Sistema de Música Luz

Juliana Seemann, Luana S. Azevedo

Engenharia de Software
Universidade da Região de Joinville (UNIVILLE) – Joinville, SC – Brazil
Juliana.seemann@univille.br, luana.azevedo@univille.br

1. Introdução

Nosso aplicativo de música foi concebido para oferecer uma experiência musical altamente personalizada, acessível a qualquer momento e em qualquer lugar. Com uma extensa biblioteca que abrange milhões de músicas e playlists cuidadosamente curadas, você pode explorar com facilidade uma diversidade de gêneros, artistas e sonoridades com um simples toque. Seja para revisitar suas faixas favoritas, montar a trilha sonora ideal para o seu dia ou descobrir novos artistas, nosso app torna sua vivência musical única. A interface é simples e intuitiva, com recomendações feitas sob medida para o seu gosto, além da possibilidade de baixar músicas para ouvir offline, garantindo que você tenha total controle da sua playlist, seja em casa, no trabalho ou durante seus deslocamentos.

2. Requisitos Funcionais

Os requisites funcionais do Sistema XPO serão apresentados em forma de história de usuário.

Requisitos Funcionais história usuário 01:

1. **Adicionar Artista:**
   - O sistema deve permitir que eu adicione um novo artista, informando o nome, nacionalidade e gênero musical.
2. **Listar Artistas:**
   - Eu quero visualizar todos os artistas cadastrados na minha biblioteca.
3. **Adicionar Álbum:**
   - O sistema deve permitir que eu adicione um álbum, informando o nome e selecionando o artista associado.
4. **Adicionar Música:**
   - Eu quero poder adicionar uma nova música a um álbum, informando o nome da música, duração, artista e álbum.
5. **Listar Músicas:**
   - O sistema deve permitir que eu visualize todas as músicas disponíveis na minha biblioteca.
6. **Editar Música:**
   - Se necessário, eu quero editar os detalhes de uma música existente, como o nome ou duração.
7. **Excluir Música:**
   - O sistema deve permitir que eu exclua músicas da minha coleção.
8. **Validação de Dados:**

- O sistema deve garantir que todos os dados inseridos sejam válidos, como nome do artista, álbum e música.

9. **Feedback ao Usuário:**
    - Após cada ação (adição, edição, exclusão), o sistema deve fornecer um feedback claro confirmando o sucesso da operação.

Requisitos Funcionais história usuário 02:

10. **Criar Playlist:**
    - O sistema deve permitir que eu crie uma nova playlist, informando o nome e associando-a a um perfil dono.
11. **Adicionar Músicas à Playlist:**
    - Eu quero poder adicionar músicas a uma playlist existente, escolhendo-as a partir da minha biblioteca.
12. **Listar Playlists:**
    - O sistema deve permitir que eu visualize todas as minhas playlists.
13. **Compartilhar Playlist:**
    - Eu quero compartilhar uma playlist com outros usuários, selecionando a conta do destinatário.
14. **Visualizar Compartilhamentos:**
    - O sistema deve permitir que eu veja todas as playlists que compartilhei e as que recebi.
15. **Excluir Compartilhamento:**
    - O sistema deve permitir que eu exclua um compartilhamento de playlist.
16. **Gerenciar Planos:**
    - Eu quero visualizar e escolher entre diferentes planos disponíveis, para personalizar minha experiência no sistema.
17. **Validação de Dados:**
    - O sistema deve validar que o nome da playlist não esteja vazio e que os artistas e músicas associados sejam válidos.
18. **Feedback ao Usuário:**
    - O sistema deve fornecer confirmação após cada ação relacionada a playlists e compartilhamentos, garantindo que eu esteja ciente das alterações realizadas.

2.1. História de Usuário 01

Como um usuário do sistema XPO, eu quero gerenciar minha coleção de músicas e artistas para ter uma biblioteca musical rica e diversificada. Quando eu adiciono um novo artista, desejo informar seu nome, nacionalidade e gênero musical. Além disso, quero poder criar álbuns associados a esses artistas, facilitando a organização das minhas faixas. Ao adicionar uma música a um álbum, preciso que o sistema me permita definir o nome e a duração da música, bem como escolher o artista e o álbum correspondentes. Quero visualizar todos os artistas, álbuns e músicas cadastrados, e ter a opção de editar ou excluir qualquer item da minha coleção. Também desejo que o sistema valide todas as informações inseridas, garantindo que não haja erros. Por fim, espero receber feedback

claro após cada ação, para saber que minha biblioteca está sempre atualizada e organizada.

