# Música XPO

**PedroVitor De Campos Neres e Felipe Grimm**

## 1. Introdução

Nosso sistema simulará um software de música que permitirá aos usuários criar planos e fazer login para salvar dados, incluindo playlists com músicas e álbuns curtidos, além de possibilitar o download dos mesmos.

## 2. Requisitos Funcionais

As histórias a seguir descreverão as funcionalidades do software, como criar um cadastro, fazer login, pesquisar músicas e artistas, salvar e baixar músicas, e criar playlists.

### *2.1. História de Usuário 01*

Como usuário do Sistema de Música, desejo criar meu cadastro. Para isso, preciso fornecer um e-mail e uma senha para garantir que meus dados sejam salvos. Em seguida, devo escolher um plano de assinatura, que pode ser gratuito ou pago. Para completar meu perfil, precisarei informar meu nome completo, adicionar uma foto, curtir as músicas que gosto, para que o aplicativo possa sugerir outras com base nos meus interesses, e configurar o tema que desejo, seja claro ou escuro.

A **Figura 01** ilustra o diagrama UML da história de usuário 01. A entidade **Usuário** está diretamente relacionada à entidade **Perfil**, sendo necessárias informações como nome, foto, músicas curtidas e configuração do tema.

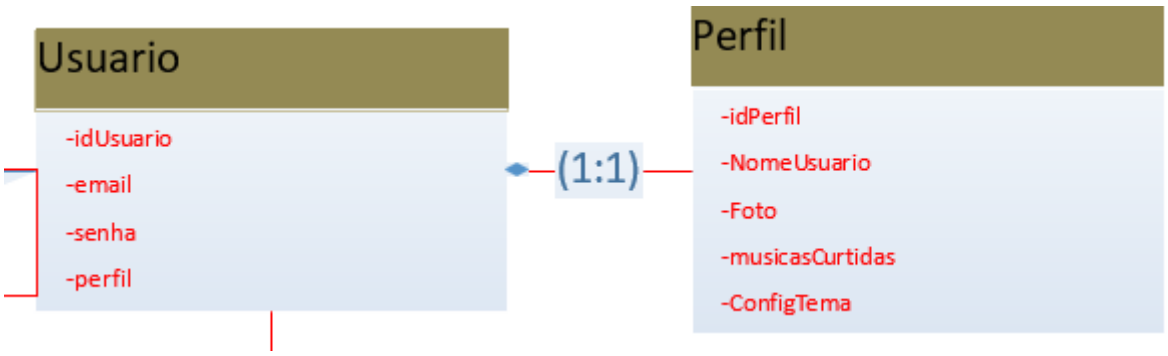


**Figura 1:** UML das entidades da História de Usuário 01.

A **Figura 02** apresenta o Modelo Entidade-Relacionamento (MER) da história de usuário 01. A entidade **Usuário** contém as chaves estrangeiras da entidade **Perfil**. A entidade **Usuário** armazena informações de login e senha. Ela estabelece um relacionamento de 1:1 com a entidade **Perfil**, que contém informações pessoais e compartilha o **idPerfil** com a entidade **Usuário**.

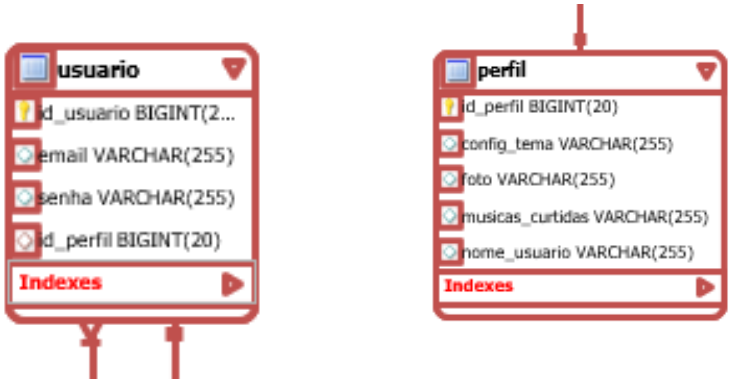


**Figura 2:** Modelo Entidade-Relacionamento da História de Usuário 01.

## 1.1. História de Usuário 02

Como usuário do Sistema de Música, desejo salvar um álbum. Para isso, preciso pesquisar o artista desejado e, através das músicas, escolher o álbum que quero salvar no meu perfil.

