UNIVERSIDADE D
COIMBRA

Bases de Dados

Grupo 38

DropMusic: Arquivo de Músicas com Partilha de Ficheiros

Autores:

Bruno Miguel Fonseca Ferreira
2014201123

Gustavo Miguel Santos Assunção
2014197707

27 de Novembro de 2018

1 Introdução

Este projeto tem como objetivo criar um sistema de gestão e partilha de músicas com funcionalidades semelhantes a outros grandes e já estabelecidos sistemas, como por exemplo os arquivos *AllMusic* e *IMDb*. Desta forma, a aplicação permite organizar e consultar informações relevantes a músicas, a um utilizador partilhar ficheiros musicais com outros utilizadores, descarregar diferentes ficheiros musicais, entre outras funcionalidades. Durante a pré-conceção da base de dados que servirá de suporte para a este projeto (*DropMusic*), o grupo teve sempre o cuidado de seguir aquilo que acontece na realidade. Ou seja, a base de dados foi projetada de forma a evitar ao máximo dados redundantes e tornar a aplicação o mais genérica possível.

Na segunda etapa deste projeto, a aplicação referida foi implementada sob a forma de um website, acessível em `http://82.155.231.44/main.html#home`, criado de raiz pelo grupo. Foram usadas as linguagens *HTML*, *CSS*, *PHP* e *JavaScript* para o desenvolvimento do *frontend* e *backend* do site, estando este hospedado num servidor pessoal *HTTP Apache/2.4* com *MySQL/8.0.13* também incorporado, dado que *MySQL* for o SGBD escolhido. Por parte do grupo, todos os requerimentos, explicitados no enunciado do projeto, foram implementados juntamente com alguns extras considerados pertinentes ou boas adições a um website apelativo e bom serviço ao *user* comum. Também durante esta segunda fase de desenvolvimento, pequenos ajustes foram feitos à concepção inicial da base de dados tendo, ainda assim, se mantido a mesma estrutura original considerada adequada ao projeto. As noções assumidas foram seguidas adequadamente e mantiveram-se as regras de integridade da base de dados. O site foi optimizado para acessos através de *Google Chrome*, e será mantido online até finais de Dezembro, início de Janeiro, dado que se trata de um servidor pessoal necessário para outros fins.

2 Noções Assumidas

De modo a explicitar os objetivos deste trabalho, são apresentadas seguidamente as noções assumidas na conceção da base de dados criada. Note-se que estas noções foram baseadas na vida real e em exemplos de serviços semelhantes já existentes e bastante estabelecidos. Salienta-se também que estas noções foram mantidas da etapa 1 para a etapa 2.

1. Editoras podem contratar grupos musicais ou músicos individuais.
2. Um editor corresponde a um utilizador com privilégios de edição.
3. Autor e compositor representam a mesma entidade.
4. Músicas podem pertencer a músicos individuais ou a grupos musicais (caso em que pertencem a todos os membros do grupo).
5. Períodos de atividade de grupos musicais podem ter hiatos.
6. Músicas podem ter um ou mais autores/compositores.
7. Músicas podem ser *singles*, isto é, não pertencer a nenhum álbum.
8. Grupos musicais são compostos por uma ou mais pessoas.
9. Uma música pode estar em dois ou mais álbuns diferentes.

10. Um músico pode estar em dois ou mais grupos musicais diferentes.
11. Um concerto[1] implica uma única atuação (músico ou grupo musical).
12. Músicos/grupos musicais não necessitam obrigatoriamente de ser contratados por uma editora.
13. Um álbum é lançado apenas por um único músico individual ou por um único grupo musical.
14. Um género musical está associado a músicas individuais e não a um álbum.
15. Críticas são feitas exclusivamente a álbuns, seguindo o exemplo de grandes e já estabelecidos sistemas de *reviews*, por exemplo o site *AllMusic.com* ou a revista *Rolling Stone*.
16. Certas datas (data de nascimento, data de lançamento, etc) não podem ser superiores à data atual.
17. Os concertos podem ser tanto passados como futuros.

3 Diagramas

Esta secção refere-se ao diagrama conceptual que corresponde à conceção da base de dados para a aplicação *DropMusic*, utilizando a ferramenta *Onda* presente neste *website*. O diagrama entidade-relacionamento (ER) encontra-se na secção A, e o diagrama físico encontra-se na secção B. Note que são apresentados apenas os diagramas atualizados, correspondentes à segunda etapa.

Uma vez que a base de dados foi projetada *à priori* durante a primeira etapa do projeto, pequenas alterações aos diagramas foram realizadas durante a segunda etapa, nomeadamente adição de atributos que foram considerados necessários nas entidades *Playlist*, *Partilha* e *Concerto*. De resto, todas as ligações e entidades anteriormente explicitadas foram mantidas.

4 Descrições de Entidades / Atributos

Seguindo o diagrama conceptual mostrado em anexo, então seguem-se as seguintes descrições de entidades e respectivos atributos criados.