O diagrama de classes apresentado na Figura 01 ilustra a estrutura e as interações entre as principais entidades do sistema, com foco nas classes Playlist, Música e Usuário. A classe Playlist é a entidade central, representando uma coleção de músicas organizadas pelo usuário. Dentro da Playlist, há um atributo que armazena uma lista de objetos da classe Música, indicando uma associação do tipo "Um para Muitos". Isso significa que uma Playlist pode conter várias músicas.

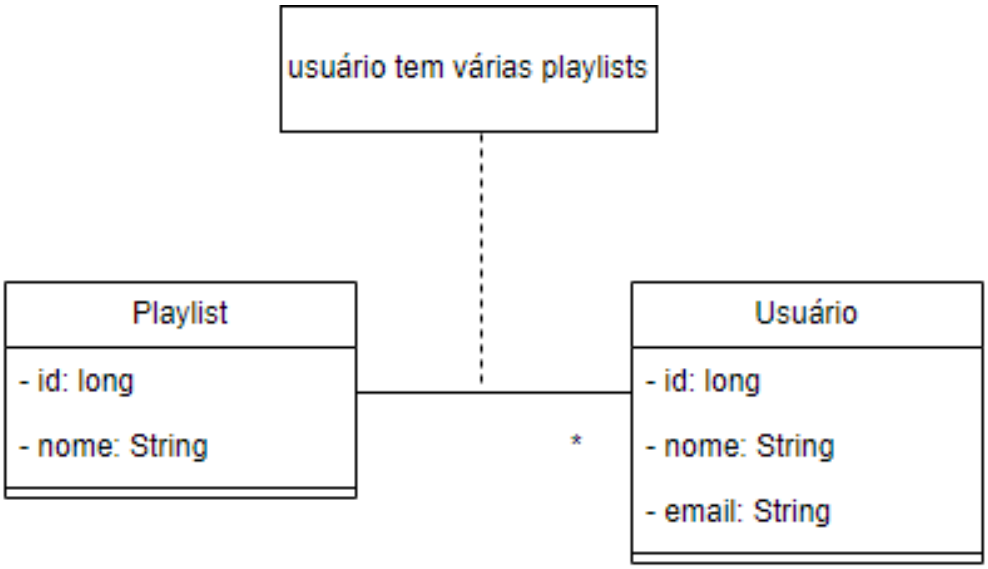

Figura 1. Diagrama de classe das entidades da História de Usuário 01.

A Figura 02 o diagrama de classe da história de usuário 01. A tabela DISCIPLINA_ALUNO contêm as chaves estrangeiras da tabela Disciplina e Aluno. Outras explicações importantes sobre as tabelas, chave única, etc.

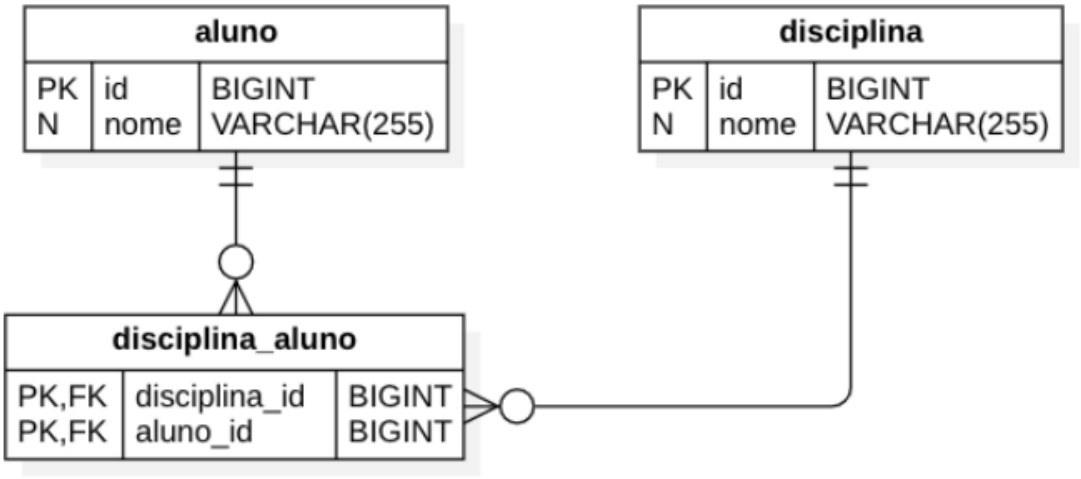

Figura 2. Modelo Entidade Relacionamento da História de Usuário 01.