A **Figura 03** apresenta o diagrama UML da história de usuário 02. A entidade **Artista** se relaciona diretamente com a entidade **Música**, que pertence a um **Álbum**. São necessárias informações sobre o artista, como o nome, e sobre o álbum, como o nome e a data de lançamento.

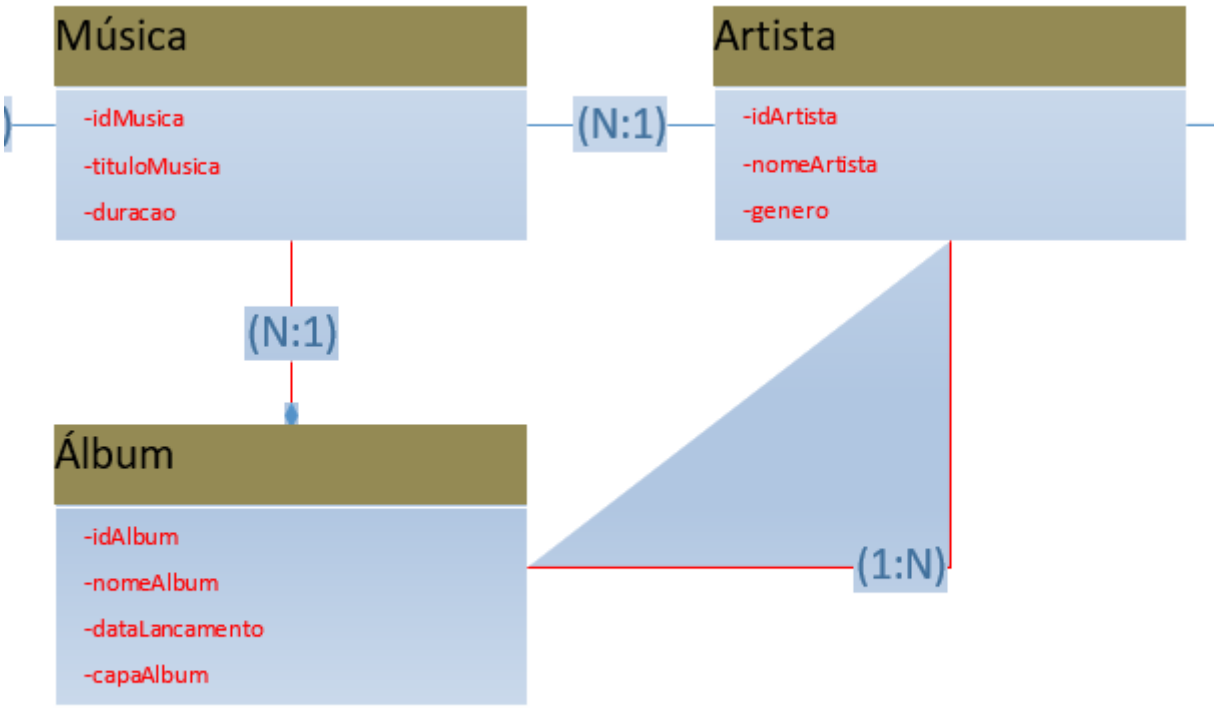


**Figura 3:** UML das entidades da História de Usuário 02.

A **Figura 04** apresenta o Modelo Entidade-Relacionamento (MER) da história de usuário 02. A entidade **Artista** contém as chaves estrangeiras da entidade **Álbum**. Ela armazena informações como nome, ID e gênero musical do artista. A entidade **Artista** estabelece um relacionamento de 1

com a entidade **Álbum**, que contém as informações sobre o ID do álbum, nome, data de lançamento e capa do álbum.

