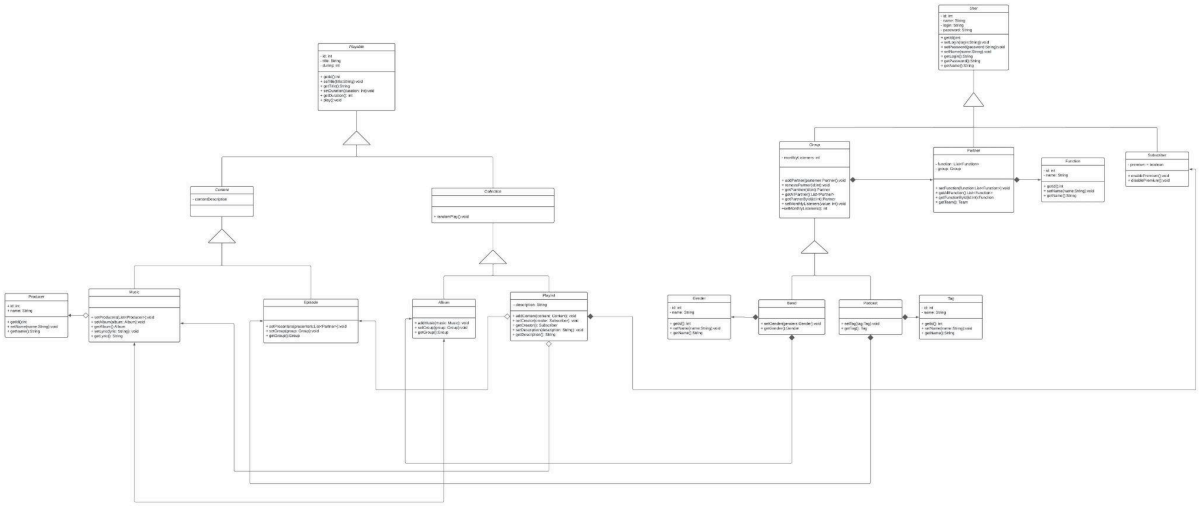
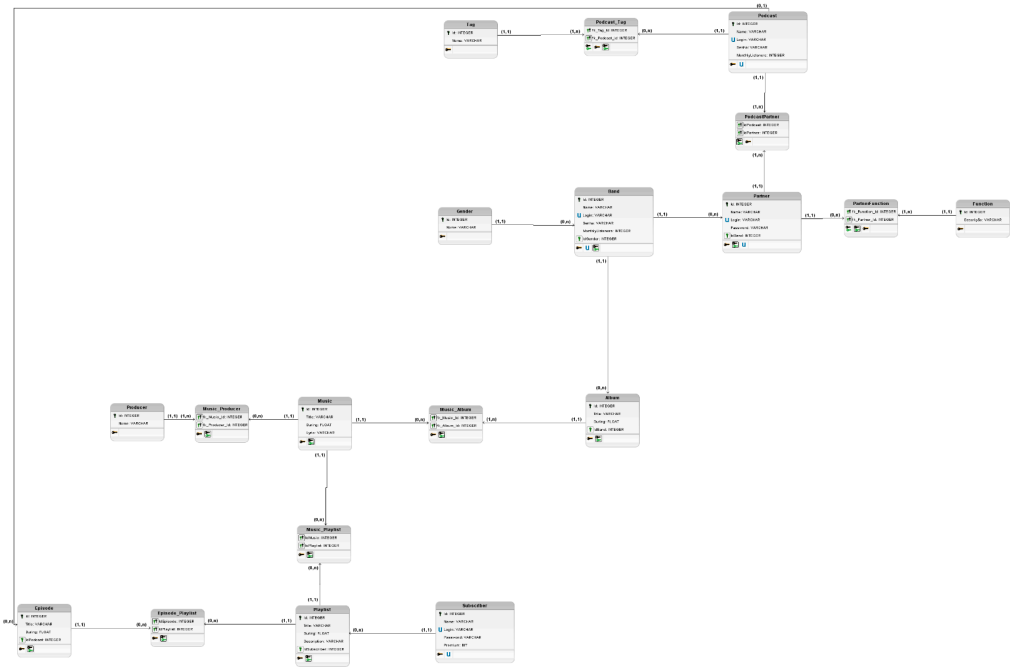
Modelagem O.O. e E.R. B.D. Spotify

Estudante: Jeferson Leandro de Oliveira dos Reis Santana

Modelagem Orientada a Objetos



Modelagem Lógica Relacional



Explicação do Modelo Orientado a Objeto

O modelo foi dividido em duas hierarquias: a hierarquia de usuários e a hierarquia de conteúdos.