1. Musica - Entidade relativa a cada música existente na base de dados.
   (a) *musica_id*: ID único para cada música.
   (b) *nome*: Nome da música respectiva.
   (c) *genero*: Género da música respectiva.
   (d) *link*: Link para o download da música respectiva.
   (e) *data_lancamento*: Data de lançamento da música respectiva (caso se trate de um single). Caso pertença a álbum é apresentada a data de lançamento do álbum correspondente.
   (f) *letra*: Letra da música respectiva.
2. Musico - Entidade relativa a cada músico existente na base de dados.

[1] Distinguindo-se então entre concertos e festivais, os quais agregam vários concertos.

   (a) *musico_id*: ID único para cada música.
   (b) *nome*: Nome do músico respectivo.
   (c) *ddn*: Data de nascimento do músico respectivo.
   (d) *bio*: Biografia do músico respectivo.

3. Compositor - Entidade relativa a cada compositor/autor existente na base de dados.
   (a) *compositor_id*: ID único para cada compositor/autor.
   (b) *nome*: Nome do respectivo autor/compositor.
   (c) *ddn*: Data de nascimento do respectivo autor/compositor.
   (d) *bio*: Biografia do respectivo autor/compositor.

4. Album - Entidade relativa a cada álbum existente na base de dados.
   (a) *album_id*: ID único para cada álbum.
   (b) *nome*: Nome do álbum respectivo.
   (c) *data_lancamento*: Data de lançamento do álbum respectivo.

5. Editora - Entidade relativa a cada editora existente na base de dados.
   (a) *editora_id*: ID único para cada editora.
   (b) *nome*: Nome da editora respectiva.
   (c) *morada*: Morada física da editora respectiva.

6. Gr_musical - Entidade relativa a cada grupo musical existente na base de dados.
   (a) *grupo_id*: ID único para cada grupo musical.
   (b) *nome*: Nome do grupo musical respectivo.
   (c) *historia*: História do grupo musical respectivo.

7. Concerto - Entidade relativa a cada concerto existente na base de dados.
   (a) *concerto_id*: ID único para cada concerto.
   (b) *duracao_min*: Duração em minutos do concerto respectivo.
   (c) *data*: Data do concerto respectivo (passada ou futura)
   (d) *lotacao*: Lotação total do concerto respectivo.
   (e) *local*: Localização/Morada do concerto respectivo.

8. Periodo - Entidade relativa a cada período de actividade de um grupo musical, existente na base de dados.
   (a) *inicio*: Data de início de um certo período de actividade.
   (b) fim: Data de fim de um certo período de actividade.

9. Critica - Entidade relativa a cada crítica a um álbum, existente na base de dados.
   (a) *titulo*: Título de determinada crítica a um álbum da base de dados.

(b) *justificacao*: Justificação textual para a crítica dada.

(c) *pontuacao*: Pontuação dada na crítica efectuada.

10. Playlist - Entidade relativa a cada playlist existente na base de dados.
    (a) *playlist_id*: ID único para cada playlist.
    (b) *nome*: Nome da respectiva playlist.
    (c) *pub_priv*: Atributo booleano para determinar se a playlist é pública ou privada.
    (d) *data_criacao*: Data de criação e adição à base de dados, da respectiva playlist.
11. Utilizador - Entidade relativa a cada utilizador existente na base de dados.
    (a) *user_id*: ID único para utilizador registado na base de dados.
    (b) *nome*: Nome do respectivo utilizador.
    (c) *is_editor*: Atributo booleano para determinar se o utilizador respectivo tem direitos de edição ou não.
    (d) *sexo*: Sexo do respectivo utilizador.
    (e) *ddn*: Data de nascimento do respectivo utilizador.
12. Partilha - Entidade relativa a cada partilha registada na base de dados.
    (a) *partilha_id*: ID único para cada partilha registada na base de dados.
    (b) *receptor_id*: ID do utilizador receptor da partilha efectuada.
    (c) *item_id*: ID da música ou playlist partilhada.
    (d) *item_type*: Tipo de item (música ou playlist) a ser partilhado.

5 Regras de Integridade

O modelo da base de dados apresentado respeita as diferentes regras de integridade lecionadas nas aulas teóricas, as quais são apresentadas de seguida:

1. Integridade de entidade - dentro de uma tabela não podem existir dois registos com a mesma chave primária.
2. Integridade de domínio - dentro das diferentes tabelas criadas foram impostas algumas, tais como:
    (a) Todos os atributos de todas as tabelas são *NOT NULL*, excepto em casos em que não se justifique que certa entrada tenho um certo campo.
    (b) Na tabela *musica*:
        i. O atributo *género* deverá ser preenchido de acordo pré-estabelecidos géneros musicais: Rock, Pop, Hip Hop, Folk, Jazz, Funk, Opera, R&B, Soul, Classic, Country, Electronic, House, Religious, Kuduro ou Other.
        ii. O atributo *data_lancamento* deverá corresponder a uma data superior a $1000-01-01$, uma vez que é permitido ao utilizador introduzir músicas do género *Classic*, as quais podem datar antes de 1500, por exemplo.