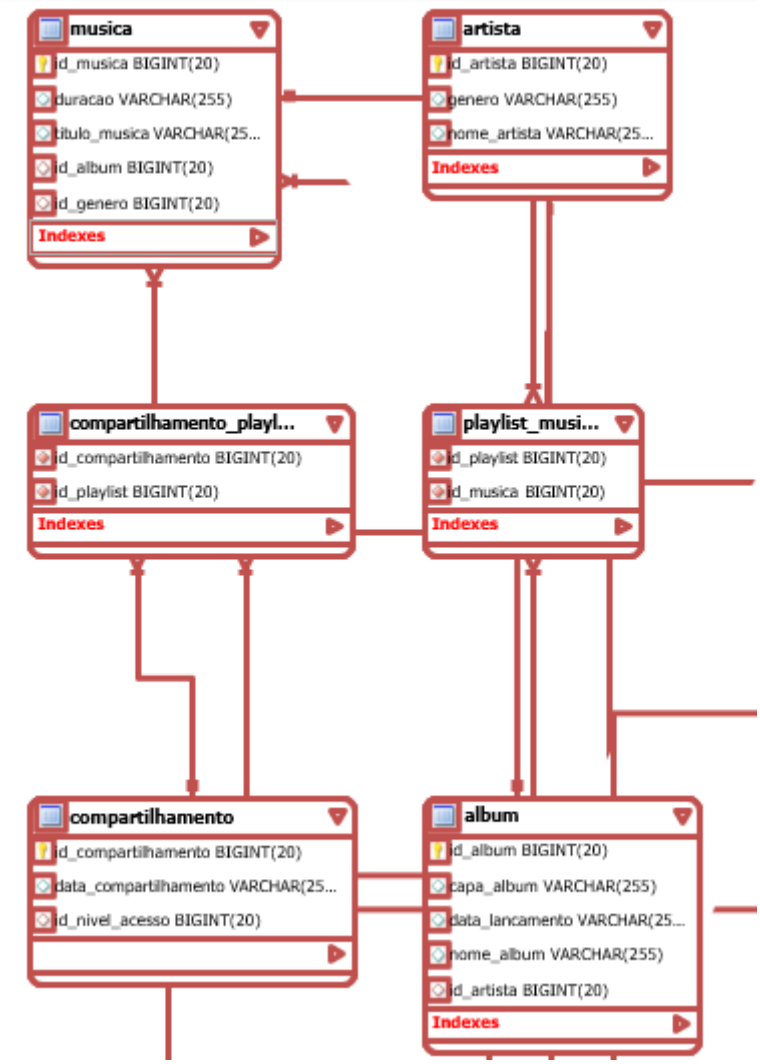


**Figura 4. MER das entidades da História de Usuário 02.**

## 1.2. História de Usuário 03

Como usuário do Sistema de Música, desejo compartilhar uma playlist com amigos. Para isso, preciso escolher a playlist e compartilhar com o usuário desejado, digitando o e-mail dele.

A **Figura 05** apresenta o diagrama UML da história de usuário 03. A entidade **Playlist** se relaciona diretamente com a entidade **Usuário**. A entidade **Usuário** se conecta à entidade **Compartilhamento**, que, por sua vez, se relaciona com a entidade **Nível de Acesso**. Essa estrutura permite definir as permissões necessárias para cada usuário que receberá a playlist compartilhada. São necessárias informações como o nome da playlist, o e-mail dos usuários que receberão o compartilhamento e o nível de acesso que cada usuário terá em relação à playlist compartilhada.

