U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Bases de Dados

Plataforma Spotify

António Cruz, Joana Ramos e João Malheiro

MIEICT1G3 Bases de Dados, 2017/2018

Descrição

Este projeto retrata um serviço de streaming de música, à semelhança do Spotify.

Para usufruírem da plataforma todos os **utilizadores** têm que se registar com o seu nome, email (que não se pode repetir) e também com o seu **país** de origem e uma foto (opcional). É atribuído ao utilizador um ID. Todos os utilizadores podem guardar (porque são, à partida, imutáveis) **músicas** e **álbuns** na sua conta e seguir (acompanhar a atividade) **playlists**, **artistas** e outros utilizadores. Interessa saber qual a música que um dado utilizador está a ouvir no momento, e de todas as músicas que ouviu é relevante saber a data (incluindo o instante temporal) e se a passou à frente.

Um **artista** (referido no singular podendo ser na verdade um conjunto de artistas, tais como uma banda, que partilham apenas uma conta profissional), tem um ID, nome artístico, uma biografia, país de origem, **género**(s) e pelo menos uma música.

Uma **música** é representada pelo seu nome, ID, duração, género(s), artista(s) que a interpretam e pelo álbum/álbuns em que se inclui.

Um **álbum** é composto por pelo menos uma música, e interessa guardar-se a informação relativa ao seu nome, ID, data de lançamento e capa do álbum.

Uma ***playlist*** é também composta por pelo menos uma música, tem um nome e ID e está associada a um utilizador responsável pela

sua criação e aos utilizadores que a seguem. Pode ainda ser pública (todas as pessoas conseguem encontrá-la) ou privada (apenas o seu criador tem acesso a ela).

Um **género** é representado pelo seu nome e ID.