(c) Na tabela *album*, o atributo *data_lancamento* deverá corresponder a uma data superior a $1000-01-01$, uma vez que é permitido utilizador introduzir álbuns que possuem músicas do género *Classic*, por exemplo.

(d) Na tabela *utilizador*, o atributo *ddn* deverá corresponder a uma data superior a $1900-01-01$, uma vez que considera-se que um utilizador não poderá ter mais de 118 anos de idade, no mínimo.

(e) Na tabela *critica*, o atributo *pontuacao* deverá ser preenchido com um valor inteiro entre 1 e 10, uma vez que a maioria dos sistemas de *rating*, utilizam uma escala entre 1 e 10, ou entre 1 e 5.

(f) Na tabela *periodo*:

i. O atributo *inicio* deverá corresponder a uma data superior a $1000-01-01$, uma vez que o grupo musical poderá corresponder a um que produziu músicas do género *Classic*, por exemplo.

ii. O atributo *fim* deverá corresponder a uma data superior a $1000-01-01$.

iii. O atributo *fim* deverá ser uma data igual ou superior à data presente no atributo *inicio*.

(g) Na tabela *concerto*

i. O atributo *data* deverá corresponder a uma data superior a $1000-01-01$.

ii. Os atributos *duracao_min* e *lotacao* deverão ser superiores a 1.

iii. O atributo *local* deverá ser não vazio.

(h) Na tabela *partilha*, os atributos *receptor_id* e *item_id* deverão ser iguais ou superiores a 0. O atributo *item_type* deverá ser não vazio.

(i) Na tabela *musico*, o atributo *ddn* deverá corresponder a uma data superior a $1000-01-01$.

(j) Na tabela *compositor*, o atributo *ddn* deverá corresponder a uma data superior a $1000-01-01$.

(k) Na tabela *playlist*, os atributos *nome* e *data_criacao* deverão ser não nulos. O atributo *pub_priv* deverá ser 1 para público e 0 para privado.

3. Integridade referencial - as seguintes regras vão ser implementadas de acordo com o código SQL mostrado.

(a) Apagar um utilizador implica apagar todas as *playlists* associadas a esse utilizador, todas as partilhas, todas as críticas escritas e todas as suas associações com músicas da base de dados.

(b) Apagar um álbum implica apagar todas as críticas associadas a esse álbum e todas as suas associações com músicas da base de dados.

(c) Apagar um grupo musical implica apagar todos os períodos de atividade associados, todos os concertos e todos os álbuns associados, bem como todas as suas associações com editoras.

(d) Apagar um músico implica apagar todos os concertos associados, bem como todas as suas associações com editoras, grupos musicais e músicas. É também apagada a sua eventual associação com a tabela de compositores.

(e) Apagar um compositor implica apagar todos as suas associações com músicas. É também apagada a sua eventual associação com a tabela de músicos.

(f) Apagar uma editora implica apagar todos as suas associações com músicos e grupos musicais.

(g) Apagar um concerto implica apagar todos as suas associações com músico ou grupo musical.

(h) Apagar uma playlist implica apagar todos as suas associações com utilizadores e músicas.

(i) Apagar uma música implica apagar todos as suas associações com utilizadores, playlists, compositores, músicos, grupos musicais e álbuns.

4. Regras complexas de negócio - estas foram implementadas no desenvolver da *front-end* do sistema, como por exemplo, não permitir caracteres especiais na introdução do nome de um músico ou compositor, o sexo do utilizador ser M (masculino) ou F (feminino), entre outros. Um caso importante a tratar foram as datas, isto é, a base de dados apresentada está preparada para apenas aceitar datas superiores às datas acima consideradas válidas, mas outras restrições tiveram de ser implementadas, como por exemplo, limitar a data de nascimento do utilizador até ao dia corrente.

6 Aplicação WebSite

O WebSite foi desenvolvido de uma forma simplista, seguindo um modelo de cliente-servidor para cumprir o requisito de arquitectura distribuída. Foi também feito de modo a ser responsivo às ações do utilizador, pelo que existe um único ficheiro MAIN.HTML, o qual apresenta ao utilizador a página principal e única de navegação. Clicando em qualquer uma das várias opções apresentadas ao utilizador, a página principal é atualizada para mostrar o conteúdo respectivo (e.g. músicas, álbuns), ou efectuar certa ação (e.g. download do manual de utilizador, criação de playlist, login). Como tal, são chamados os ficheiros .PHP correspondentes que efectuam as acções necessárias, bem como actualizam o que for preciso na página. Considere-se então o seguinte diagrama:

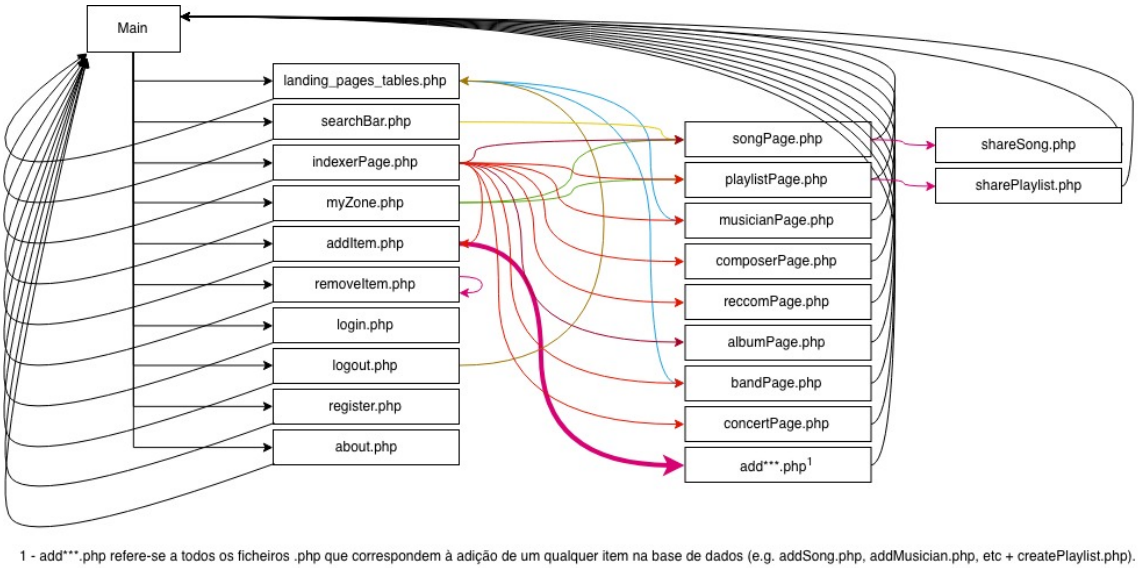


Figura 1: Diagrama de Fluxo entre ficheiros

Como se pode ver, foram cumpridos todos os requisitos expostos no enunciado do projecto, tendo também alguns extras sido adicionados. Tal como referido, o site está disponível em http://82.155.231.44/main.html#home, onde se podem testar todas estas funcionalidades mostradas. O código-fonte, para além de submetido com este relatório, está também indexado em http://82.155.231.44/.

Visto a aplicação tratar-se de um website já online, não necessita de instalação. Basta aceder pelo URL disponibilizado.

7 *Queries* Principais

Durante o desenvolvimento do projeto foi necessário realizar pedidos e inserções de dados através de *queries* organizadas à base de dados do sistema. A tabela 1 representa as *queries* referentes à adição de conteúdo ao *website*, isto é, inserção de dados na base de dados. Por outro lado, para eliminar conteúdo do *website* são realizadas, principalmente, as *queries* na tabela 2. Por último, para visualizar conteúdos, é necessário realizar *queries* do tipo *SELECT*, sendo que as principais encontram-se na tabela 3.

Função	**Comando SQL**
Adicionar Álbum	INSERT INTO album VALUES (default,'albumName','albumDate', NULL,'artist'); OU INSERT INTO album VALUES (default,'albumName','albumDate','artist',NULL); + INSERT INTO album_musica VALUES (album_id,'song');
Adicionar Grupo Musical	INSERT INTO gr_musical VALUES (default,'bandName', 'bandHistory'); INSERT INTO musico_gr_musical VALUES (musico_id, grupo_id); INSERT INTO gr_musical_editora VALUES (grupo_id, editora_id); INSERT INTO periodo VALUES ('date_inicio', 'date_fim', grupo_id);
Adicionar Compositor	INSERT INTO compositor VALUES (default,'nome', 'data_nascimento', 'biografia');
Adicionar Concerto	INSERT INTO concerto VALUES (DEFAULT, duração, 'data', lotacao, 'artist', NULL, 'local'); OU INSERT INTO concerto VALUES (DEFAULT, duração, 'data', lotacao, NULL, 'artist', 'local');
Adicionar Músico	INSERT INTO musico VALUES (default,'nome', 'data_nascimento', 'biografia'); INSERT INTO musico_editora VALUES (musico_id, editora_id); INSERT INTO compositor_musico VALUES (compositor_id, musico_id);
Adicionar Editora	INSERT INTO editora VALUES (default,'nome','morada');

Adicionar Review	INSERT INTO critica VALUES ('titulo','justificacao',pontuacao, user_id,album_id);
Adicionar Música	INSERT INTO musica VALUES (default,'nome','genero','link', 'data_lancamento','letra'); INSERT INTO musico_musica VALUES (musico_id,musica_id); INSERT INTO compositor_musica VALUES (compositor_id, musica_id); INSERT INTO utilizador_musica VALUES (user_id,musica_id);
Criar Playlist	INSERT INTO playlist VALUES (default,'titulo',publica_privada, user_id,'data_criacao'); INSERT INTO playlist_musica VALUES (playlist_id,musica_id);

Tabela 1: Inserção de dados.