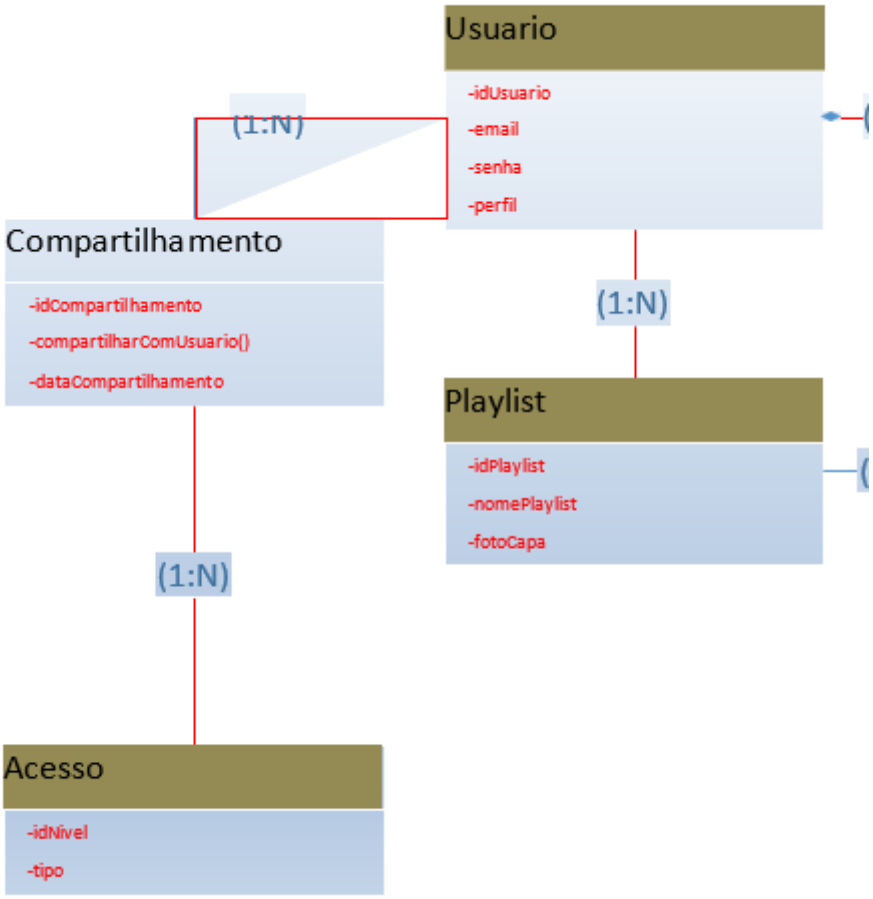

**Figura 5:** UML das entidades da História de Usuário 02.

A **Figura 06** apresenta o Modelo Entidade-Relacionamento (MER) da história de usuário 03. A entidade **Usuário** contém as chaves estrangeiras da entidade **Compartilhamento**. A entidade **Usuário** fornece o e-mail da pessoa que receberá a playlist. A entidade **Nível de Acesso** se relaciona com a entidade **Compartilhamento** em um relacionamento de 1

, onde estão armazenadas as informações do **idCompartilhamento**.

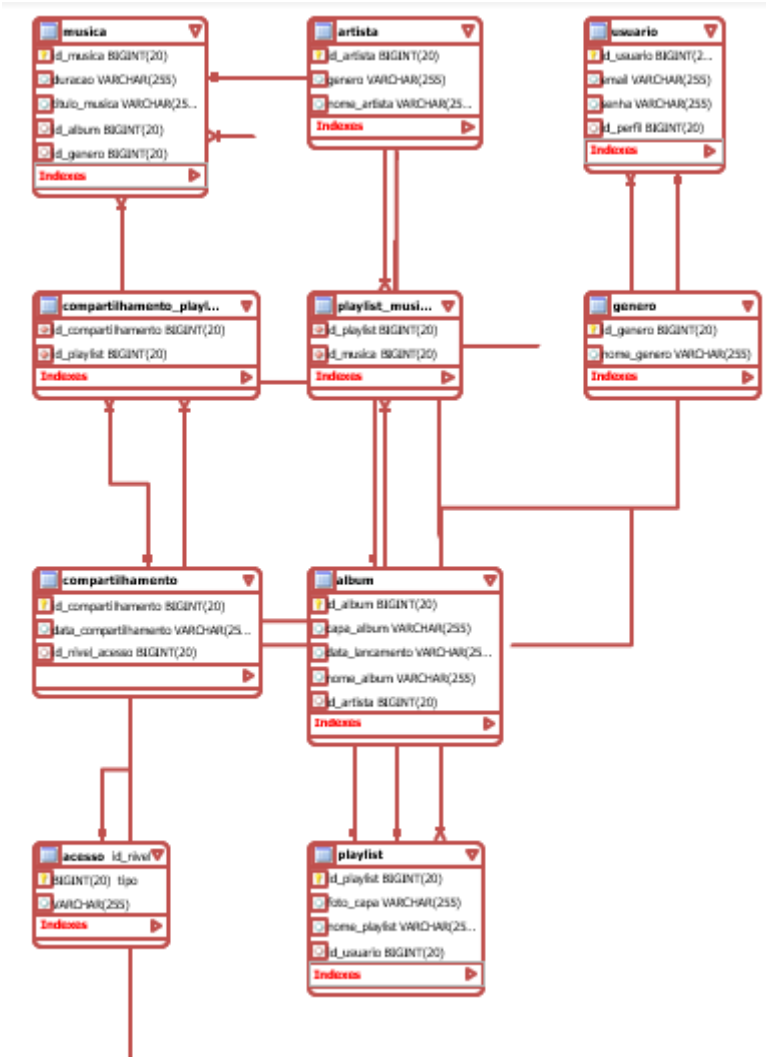


**Figura 6:** MER das entidades da História de Usuário 03.

## 1.3. História de Usuário 04

Como usuário do Sistema de Música, desejo criar uma playlist. Para isso, quero adicionar a primeira música a uma nova playlist, nomeá-la e incluir outras músicas.