Diagrama UML

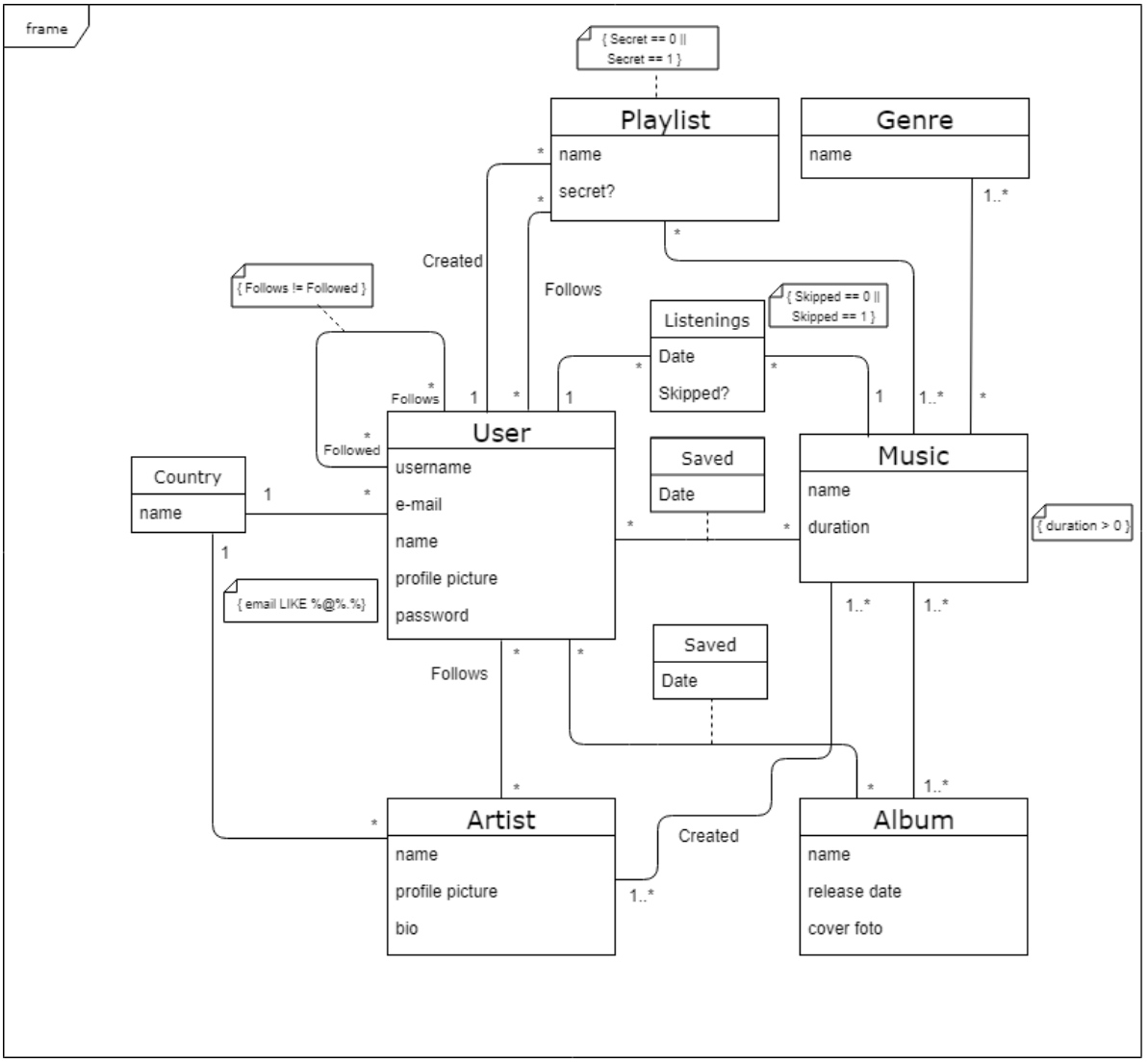
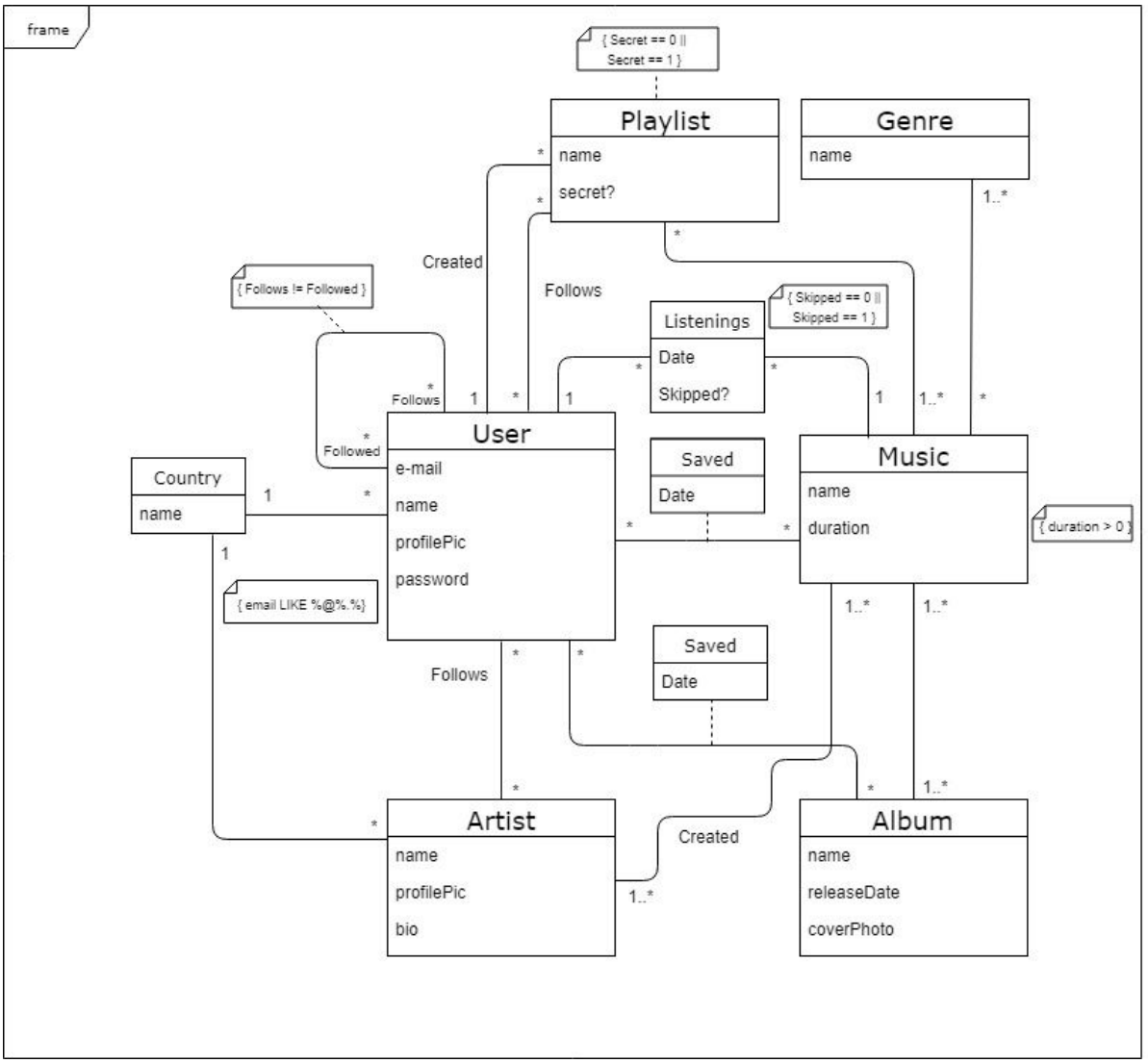