Função	**Comando SQL**
Remover música	DELETE FROM utilizador_musica WHERE musica_musica_id=id_pretendido; DELETE FROM utilizador_musica WHERE musica_musica_id=id_pretendido; DELETE FROM playlist_musica WHERE musica_musica_id=id_pretendido; DELETE FROM compositor_musica WHERE musica_musica_id=id_pretendido; DELETE FROM musico_musica WHERE musica_musica_id=id_pretendido; DELETE FROM album_musica WHERE musica_musica_id=id_pretendido; DELETE FROM partilha WHERE item_id=id_pretendido AND item_type='musica'; DELETE FROM musica WHERE musica_id=id_pretendido;
Remover álbum	DELETE FROM critica WHERE album_album_id=id_pretendido; DELETE FROM album_musica WHERE album_album_id=id_pretendido; DELETE FROM album WHERE album_id=id_pretendido;
Remover Playlist	DELETE FROM playlist_musica WHERE playlist_playlist_id=id_pretendido; DELETE FROM partilha WHERE item_id=id_pretendido AND item_type='playlist'; DELETE FROM playlist WHERE playlist_id=id_pretendido;

Remover Grupo Musical	Primeiro remove todos os álbuns do grupo, como mostrado acima DELETE FROM periodo WHERE gr_musical_grupo_id=id_pretendido; DELETE FROM gr_musical_editora WHERE gr_musical_grupo_id=id_pretendido; DELETE FROM concerto WHERE gr_musical_grupo_id=id_pretendido; DELETE FROM musico_gr_musical WHERE gr_musical_grupo_id=id_pretendido; DELETE FROM gr_musical WHERE grupo_id=id_pretendido;
Remover Músico	Primeiro remove todos os álbuns do músico, como mostrado acima DELETE FROM musico_musica WHERE musico_musico_id=id_prentedido; DELETE FROM compositor_musico WHERE musico_musico_id=id_pretendido; DELETE FROM musico_editora WHERE musico_musico_id=id_pretendido; DELETE FROM concerto WHERE musico_musico_id=id_pretendido; DELETE FROM musico_gr_musical WHERE musico_musico_id=id_pretendido; DELETE FROM musico WHERE musico_id=id_pretendido;
Remover Compositor	DELETE FROM compositor_musica WHERE compositor_compositor_id=id_pretendido; DELETE FROM compositor_musico WHERE compositor_compositor_id=id_pretendido; DELETE FROM compositor WHERE compositor_id=id_pretendido;
Remover Concerto	DELETE FROM concerto WHERE concerto_id=id_pretendido;
Remover Editora	DELETE FROM musico_editora WHERE editora_editora_id=id_pretendido; DELETE FROM gr_musical_editora WHERE editora_editora_id=id_pretendido; DELETE FROM editora WHERE editora_id=id_pretendido;

Tabela 2: Remover dados.

Efeito em:	Função	Comando SQL
Página Principal	Para obter, até um máximo de 10, as playlists que tenham sido mais partilhadas até ao momento.	SELECT pa.item_id paid, count(*) soma FROM partilha pa WHERE pa.item_type = 'playlist' GROUP BY pa.item_id ORDER BY pa.item_id DESC LIMIT 10;
	Para obter, até um máximo de 10, os concertos e respectivas informações, que iram ocorrer na semana corrente.	SELECT co.duracao_min coduracao, co.data codata, co.lotacao colotacao, co.local colocal, co.musico_musico_id comuid, co.gr_musical_grupo_id cogrid FROM concerto co WHERE co.data >= CURDATE() AND YEARWEEK(co.data, 1) = YEARWEEK(CURDATE(), 1) ORDER BY co.data ASC LIMIT 10;
	Para obter os nomes dos utilizadores que tenham nascido no dia corrente.	SELECT nome FROM utilizador WHERE MONTH(ddn) = MONTH(CURDATE()) AND DAY(ddn) = DAY(CURDATE()) LIMIT 10;
Indexador	Query principal, onde query_term varia mediante os itens cujos ids e informação se quer obter, e identifier se refere à tabela de onde serão obtidos (e.g. query_term = nome, musica_id & identifier = musica). Variando-se então esses termos mediante a categoria de item musical que se quer indexar e a informação que se quer obter, basta haver uma query no ficheiro que trate disto.	SELECT query_term FROM identifier;