A **Figura 07** apresenta o diagrama UML da história de usuário 04. A entidade **Usuário** se relaciona com a entidade **Playlist**, que, por sua vez, se relaciona com a entidade **Música**. Esse relacionamento permite que o sistema receba o título da música, o nome do artista e o nome da playlist.

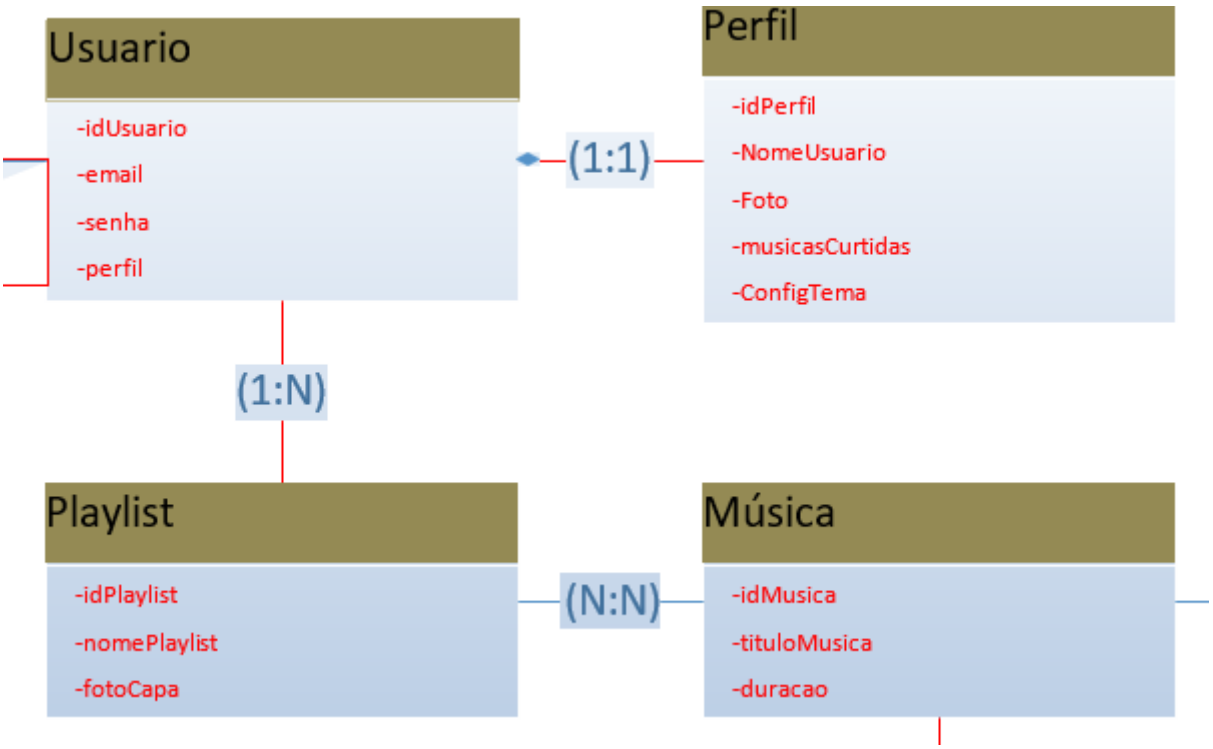


**Figura 7:** UML das entidades da História de Usuário 04.

A **Figura 08** apresenta o Modelo Entidade-Relacionamento (MER) da história de usuário 04. A entidade **Usuário** contém uma chave estrangeira que referencia a entidade **Playlist**, a qual, por sua vez, contém uma chave estrangeira que aponta para a entidade **Música**. A entidade **Usuário** fornece o título da música e o nome da playlist a ser atribuída. O relacionamento entre a entidade **Usuário** e a entidade **Playlist** é de 1

, enquanto o relacionamento entre a entidade **Playlist** e a entidade **Música** é de N
, fornecendo informações sobre o título da música, duração e ID da música.

