe um relacionamento muitos para um com a entidade Usuario, indicando que cada playlist pertence a um único usuário, permitindo a personalização das playlists conforme as preferências individuais.

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

    private String nome;
    private String descricao;

    @ManyToMany
    @JoinTable(name = "playlist_musica",
            joinColumns = @JoinColumn(name =
"playlist_id"),
```

```
            inverseJoinColumns = @JoinColumn(name =
"musica_id"))
    private List<Musica> musicas;


    @ManyToOne
    @JoinColumn(name = "usuario_id")
    private Usuario usuario;
}
```

**Figura 15. Código da entidade Playlist**

### 3.6. Entidade Usuario

A entidade Usuario representa os usuários do sistema e contém os atributos id, nome e email. A entidade Usuario estabelece um relacionamento um para muitos com a entidade Playlist, permitindo que um usuário crie várias playlists. Isso possibilita que os usuários organizem e gerenciem suas preferências musicais de forma personalizada e eficaz.

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


    private String nome;
    private String email;


    @OneToMany(mappedBy = "usuario")
    private List<Playlist> playlists;
}
```

**Figura 16. Código da entidade Usuario**

## 4. Banco de dados

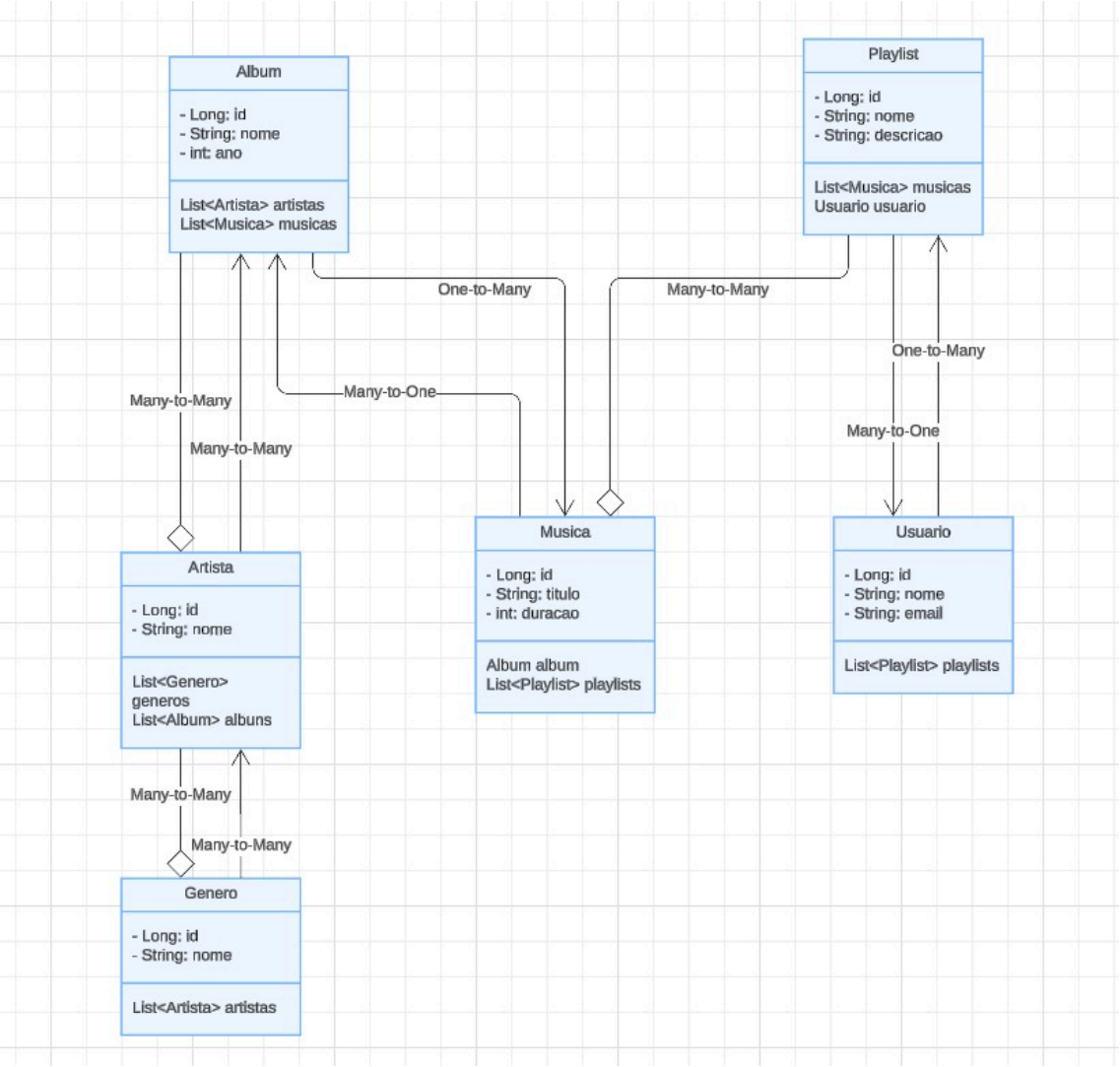