Item específico	Obter informação sobre item específico, mediante categoria, para mostrar na sua página. Note-se que certas vezes "inner joins" são necessários para obter informações que se encontrem em duas ou mais outras tabelas, as quais se relacionam com a tabela principal dessa query por uma tabela extra de associações (e.g. playlist_musica em Partilhas, que guarda informação sobre que músicas são incluídas numa certa playlist):	Álbum: SELECT a.nome anome, a.data_lancamento adata, a.gr_musical_grupo_id agrid, a.musico_musico_id amuid FROM album a WHERE a.album_id=id_pretendido; Grupo Musical: SELECT gr.grupo_id grid, gr.nome grnome, gr.historia hist FROM gr_musical gr WHERE gr.grupo_id=id_pretendido; Compositor: SELECT c.nome cnome, c.ddn cddn, c.bio cbio FROM compositor c WHERE c.compositor_id=id_pretendido Concerto: SELECT co.duracao_min coduracao, co.data codata, co.lotacao colotacao, co.local colocal, co.musico_musico_id comuid, co.gr_musical_grupo_id cogrid FROM concerto co WHERE co.concerto_id=id_pretendido; Músico: SELECT mu.nome munome, mu.ddn muddn, mu.bio mubio FROM musico mu WHERE mu.musico_id=id_pretendido; Playlist: SELECT p.nome pnome, u.nome unome, m.nome mnome, m.musica_id mid FROM playlist p, playlist_musica pm, musica m, utilizador u WHERE u.user_id=p.utilizador_user_id AND p.playlist_id=pm.playlist_playlist_id AND pm.musica_musica_id=m.musica_id AND p.playlist_id=id_pretendido; Editora: SELECT e.nome enome, e.morada emorada FROM editora e WHERE e.editora_id=id_pretendido; Música: SELECT m.nome mnome, m.genero mgenero, m.link mlink, m.data_lancamento mdata, m.letra mletra, mu.nome munome, mu.musico_id muid FROM musica m, musico mu, musico_musica mum WHERE m.musica_id= mum.musica_musica_id AND mum.musico_musico_id=mu.musico_id AND musica_id=id_pretendido;
Partilhas	Inserção de nova partilha de uma música ou uma playlist (tipos de item), feita por um utilizador emissor para um utilizador receptor;	INSERT INTO partilha VALUES (default, id_receptor,id_item_partilhado, id_emissor,'tipo_item_partilhado');

Login	Para verificar se o username do utilizador a tentar fazer login existe ou não na base de dados.	SELECT u.nome unome, u.user_id uid, u.is_editor uie FROM utilizador u WHERE u.nome='nome_submetido';
Registo	Para verificar se o username introduzido já está em utilização por outra pessoa no sistema, ou não.	SELECT nome n FROM utilizador;
	Inserção de novo utilizador na base de dados, caso username seja novo e único.	INSERT INTO utilizador VALUES (default, 'username', 0, 'genero', 'data_nascimento');
My Zone	Mostra as músicas e respectivas informações, que tenham sido partilhadas com o utilizador que esteja a consultar a sua própria My Zone.	SELECT m.musica_id mid, m.nome mnome, m.genero mgenero, m.data_lancamento mdata, m.link mlink, u.nome unome FROM musica m, partilha pa, utilizador u WHERE pa.item_type='musica' AND pa.item_id=m.musica_id AND u.user_id=pa.utilizador_user_id AND pa.receptor_id=id_pretendido;
	Mostra as playlists e respectivas informações, que tenham sido partilhadas com o utilizador que esteja a consultar a sua própria My Zone.	SELECT p.playlist_id pid, p.nome pnome, p.pub_priv ppub, u.nome unome FROM playlist p, partilha pa, utilizador u WHERE pa.item_type='playlist' AND pa.item_id=p.playlist_id AND u.user_id=pa.utilizador_user_id AND pa.receptor_id=id_pretendido;
SearchBar	*	SELECT DISTINCT m.musica_id mid, m.nome mnome FROM musica m, tabela1, tabela2, ... WHERE condicao1 AND condicao2 AND ... AND (search_type LIKE search_term1 OR search_type LIKE search_term2 OR . . .);

Tabela 3: Visualizar os dados.

O *query* (*) acima referente à função *SearchBar* trata das pesquisas feitas por um utilizador. De modo a generalizar a mesma, esta vai buscar as informações das músicas mediante o tipo de pesquisa ($search_type$) usado, podendo este ser por nome, autor, álbum, género, data (ano, ano + mês, ano + mês + dia) e pontuação. Assim sendo, mediante este $search_type$, as tabelas 1 e 2, etc, são selecionadas e *inner joins* são efectuados com a tabela musica (e.g. se se pesquisar