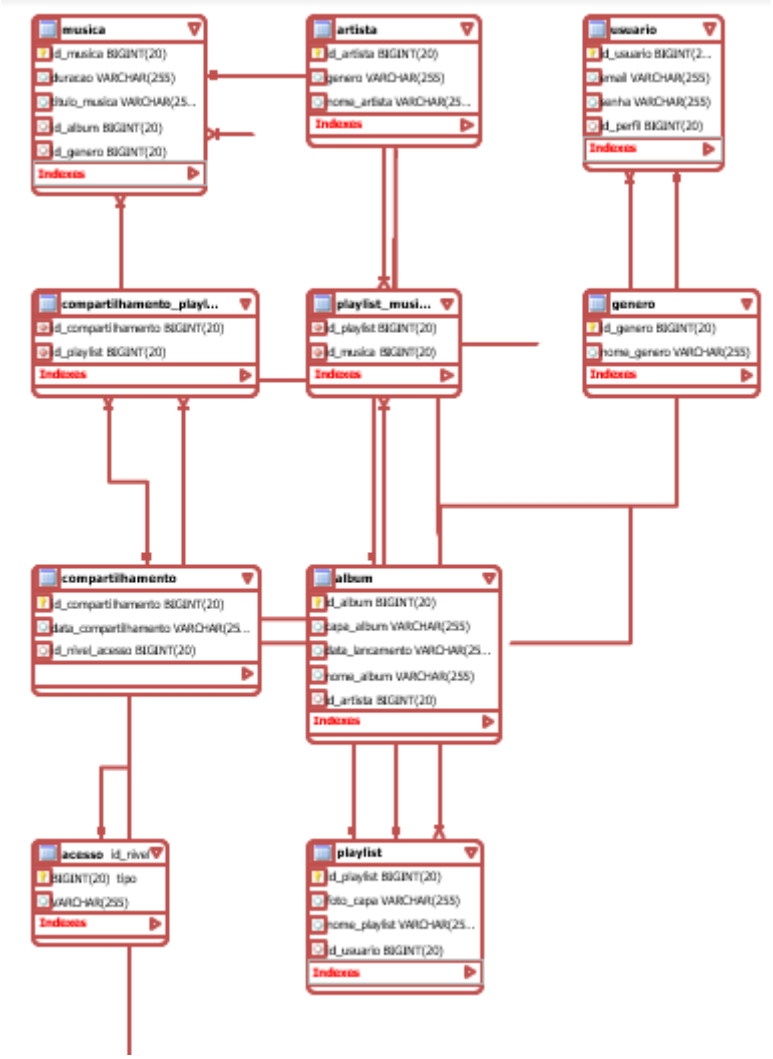


**Figura 8:** MER das entidades da História de Usuário 04.

## 1.4. História de Usuário 05

Como usuário do Sistema de Música, desejo encontrar artistas de um determinado gênero. Para isso, vou pesquisar na seção de gênero e visualizar os artistas que pertencem a esse gênero.

A **Figura 09** apresenta o diagrama UML da história de usuário 05. A entidade **Artista** se relaciona com a entidade **Gênero**, recebendo o gênero desejado e fornecendo os nomes dos artistas correspondentes.