2.2. História de Usuário 02

Como um usuário do sistema XPO, eu quero compartilhar minhas playlists com amigos para que possamos desfrutar de música juntos. Quando crio uma nova playlist, desejo informar seu nome e associá-la a um perfil específico. Quero adicionar músicas facilmente a essa playlist, escolhendo as faixas que mais gosto da minha coleção. Além disso, desejo ter a opção de visualizar todas as minhas playlists em um só lugar. Quando compartilhar uma playlist, preciso que o sistema me permita selecionar a conta do destinatário sem complicações. Quero também ver todas as playlists que compartilhei e as que recebi de outros usuários. Caso eu não queira mais uma playlist, desejo excluí-la de forma simples. O sistema deve validar que o nome da playlist não esteja vazio e que as músicas associadas sejam válidas. Por último, espero receber confirmações de que minhas ações foram bem-sucedidas, garantindo que minha experiência seja fluida e agradável.

3. Codificação

O diagrama UML descreve três entidades principais: Playlist, Música e uma representação conceitual de "álbum". A Playlist contém um identificador único (id) e um nome (`nome`), e pode armazenar várias músicas, representando uma relação "Um para Muitos" com a entidade Música. A entidade `Música`, por sua vez, tem atributos como id, nome, e duração, e pode ser parte de múltiplas playlists. Além disso, o diagrama indica que um álbum é composto por várias músicas, representando outra relação "Um para Muitos".

Em Java, essa estrutura pode ser implementada com a classe Playlist contendo uma lista de objetos Música, enquanto a classe Música pode ter uma associação opcional com uma entidade Álbum. Assim, o diagrama destaca como playlists são organizadas em coleções de músicas, e como as músicas também podem estar vinculadas a álbuns, refletindo um cenário típico de aplicativos de música. Abaixo pode-se ver a figura 03 que demonstra esta relação entre as entidades.

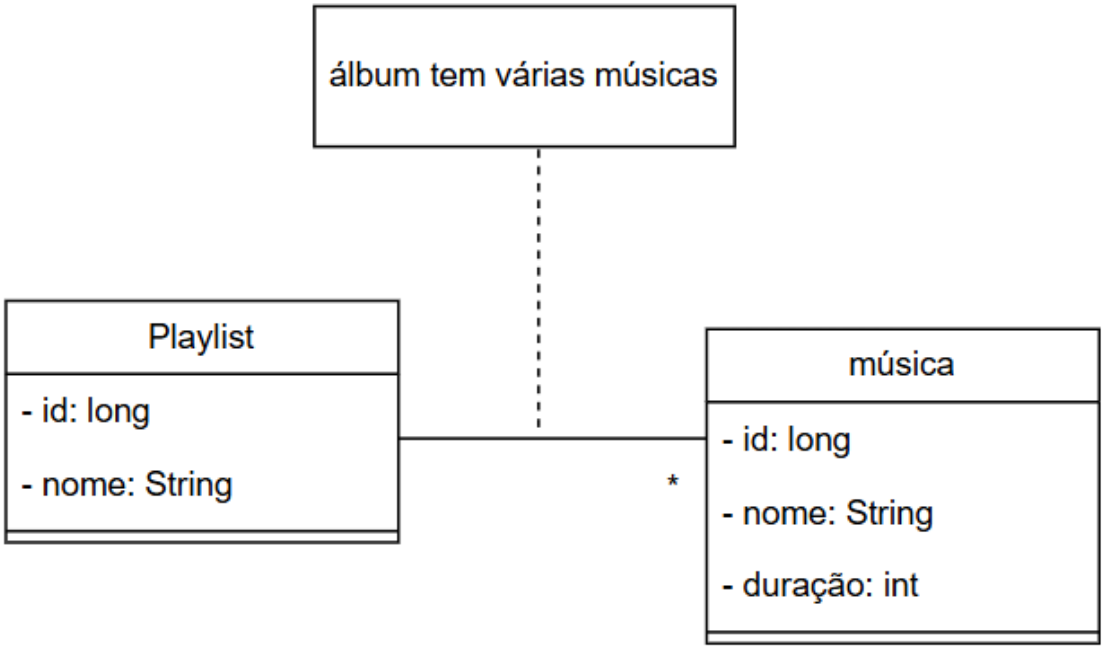

Figura 03. Diagrama de classe do Sistema de música Luz

3.1. Entidade Playlist

A entidade Playlistdefine uma tabela no banco de dados possui o campo id, que é a chave primária, gerada automaticamente. A classe contém listas de músicas e erfil_ouvinte, ambas com relações @ManyToMany, e um perfil_dono com relação @ManyToOne. Isso permite explorar as relações complexas entre playlists, músicas e perfis de usuários.