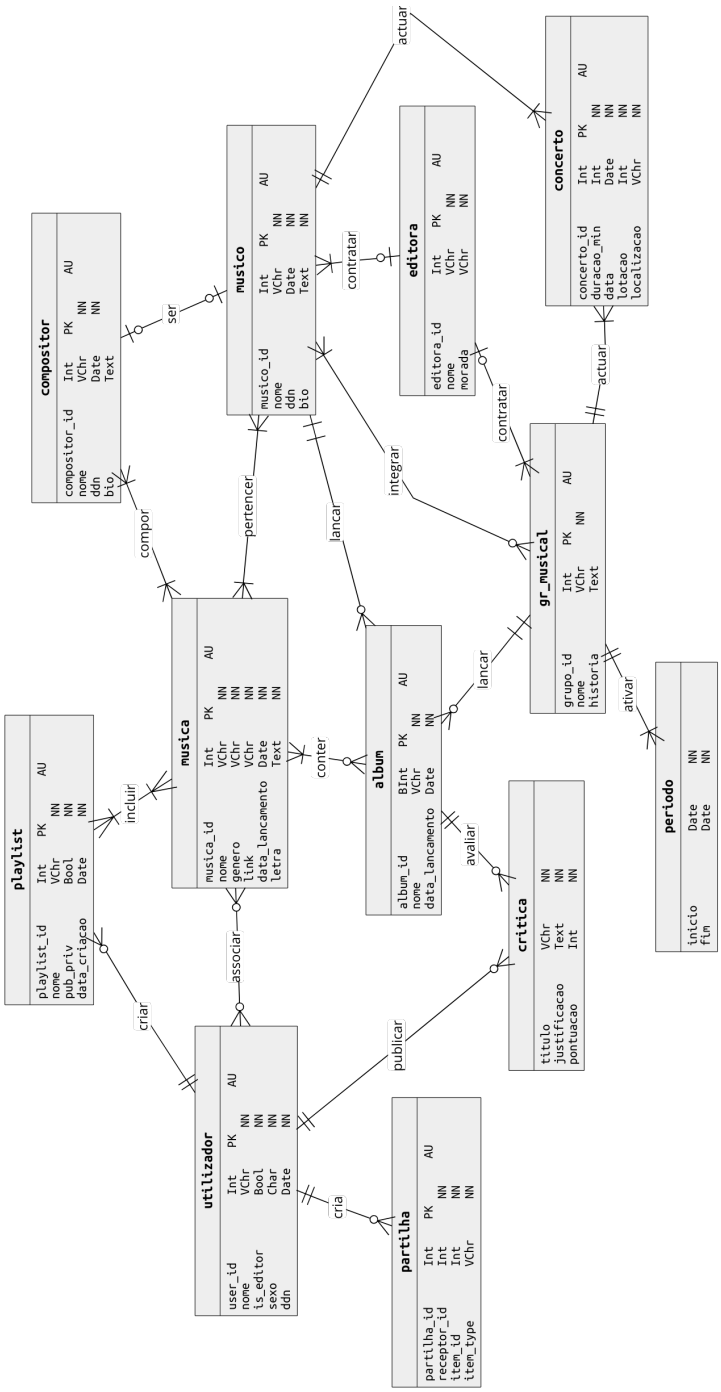
músicas por álbum, fará-se-ão *joins* de *musica* com *album_musica*, e esta tabela com *album*), sendo estes tratados pelas condições 1, 2, etc. De seguida, o *search_term* total especificado pelo utilizador é partido em vários *search_terms* no caso de haver várias palavras divididas por espaços. Estes *search_terms* mais pequenos 1, 2, etc são então usados para fazer comparações nas tabelas respectivas do *MySQl*, usando a palavra *LIKE* e o caracter % para permitir encontrar evidências do que foi pesquisado.

Outras queries não tão importantes ou mais simples, foram também usadas neste projecto. Estas queries não mencionadas aqui tratam-se simplesmente de retornos básicos de informação da base de dados, como obtenção de nomes, datas ou outros campos das tabelas, usando "SELECT". Por vezes estas queries simples incorporam também um "inner join", para conexão de tabelas associadas, caso em que essa conexão é tratada por condições similares a alguns queries expostos acima.

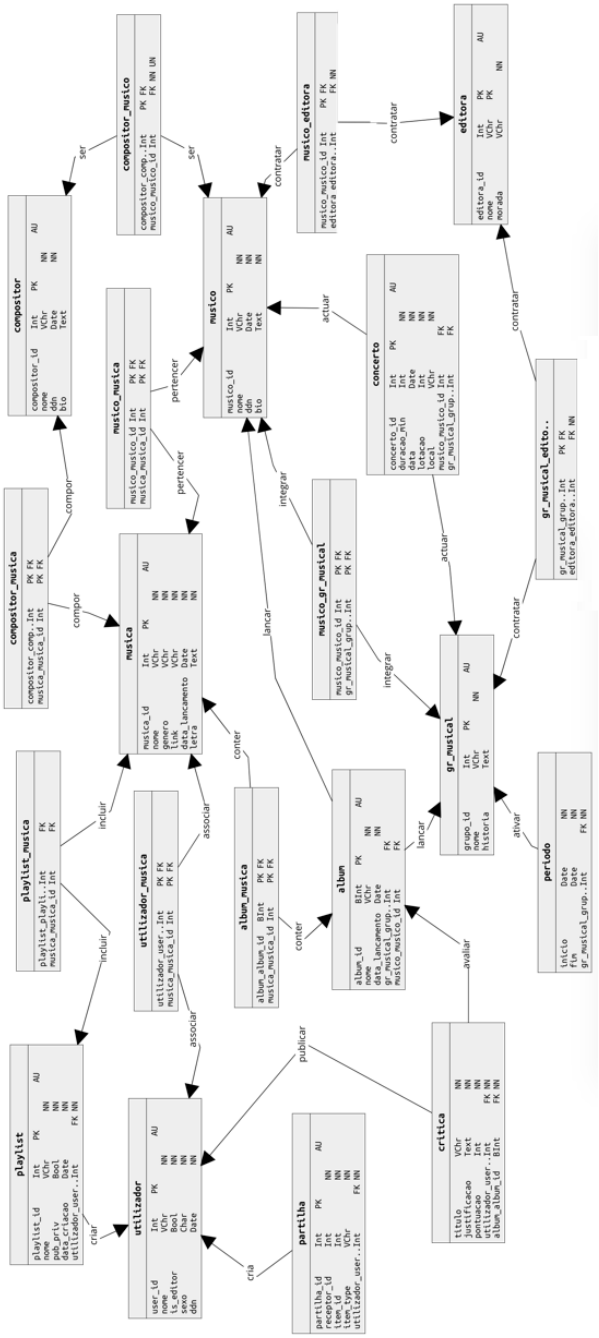
8 Notas

A versão *MySQL* do *script* presente no apêndice C foi aplicada directamente num servidor *MySQL/8.0.13*, não tendo havido erros. Assim sendo, foi a versão usada durante o desenvolvimento do website nesta segunda etapa.