Diagrama UML (nomes corrigidos)



Modelo Relacional

Album(id,name,releaseDate,coverPhoto)

Artist(id,name,profilePic,bio,country->Country)

BelongsAlbum(music->Music,album->Album)

BelongsGenre(music->Music,genre->Genre)

BelongsPlayList(music->Music,playlist->Playlist)

Country(id,name)

CreatedMusic(artist->Artist,music->Music)

FollowArtist(user->User,artist->Artist)

FollowPlaylist(user->User,playlist->Playlist)

FollowUser(followed->User, follows->User)

Genre(id,name)

Listenings(id,date,skipped,user->User,music->Music)

Music(id,name,duration)

Playlist(id,name,secret,creator->User)

SavedAlbum(user->User,album>Album,date)

SavedMusic(user->User,music->Music,date)

User(id, email,name,profilePic,password,country->Country)

Dependências Funcionais

Album(id,name,releaseDate,coverPhoto)
FDs:
id->name,releaseDate,coverPhoto
name->id,releaseDate,coverPhoto

Chaves da relação: {id}, {name}

Artist(id,name,profilePic,bio,country->Country)
FDs:
id->name,profilePic,bio,country
name->id,profilePic,bio,country

Chaves da relação: {id}, {name}

Country(id,name)
FDs:
id->name
name->id

Chaves da relação: {id, name}

Genre(id,name)
FDs:
id->name
name->id

Chaves da relação: {id, name}

Listenings(id,date,skipped,user->User,music->Music)
FDs:
id->date,skipped,user,music

user,music,date -> id, skipped

Chaves da relação: {id}, {user, music, date}

Music(id,name,duration)
FDs:
id->name,duration

Chaves da relação: {id}

Playlist(id,name,secret,creator->User)
FDs:
id->name,secret,creator
name, creator -> id,secret

Chaves da relação: {id}, {name, creator}

SavedAlbum(user->User,album>Album,date)
FDs:
user,album->date

Chaves da relação: {user, album}

SavedMusic(user->User,music->Music,date)
FDs:
user,music -> date

Chaves da relação: {user, music}

User(id, email,name,profilePic,password,country->Country)
FDs:
id->email,name,profilePic,password,country
email->id,name,profilePic,password,country

Chaves da relação: {id}, {email}

Todas as dependências funcionais das relações que se seguem são **triviais**.