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

    @ManyToMany
    private List<Musica> musicas;
    @ManyToMany
    private List<Perfil> perfil_ouvinte;
    @ManyToOne
```

```
    private Perfil perfil_dono;
}
```

Figura 04. Código da entidade Playlist

3.2. Entidade Album

A entidade Album em Java representa uma tabela no banco de dados, possui o campo id, que é definido como a chave primária através da anotação @Id e é gerado automaticamente pelo banco de dados. A relação entre Album e Artista é definida pela anotação @ManyToOne, indicando que um artista pode ter vários álbuns, enquanto cada álbum está vinculado a um único artista. `@Data`

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

    @ManyToOne
    private Artista artista;
}
```

Figura 05. Código da entidade Album

3.2. Entidade Conta

A entidade Conta em Java representa uma tabela no banco de dados, possui o campo id, que é a chave primária da tabela e seu valor é gerado automaticamente pelo banco de dados. Além do id, a classe possui um campo `email`, que é armazenado na tabela como uma coluna. A anotação `@Column(unique = true)` garante que o valor do email seja único, evitando a duplicação de contas com o mesmo endereço de email.

```
import jakarta.persistence.*;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@Entity
```

```
public class Conta {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    @Column(unique = true)
    private String email;
}
```

Figura 06. Código da entidade Conta

4. Banco de dados

O diagrama UML para as entidades Conta, Album e Playlist mostra a estrutura e as relações entre elas no banco de dados. A entidade Conta gerencia usuários, identificados por um `id` único e um `email`, que deve ser exclusivo. A entidade `Album` representa álbuns musicais e estabelece uma relação `ManyToOne` com a entidade `Artista`, indicando que um artista pode ter vários álbuns, mas cada álbum pertence a um único artista. A entidade Playlist contém listas de músicas e perfis de ouvintes, configuradas como relações `ManyToMany`, permitindo que várias playlists incluam múltiplas músicas e sejam associadas a diversos perfis. Além disso, cada playlist tem um único perfil de dono, definido por uma relação `ManyToOne`.

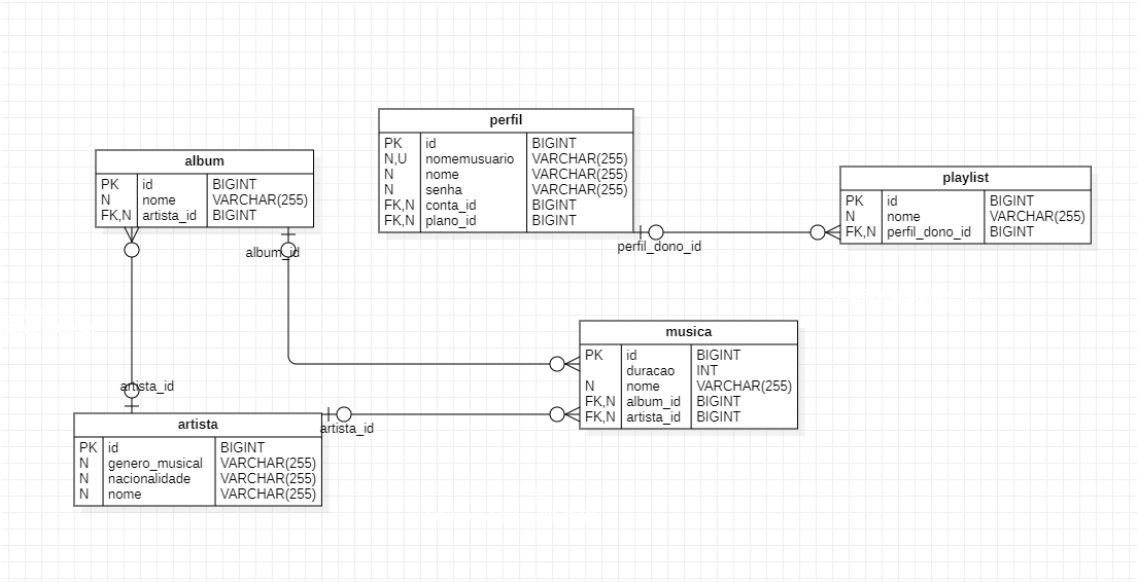

Figura 2. Modelo Entidade Relacionamento do Sistema de Música

4. Conclusão

O aplicativo de música oferece uma experiência personalizada que permite aos usuários explorar, organizar e desfrutar de suas músicas favoritas de forma ilimitada e sem interrupções. As histórias de usuários ilustram a importância de proporcionar um ambiente intuitivo e flexível para criar e gerenciar playlists, garantindo que o usuário tenha total controle sobre suas escolhas musicais. O diagrama de classes apresentado reflete essa estrutura, destacando as interações entre as entidades Playlist, Música e Usuário, e como essas relações viabilizam uma experiência musical contínua e adaptada aos gostos individuais. Assim, o aplicativo transforma o ato de ouvir música em uma atividade totalmente personalizada e acessível em qualquer momento e lugar.

5. Referências.

STARUML. *Entity Relationship Diagram*. Disponível em: https://docs.staruml.io/working-with-additional-diagrams/entity-relationship-diagram#create-entity-relationship-diagram. Acesso em: 25 set. 2024.