**Bases de Dados**

2019/2020



Biblioteca de Música

Grupo 402 (Turma 4 – Grupo 2)

Trabalho realizado por:

Diogo Samuel Gonçalves Fernandes – up201806250

Hugo Miguel Monteiro Guimarães - up201806490

Paulo Jorge Salgado Marinho Ribeiro – up201806505

**Descrição do tema**

O projeto escolhido consiste na gestão de uma biblioteca de música com funções semelhantes ao *Spotify*.

O utilizador necessita de uma inscrição para poder usufruir do serviço. Sendo assim é necessário armazenar o seu e-mail, username, password e outros dados pessoais. Após a inscrição, o mesmo fica apto a seguir outros utilizadores e entidades musicais. O utilizador também pode criar diversas playlists, isto é, uma compilação de diversas músicas. As playlists são criadas apenas por um utilizador. Não obstante o criador pode permitir que outros utilizadores adicionem músicas à sua fazendo com que esta se torne numa playlist colaborativa.

O utilizador também pode adicionar um determinado álbum, playlist ou música aos favoritos. No caso da música é possível identificar a data em que foi adicionada, estando também associada a pelo menos um estilo musical e agregada a um álbum. Além disso é mantido o registo sobre quanto tempo o utilizador esteve a ouvir música numa determinada data, através desta plataforma.

Cada álbum foi lançado por pelo menos uma entidade musical. Esta é constituída por mais do que um artista, caso seja uma banda, ou apenas por um artistacaso este atue a solo. Aentidade musicalvai possuir um nome, assim como a data da fundação.  É também possível saber o papeldesempenhado por cada um dos artistas, que a compõem, por exemplo se o artista é vocalista, guitarrista, entre outros. A partir dos álbuns que uma determinada entidade musical compôs é possível inferir as músicas da sua autoria.

É importante ainda referir que um artista pode ter mais do que uma entidade musical, uma vez que este pode pertencer a diversas bandas simultaneamente.