BelongsAlbum(music->Music,album->Album)

BelongsGenre(music->Music,genre->Genre)

BelongsPlayList(music->Music,playlist->Playlist)

CreatedMusic(artist->Artist,music->Music)

FollowArtist(user->User,artist->Artist)

FollowPlaylist(user->User,playlist->Playlist)

FollowUser(followed->User, follows->User)

Formas normais

Com a informação anterior podemos concluir que todas as relações do modelo estão na *Boyce-Codd Normal Form* uma vez que o lado esquerdo de todas as dependências funcionais não triviais são super chaves das relações correspondentes. Assim sendo, podemos dizer que também estão na 3ª Forma Normal.

Restrições

Na criação da base de dados foram implementadas restrições para garantir a manutenção da integridade da mesma.

➔ Restrição NOT NULL:

Considera-se que as relações à frente mencionadas não fariam sentido sem os atributos referidos, pelo que se manifesta a obrigatoriedade de estes existirem através da restrição NOT NULL:

Na classe **Album**: name, releaseDate

Na classe **Artist**: name, country

Na classe **Country**: name

Na classe **Genre**: name

Na classe **Listenings**: date, user, music

Na classe **Music**: name, duration

Na classe **Playlist**: name, secret, creator

Na classe **SavedAlbum**: date

Na classe **SavedMusic**: date

Na classe **User**: email, name, password, country

➔ Restrição chave - UNIQUE

Não podem existir dois álbuns com o mesmo nome pelo que o atributo name da respetiva relação é UNIQUE. Ao contrário dos álbuns, as músicas podem repetir nomes pelo que não foram implementadas com esta restrição.

Não podem existir dois artistas com o mesmo nome pelo que o atributo name da respetiva relação é UNIQUE.
Não podem existir dois países com o mesmo nome pelo que o atributo nome da respetiva relação é UNIQUE.
Não podem existir dois géneros musicais com o mesmo nome pelo que o atributo nome da respetiva relação é UNIQUE.
Não é possível um utilizador ouvir mais do que uma música ao mesmo tempo, pelo que o conjunto das colunas user, music e date na relação Listenings é UNIQUE.
Não é possível um utilizador criar duas playlists diferentes com o mesmo nome logo o conjunto de colunas name e creator na relação Playlist é UNIQUE.
Um utilizador não pode criar duas contas utilizando o mesmo email, portanto, este atributo é UNIQUE.

➔ Restrição chave - PRIMARY KEY

Os atributos mencionados à frente são chaves primárias das relações a que pertencem, pelo que foram implementados com a restrição PRIMARY KEY:
Na classe **Album**: id
Na classe **Artist**: id
Na classe **BelongsAlbum**: music, album
Na classe **BelongsGenre**: music, genre
Na classe **BelongsPlaylist**: music, playlist
Na classe **Country**: id
Na classe **CreatedMusic**: artist, music
Na classe **FollowArtist**: user, artist

Na classe **FollowPlaylist**: user, playlist

Na classe **FollowUser**: followed, follows

Na classe **Genre**: id

Na classe **Listenings**: id

Na classe **Music**: id

Na classe **Playlist**: id

Na classe **SavedAlbum**: user, album

Na classe **SavedMusic**: user, music

Na classe **User**: id

➔ Restrição de integridade referencial - REFERENCES

O atributo country da relação **Artista** e **User** é uma chave estrangeira que aponta para a relação **Country**. É o resultado da modelação de uma associação *many to one*.

A relação **Listenings**, criada para armazenar informação acerca da atividade de um utilizador em relação às músicas, contém duas chaves estrangeiras, uma para **User** e outra para **Music**.

A relação **Playlist** tem uma chave estrangeira, creator, que aponta para **User**, representa o único criador da playlist. É o resultado da modelação de uma associação *many to one*.

Da definição do modelo relacional resultaram várias relações extra das associações *many to many*. Estas relações têm chaves estrangeiras para as relações que associam. Este foi o caso das seguintes relações: **BelongsAlbum**, **BelongsGenre**, **BelongsPlaylist**, **CreatedMusic**, **FollowArtist**, **FollowPlaylist**, **FollowUser**, **SavedAlbum**, **SavedMusic**.