**Figura 16. Diagrama de classes do Sistema de música**

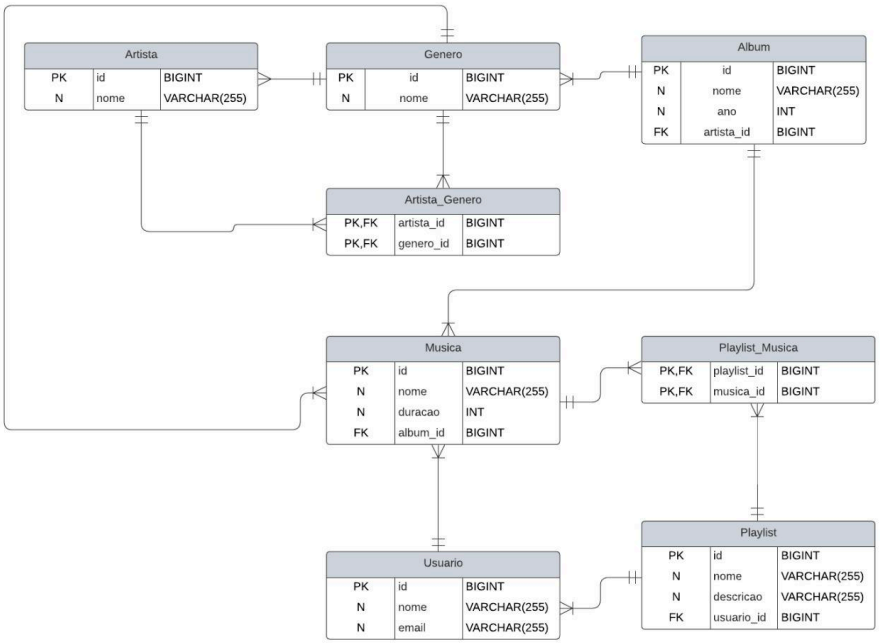


**Figura 17. Modelo Entidade Relacionamento do Sistema de música**

## 5. Conclusão

Este trabalho descreve o desenvolvimento de um sistema de gerenciamento de músicas, que organiza entidades essenciais como Artista, Álbum, Música, Gênero, Playlist e Usuário. Os relacionamentos entre essas entidades foram claramente definidos, permitindo uma interação intuitiva.

Histórias de usuário guiaram a implementação das funcionalidades, atendendo às necessidades de administradores e usuários finais. O sistema facilita o cadastro e a gestão de conteúdos musicais, além de proporcionar uma experiência personalizada na criação de playlists.

Com uma abordagem centrada no usuário e uma navegação eficiente, o sistema se torna uma ferramenta valiosa para amantes da música. Futuros aprimoramentos poderão incluir recomendações de músicas e interação social entre usuários, enriquecendo a experiência da plataforma.

## 6. Referências

Spring Framework Documentation. (n.d.). “Spring Framework Documentation.” Disponível em: https://spring.io/

Oracle. (2024). “Java Platform, Standard Edition Documentation.” Disponível em: https://docs.oracle.com/en/java/javase/