**Esquema Relacional**

**Pessoa** (idPessoa, nome, dataNascimento, codPostal, morada)

**Artista** (idArtista->Pessoa, inicioCarreira)

**Utilizador** (idUtilizador->Pessoa, email, username, password)

**Papel**( idPapel, atividade)

**EntidadeMusical** (idEntidadeMusical, nomeArtistico, imagem, dataFundacao, descricao)

**Album** (idAlbum, nome, capa, anoLancamento)

**Musica** (idMusica, idAlbum->Album, nome, duracao)

**EstiloMusical** (idEstiloMusical, nome)

**Playlist** (idPlaylist, criador->Utilizador, nome, imagem, dataCriacao, descricao, privada)

**Sessao** (idSessao, dataInicio)

**TempoOuvido** (idMusica->Musica, idSessao->Sessao, duracao)

**Desempenha** (idArtista->Artista, idPapel->Papel)

**Membro** (idArtista->Artista, idEntidadeMusical->EntidadeMusical)

**Possui** (idPapel->Papel, idEntidadeMusical->EntidadeMusical)

**Compoe** (idAlbum->Album, idEntidadeMusical->EntidadeMusical)

**Segue** (idUtilizador->Utilizador, idEntidadeMusical->EntidadeMusical)

**FavoritoAlbum** (idUtilizador->Utilizador, idAlbum->Album)

**FavoritoMusica** (idUtilizador->Utilizador, idMusica->Musica, data)

**FavoritoPlaylist** (idUtilizador->Utilizador, idPlaylist->Playlist)

**Colabora** (idUtilizador->Utilizador, idPlaylist->Playlist)

**MusicaEstilo** (idEstiloMusical->EstiloMusical, idMusica->Musica)

**UtilizadorSessao** (idUtilizador->Utilizador, idSessao->Sessao)

**Pertence** (idMusica->Musica, idPlaylist->Playlist)

**Seguir** (idUtilizador->Utilizador, idUtilizadorSeguido->Utilizador)

**Dependências Funcionais**

**Pessoa**

{idPessoa} -> {nome, dataNascimento, codPostal, morada}

{morada} -> {codPostal}

**Artista**

{idArtista} -> {inicioCarreira}

**Utilizador**

{idUtilizador} -> {email, username, password}

{email} -> {idUtilizador, username, password}

{username} -> {idUtilizador, email, password}

**Papel**

{idPapel} -> {atividade}

{atividade} -> {idPapel}

**EntidadeMusical**

{idEntidadeMusical} -> {nomeArtistico, imagem, dataFundacao, descricao}

**Album**

{idAlbum} -> {nome, capa, anoLancamento}

**Musica**

{idMusica} -> {idAlbum, nome, duracao}

**EstiloMusical**

{idEstiloMusical} -> {nome}

{nome} -> {idEstiloMusical}

**Playlist**

{idPlaylist} -> {criador, nome, imagem, dataCriacao, descricao, privada}

**Sessao**

{idSessao} -> {dataInicio}

**FavoritoMusica**

{utilizador, musica} -> {data}

**TempoOuvido**

{musica, sessao} -> {duracao}

**Formas Normais**

Segundo a Forma Normal Boyce-Codd (BCNF), para todas as relações não triviais do tipo A->B, é condição necessária A ser uma (super) chave.

Esta regra é unicamente violada na dependência funcional *{morada} -> {codPostal}* dado que a única chave da relação é *{idPessoa}*, pelo que *{morada}* não é (super) chave.

Para uma dependência cumprir a 3ª Forma Normal (3NF) é necessário que, ora cumpra a BCNF, ora B consista em atributos primos.

Deste modo, podemos analisar a dependência funcional anterior *{morada} -> {codPostal}*, que não cumpre a BCNF. Dado que *{codPostal}* não é um atributo primo, não pertence a nenhuma chave da relação. Logo, como não se verifica nenhuma das condições mencionadas acima, conclui-se que esta Dependência Funcional é a única que viola a 3ª Forma Normal, uma vez que todas as outras cumprem a BCNF e por isso cumprem também a 3NF.

**Restrições**

Para manter a integridade dos dados armazenados utilizamos uma série de restrições na base de dados. Utilizamos a restrição de chave PRIMARY KEY e UNIQUE, assim como CHECK e a restrição NOT NULL.

Em cada uma das relações foram utilizadas também chaves estrangeiras para garantir a integridade referencial, no modo ON DELETE SET NULL ON UPDATE CASCADE.

**Pessoa**

* Cada Pessoa possui um ID único, que é a sua PRIMARY KEY
* O nome de uma Pessoa é NOT NULL

**Utilizador**

* Cada Utilizador possui um ID único que é uma chave estrangeira para Pessoa e a sua PRIMARY KEY
* O username de cada pessoa, assim como o seu email, é UNIQUE
* O email, username e password são NOT NULL uma vez que é necessária a informação dos mesmos para o utilizador poder usar a plataforma
* A password tem de ter pelo menos 8 carateres por motivos de segurança

**Artista**

* Cada Artista possui um ID único que também é uma chave estrangeira para Pessoa e a sua PRIMARY KEY

**Papel**

* Um Papel (referente a uma habilidade, como por exemplo Guitarrista ou Vocalista) tem de ser NOT NULL e UNIQUE possuindo um ID único que é PRIMARY KEY

**EntidadeMusical**

* Cada Entidade Musical possui um ID único que é a sua PRIMARY KEY
* O nome da Entidade Musical é NOT NULL

**Album**

* O nome do álbum é NOT NULL e possui um ID único que é a sua PRIMARY KEY

**Musica**

* Cada Música possui um ID único, que é a sua PRIMARY KEY
* O nome da Música, assim como a sua duração é NOT NULL
* Existe uma chave estrangeira que permite saber o álbum a que pertence a música
* A duração de uma música tem de ser maior do que 0 segundos.

**EstiloMusical**

* O nome do Estilo Musical é NOT NULL e UNIQUE e cada Estilo Musical possui um ID único, que é a sua PRIMARY KEY

**Playlist**

* Cada Playlist possui um ID único, que é a sua PRIMARY KEY
* O nome de uma Playlist, assim como a sua data de criação e o atributo que indica se uma playlist é privada ou não são NOT NULL.
* idUtilizador é uma chave estrangeira para o criador da Playlist

**Sessao**

* Cada Sessão possui um ID único, que é a sua PRIMARY KEY
* Uma Sessão tem uma data de inicio NOT NULL

**FavoritoMusica**

* As PRIMARY KEYS que compõem FavoritoMusica são o id da Música adicionada às Favoritas e o id do Utilizador que a adicionou (ambas são chaves estrangeiras)
* Uma Música é adicionada a

**TempoOuvido**

* A duração do tempo ouvido tem de ser maior do que 0 segundos.
* idMusica e idSessao são chaves estrangeiras para a música que foi ouvida durante a sessão.