Hierarquia de Usuários

Classes:

User, Partner, *Group*, Function, Subscriber, Band, Podcast,Gender e Tag.

Relacionamentos:

User é a classe abstrata base que fornece uma interface para Partner, Group e Subscriber, essa interface é necessária para todos que precisam estar na plataforma, seja para consumir seja para inserir conteúdo, tais como login e senha.

Group é a subclasse abstrata de User que fornece uma interface específica para grupos que precisam estar na plataforma, tais como Band e Podcast. O Spotify além de música, suporta também podcasts, então, as classes concretas de Group podem ser bandas ou equipes de podcasts.

Partner é a subclasse concreta que representa um integrante, seja de uma banda, seja de uma equipe de podcast. Independente do grupo que um objeto partner pertença, ele sempre terá uma função, Function, podendo, portanto, ser um instrumentista, escritor, produtor, apresentador ou qualquer outra função.

Band e Podcast são grupos, Groups, cada um produz um tipo de conteúdo diferente, respectivamente álbuns de música e episódios de podcast. Portanto, objetos Band tem sempre um gênero, Gender, para caracterizar o seu estilo, tais como Indie, Rock, Rap, Pop e etc. Igualmente, objetos Podcast sempre tem tags, Tag, que definem a classe de conteúdo de seus episódios, tais como Life Style, Estudos, Academia, Cultura Pop e etc.

Subscriber são usuários comuns do spotify, consumidores.

Hierarquia de “Playáveis”

Classes:

Playable, Music, Episode, *Collection*, *Content,* Album, Playlist e Producer.

Relacionamentos:

Playable é a classe abstrata que fornece uma interface para todos os objetos que são “playáveis”, isto é, que a gente pode dar play, como qual subclasse concreta de *Content,* como Music e Episode, ou qualquer subclasse concreta de *Collection*, como Album e Playlist.

Music e Episode são conteúdos encontrados no spotify. Music representa uma música, portanto, que no spotify, está sempre associada a um Album, ou Ep ou Single, mas nesse modelo optamos por simplificar e representar apenas álbuns. Episode representa um episódio de podcast. Episódios estão diretamente associados a um grupo de podcast.

Playlist e Album são coleções de Content e Music, respectivamente. No modelo do spotify, nenhuma música está “solta” no perfil de uma banda, ela sempre está dentro de um álbum, single ou ep. Playlists são, naturalmente, conjuntos de objetos reproduzíveis e não necessariamente de músicas, então num objeto Playlist podem haver objetos Music e Episode também, já que ambos são Content.

Producer representa uma Gravadora/Produtora. Uma produtora pode não produzir todo um álbum de uma banda, mas pode produzir algumas músicas dele, assim como num mesmo álbum podem haver músicas produzidas por produtoras diferentes, além de uma mesma música ser produzida por duas ou mais produtoras. Nesse modelo consideramos que a produtora não precisar ter um perfil no Spotify, ela precisa apenas estar cadastrada no Banco.

Mapeamento OO => ER

A forma de mapeamento das classes foi o mapeamento somente das classes concretas para tabelas, onde os atributos das classes pai abstratas são repetidos nas tabelas que mapeiam as classes concretas. Essa foi vista como a melhor opção em relação às outras duas formas visto que é a que menos fere a modelagem conceitual, não se afastando da modelagem original e por fazer menos junções possibilitaria um desempenho melhor da aplicação real, caso implementada a partir desse modelo.

Visto que, os comportamentos podem ser aplicados como atributos da tabela ou triggers/funções a depender do tipo de campo que acessam, como no modelo não há métodos nos quais precisam ser realizados cálculos sobre atributos para gerar informações, isto é, todos os métodos acessam diretamente os valores de atributos seja para ler ou escrever, então todos podem ser mapeados como funções.