➔ CHECK

Um email deve ser do tipo %@%.%, esta restrição foi implementada através de CHECK.

O atributo secret da relação **Playlist** e o atributo skipped da relação **Listenings** devem ser usados como booleanos e, como tal, só podem ter o valor 0 ou 1. Esta restrição foi implementada através de CHECK.

O atributo duration da relação **Music** referente à duração de uma música tem que ser superior a 0. Esta restrição foi implementada através de CHECK.

Na classe **FollowUser** é utilizada a restrição CHECK para garantir que o utilizador seguido é diferente do seguidor (um utilizador não pode seguir-se a si próprio).

Interrogações

Número de seguidores de cada utilizador : interrogação que verifica a ligação entre **User** e **FollowUser** e mostra quantos seguidores cada utilizador possui.

Todas as músicas disponíveis na plataforma : lista todas as músicas presentes na base de dados (**Music**).

Música e respectivos artistas : verifica a ligação entre **Music**, **Artist** e **CreatedMusic**, listando as todas as músicas juntamente com os artistas criadores.

Músicas e respectivos skips : mostra todas as músicas que foram skipped e o número de vezes que o foram. A lista é agrupada por nome das músicas ordenada pelo número de skips (group by e order by).

Músicas ouvidas por utilizadores recentemente (recently played) : interrogação que verifica a ligação entre **Music**, **User** e **Listenings**. Mostra a última música ouvida por cada utilizador.

Utilizadores que ouviram uma playlist toda : verifica a ligação entre **User**, **Music**, **Playlist**, **BelongsPlaylist**, **Listenings** que através de uma **junção** consegue mostrar todos os users que ouviram uma playlist na sua íntegra.

O Utilizador 2 guardou pelo menos 80% das músicas que o utilizador 1 guardou : através da utilização de **vistas** e do operador **UNION**, esta interrogação é útil para sugerir amigos baseado nas preferências musicais de cada utilizador, por exemplo.

Música mais ouvida em cada país nos últimos 7 dias : esta interrogação utiliza **junção** e o operador **having** para saber qual a música mais ouvida em cada país da base de dados nos últimos 7 dias (comparação de data realizada através de julianday( )).

Utilizadores que guardaram musicas que deram Skip : interrogação que utiliza vistas e o operador **Union** para mostrar todas os utilizadores e as suas músicas guardadas (“saved”) mas não ouvidas.

Género favorito de cada user : interrogação que mostra os géneros favoritos de cada utilizador.

Gatilhos

seguirPlaylist e seguirPlaylist2 - não deixar um utilizador criador de uma playlist seguir a sua própria playlist. Após a relação **FollowPlaylist** ser criada ou actualizada, é testado se o utilizador que está a seguir a playlist é diferente do utilizador que a criou. Se for o mesmo é lançado um **erro** através de SELECT raise(rollback,' ');. O primeiro é **AFTER INSERT** e o segundo **AFTER UPDATE**.

listeningBeforeDate - gatilho com vista a impedir que a data de um listening seja anterior à de criação do álbum que contém a música ouvida. Utiliza julianday( ) para a comparação das datas bem como o operador **MIN**.

Tal como o primeiro gatilho, este também envia um **erro** através de SELECT raise(rollback,' ') no caso de se verificar a condição.

deleteFromUser e deleteFromPlaylist - o primeiro gatilho serve para "limpar" a base de dados antes de se eliminar um utilizador. Este gatilho elimina **FollowPlaylist**, **FollowArtist**, **FollowUser**, **Listenings**, **SavedAlbum**, **SavedMusic**, **Playlist** relacionados com o **User**. De seguida, o segundo gatilho aqui apresentado elimina **BelongsPlaylist** e **FollowPlaylist** antes de eliminar uma determinada **Playlist**. Ambos são **BEFORE DELETE**.