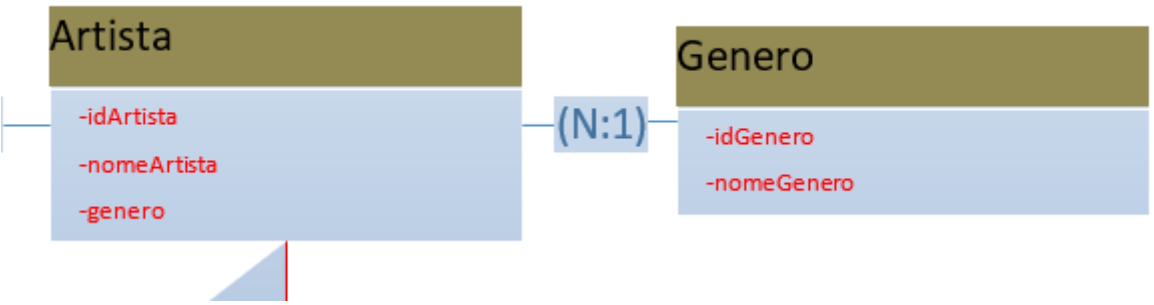


**Figura 9. UML das entidades da História de Usuário 05.**

A figura 10 apresenta o Modelo Entidade-Relacionamento (MER) da história de usuário 05. A entidade **Artista** contém uma chave estrangeira que referencia a entidade **Gênero**. Essa entidade **Gênero** fornece o nome dos artistas associados ao gênero solicitado. O relacionamento entre as entidades é de N:1, onde um mesmo gênero pode estar relacionado a vários artistas, enquanto cada artista pode pertencer a apenas um gênero.


**Figura 10. MER das entidades da História de Usuário 05.**

## 1. Codificação

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

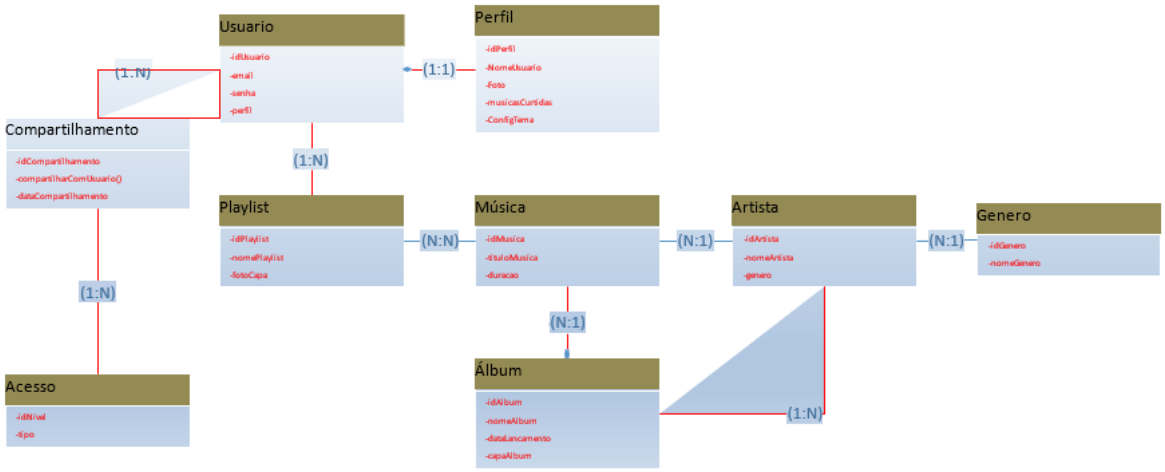


**Figura 11. Diagrama de classe do Sistema de música Sorriso Brilhante.**

### 1.1. Entidade Album

A entidade Album representa o relacionamento com o artista e as músicas.

```
1: @Entity
2: public class Album {
3: @Id
4: @GeneratedValue(strategy = GenerationType.IDENTITY)
5: @OneToMany(mappedBy = "album")
6: private List<Musica> musicas;
7: @ManyToOne
8: @JoinColumn(name = "idArtista")
9: private Artista artista;
10: }
```

**Figura 12. Código da entidade Album**

A entidade Artista representa o relacionamento com o Álbum.

```
1: @Entity
2: public class Artista {
3: @Id
4: @GeneratedValue(strategy = GenerationType.IDENTITY)
5: @ManyToMany(mappedBy = "artista")
6: private List<Album> albuns;
10: }
```

**Figura 13. Código da entidade Artista**

A entidade Compartilhamento representa o relacionamento com o a playlist a ser compartilhada e os níveis de acesso a serem concedidos.

```
1: @Entity
2: public class Compartilhamento {
3:@Id
4:@GeneratedValue(strategy = GenerationType.IDENTITY)
5: @ManyToOne
6: @JoinColumn(name = "idNivelAcesso")
7: private NivelAcesso nivelAcesso;
8: @ManyToMany
9: @JoinTable(name = "compartilhamento_playlist",
10: joinColumns = @JoinColumn(name = "idCompartilhamento"),
11: inverseJoinColumns = @JoinColumn(name = "idPlaylist"))
12: private List<Playlist> playlists;
13: }
```