**Interrogações**

Segue-se uma lista de interrogações à base de dados, que consideramos pertinentes tendo em conta o seu contexto. Esta lista encontra-se ordenada, pelo que à interrogação 1 corresponde ao código SQL do ficheiro int1.sql, e o mesmo para as restantes. Em todas as interrogações são utilizados os operadorres SELECT e FROM, para listar os atributos desejados.

**Interrogação 1**

* **Descrição:** Lista o tempo total de audição de música para cada utilizador, em segundos.
* **Explicação:** Recorremos à junção das tabelas UtilizadorSessao, Sessao e TempoOuvido, em função do atributo “idSessao” e utilizando de seguida o operador SUM para contabilizar o tempo total de música ouvido por cada utilizador, AS para modificar o nome apresentado (tempoTotal). Terminamos com um GROUP BY, que permite listar o tempo ouvido para cada utilizador.
* **Operadores:** SUM, NATURAL JOIN, GROUP BY.

**Interrogação 2**

* **Descrição:** Apresenta o número de Estilos Musicais favoritados por cada Utilizador.
* **Explicação:** Recorremos à junção das tabelas FavoritoMusica, Utilizador (em função do atributo idUtilizador), seguida da junção da tabela resultante com a tabela MusicaEstilo (em função do atributo idMusica) e utilizando de seguida o operador COUNT para contabilizar o número de EstilosMusicais favoritados pelo utilizador. Terminamos com um GROUP BY, que permite listar o número de estilos musicais favoritados por cada utilizador.
* **Operadores:** COUNT, DISTINCT, AS, NATURAL JOIN, GROUP BY.

**Interrogação 3**

* **Descrição:** Lista os artistas que são vocalistas, que nasceram entre 1970 e 1980
* **Explicação:** Começamos por realizar uma sub-query para listar todos os idPessoa dos Artistas que desempenham a função de vocalista. Unimos esta tabela com a tabela Pessoa, para obter os seus dados, aplicando um WHERE que filtra aqueles que nasceram entre 1970 e 1980, e por fim é utilizado um ORDER BY para apresentar os tuplos por ordem crescente da data de nascimento.
* **Operadores:**  AS, NATURAL JOIN, WHERE, AND, ORDER BY.

**Interrogação 4**

* **Descrição:** Apresenta o Top10 das músicas mais favoritadas.
* **Explicação:** Realizamos a junção entre as tabelas FavoritoMusica e Musica (em função do atributo idMusica), utilizando um COUNT e GROUP BY para contabilizar o número de vezes que cada música foi favoritada, seguindo-se um ORDER BY com DESC e LIMIT para listar por ordem decrescente as 10 músicas mais favoritadas.
* **Operadores:** COUNT, AS, NATURAL JOIN, GROUP BY, ORDER BY, DESC, LIMIT.

**Interrogação 5**

* **Descrição:** Lista todos os pares de seguidores que se seguem reciprocamente.
* **Explicação:**
* **Operadores:** AND, NATURAL JOIN, AS, JOIN, WHERE.

**Interrogação 6**

* **Descrição:** Lista todos os pares de Utilizadores e Entidades Musicais cujo Utilizador adicionou aos favoritos todos os álbuns compostos pela Entidade Musical.
* **Operadores:** COUNT, AS, NATURAL JOIN, GROUP BY, ORDER BY.

**Interrogação 7**

* **Descrição:** Apresenta o Estilo Musical predominante em cada Entidade Musical, deduzido a partir dos Estilos Musicais das músicas da sua autoria.
* **Operadores:** AS, NATURAL JOIN, COUNT, JOIN, GROUP BY.

**Interrogação 8**

* **Descrição:** Lista todos os pares de utilizadores cujo tempo de audição de uma dada música é superior a 5 minutos.
* **Operadores:** SUM, AS, NATURAL JOIN, JOIN, GROUP BY, USING, WHERE, ORDER BY.

**Interrogação 9**

* **Descrição:** Lista todos os utilizadores que já ouviram na totalidade todas as músicas de todos os álbuns de uma entidade musical
* **Operadores:** SUM, AS, NATURAL JOIN, JOIN, GROUP BY, USING, WHERE, ORDER BY.