**Figura 14. Código da entidade Compartilhamento**

A entidade Genero representa o relacionamento com as músicas.

```
1: @Entity
2: public class Genero {
3: @Id
4: @GeneratedValue(strategy = GenerationType.IDENTITY)
5: @OneToMany(mappedBy = "genero")
6: private List<Musica> musicas;
7: }
```

**Figura 15. Código da entidade Genero**

A entidade Musica representa o relacionamento com a playlist, o álbum e o gênero.

```
1: @Entity
2: public class Musica {
3: @Id
4: @GeneratedValue(strategy = GenerationType.IDENTITY)
5: @ManyToMany(mappedBy = "musicas")
6: private List<Playlist> playlists;
7: @ManyToOne
8: @JoinColumn(name = "idAlbum")
9: private Album album;
10: @ManyToOne
11: @JoinColumn(name = "idGenero")
12: private Genero genero;
13: }
```

**Figura 16. Código da entidade Genero**

A entidade NivelAcesso representa o relacionamento direto com o compartilhamento.

```
1: @Entity
2: public class NivelAcesso {
3: @Id
4: @GeneratedValue(strategy = GenerationType.IDENTITY)
5: @OneToMany(mappedBy = "nivelAcesso")
6: private List<Compartilhamento> compartilhamentos;
7: }
```

**Figura 17. Código da entidade NivelAcesso**

A entidade Perfil representa o relacionamento direto com o usuário.

```
1: @Entity
2: public class Perfil {
3: @Id
4: @GeneratedValue(strategy = GenerationType.IDENTITY)
5: @OneToMany(mappedBy = "perfil")
6: private Usuario usuario;
7: }
```

**Figura 18. Código da entidade Perfil**

A entidade Playlist representa o relacionamento com o usuário e com as músicas.

```
1: @Entity
2: public class Playlist {
3: @Id
4: @GeneratedValue(strategy = GenerationType.IDENTITY)
5: @ManyToOne
6: @JoinColumn(name = "idUsuario")
7: private Usuario usuario;
8: @ManyToMany
9: @JoinTable(name = "playlist_musica",
10: joinColumns = @JoinColumn(name = "idPlaylist"),
11: inverseJoinColumns = @JoinColumn(name = "idMusica"))
12: private List<Musica> musicas;
13: }
```

**Figura 19. Código da entidade Playlist**

A entidade Usuario representa o relacionamento com o perfil e as playlists.

```
1: @Entity
2: public class Usuario {
3: @Id
4: @GeneratedValue(strategy = GenerationType.IDENTITY)
5: @OneToOne(cascade = CascadeType.ALL)
6: @JoinColumn(name = "idPerfil")
7: private Perfil perfil;
8: @OneToMany(mappedBy = "usuario", cascade =
9: CascadeType.ALL)
10: private List<Playlist> playlists;
11: }
```

**Figura 20. Código da entidade Usuario**

## 2. Banco de dados


**Figura 21. Modelo Entidade Relacionamento do Sistema de Música**

## 4. Conclusão

O código tem um bom potencial, mas ainda apresenta alguns erros. Com esforço e futuras atualizações, acredito que o software poderá se tornar tão eficaz quanto os reprodutores de música mais populares.

## Referências


Boulic, R. e Renault, O. (1991) “Hierarquias 3D para Animação”, em: Novas Tendências em Animação e Visualização, Editado por Nadia Magnenat-Thalmann e Daniel Thalmann, John Wiley & Sons Ltd., Inglaterra.

Dyer, S., Martin, J. e Zulauf, J. (1995) “Documento sobre Captura de Movimento”, disponível em http://reality.sgi.com/employees/jam_sb/mocap/MoCapWP_v2.0.html, dezembro.

Holton, M. e Alexander, S. (1995) “Modelagem Celular Suave: Uma Técnica para a Simulação de Materiais Não-Rígidos”, em: Gráficos Computacionais: Avanços em Ambientes Virtuais, R. A. Earnshaw e J. A. Vince, Inglaterra, Academic Press Ltd., p. 449-460.

Knuth, D. E. (1984) O TeXbook, Addison Wesley, 15ª edição.

Smith, A. e Jones, B. (1999). Sobre a Complexidade do Cálculo. Em: Avanços em Ciência da Computação, p. 555–566. Publishing Press.