**Interrogação 10**

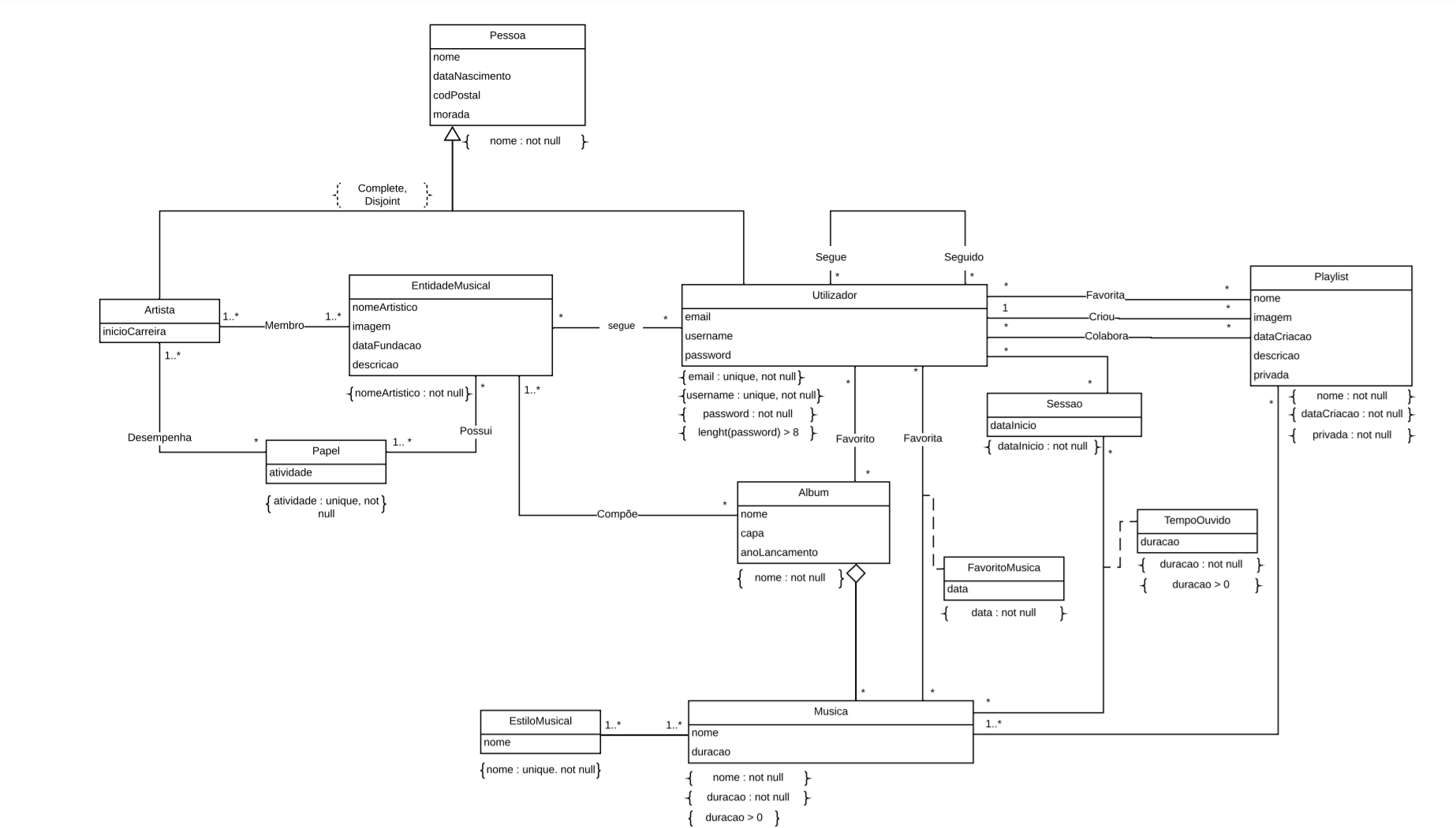
**Gatilhos**

A lista de gatilhos implementados encontra-se ordenada, pelo que ao gatilho 1 corresponde ao código SQL do ficheiro gatilho1\_adiciona.sql, gatilho1\_remove.sql, gatilho1\_verifica.sql.

**Gatilho 1**

**Gatilho 2**

**Gatilho 3**

**Diagrama de Classes**