A Diagrama ER



B Diagrama físico



C Código MySQL

```
CREATE TABLE musica (
musica_id  int AUTO_INCREMENT,
nome  varchar(255) NOT NULL,
genero  varchar(255) NOT NULL,
link  varchar(512) NOT NULL,
data_lancamento date NOT NULL,
letra  text NOT NULL,
PRIMARY KEY(musica_id)
);

CREATE TABLE album (
album_id  bigint AUTO_INCREMENT,
nome  varchar(255) NOT NULL,
data_lancamento  date NOT NULL,
gr_musical_grupo_id int,
musico_musico_id  int,
PRIMARY KEY(album_id)
);

CREATE TABLE musico (
musico_id int AUTO_INCREMENT,
nome  varchar(128) NOT NULL,
ddn  date NOT NULL,
bio  text NOT NULL,
PRIMARY KEY(musico_id)
);

CREATE TABLE gr_musical (
grupo_id int AUTO_INCREMENT,
nome  varchar(128) NOT NULL,
historia text,
PRIMARY KEY(grupo_id)
);

CREATE TABLE playlist (
playlist_id  int AUTO_INCREMENT,
nome  varchar(255) NOT NULL,
pub_priv  boolean NOT NULL,
data_criacao  date NOT NULL,
utilizador_user_id int NOT NULL,
PRIMARY KEY(playlist_id)
);

CREATE TABLE critica (
titulo  varchar(255) NOT NULL,
justificacao  text NOT NULL,
```

```
pontuacao  int NOT NULL,
utilizador_user_id int NOT NULL,
album_album_id  bigint NOT NULL
);

CREATE TABLE utilizador (
user_id  int AUTO_INCREMENT,
nome  varchar(128) NOT NULL,
is_editor boolean NOT NULL DEFAULT 0,
sexo  char(1) NOT NULL,
ddn  date NOT NULL,
PRIMARY KEY(user_id)
);

CREATE TABLE editora (
editora_id int AUTO_INCREMENT,
nome  varchar(256),
morada  varchar(512) NOT NULL,
PRIMARY KEY(editora_id,nome)
);

CREATE TABLE concerto (
concerto_id  int AUTO_INCREMENT,
duracao_min  int NOT NULL,
data  date NOT NULL,
lotacao  int NOT NULL,
local  varchar(512) NOT NULL,
musico_musico_id  int,
gr_musical_grupo_id int,
PRIMARY KEY(concerto_id)
);

CREATE TABLE compositor (
compositor_id int AUTO_INCREMENT,
nome  varchar(128) NOT NULL,
ddn  date NOT NULL,
bio  text,
PRIMARY KEY(compositor_id)
);

CREATE TABLE periodo (
inicio  date NOT NULL,
fim  date NOT NULL,
gr_musical_grupo_id int NOT NULL
);

CREATE TABLE partilha (
partilha_id  int AUTO_INCREMENT,
```

```
receptor_id  int NOT NULL,
item_id  int NOT NULL,
item_type  varchar(32) NOT NULL,
utilizador_user_id int NOT NULL,
PRIMARY KEY(partilha_id)
);

CREATE TABLE utilizador_musica (
utilizador_user_id int,
musica_musica_id  int,
PRIMARY KEY(utilizador_user_id,musica_musica_id)
);

CREATE TABLE musico_musica (
musico_musico_id int,
musica_musica_id int,
PRIMARY KEY(musico_musico_id,musica_musica_id)
);

CREATE TABLE musico_editora (
musico_musico_id  int,
editora_editora_id int NOT NULL,
editora_nome  varchar(256) NOT NULL,
PRIMARY KEY(musico_musico_id)
);

CREATE TABLE compositor_musico (
compositor_compositor_id int,
musico_musico_id  int UNIQUE NOT NULL,
PRIMARY KEY(compositor_compositor_id)
);

CREATE TABLE compositor_musica (
compositor_compositor_id int,
musica_musica_id  int,
PRIMARY KEY(compositor_compositor_id,musica_musica_id)
);

CREATE TABLE gr_musical_editora (
gr_musical_grupo_id int,
editora_editora_id  int NOT NULL,
editora_nome  varchar(256) NOT NULL,
PRIMARY KEY(gr_musical_grupo_id)
);

CREATE TABLE playlist_musica (
playlist_playlist_id int,
musica_musica_id  int,
```

```
);

CREATE TABLE musico_gr_musical (
musico_musico_id  int,
gr_musical_grupo_id int,
PRIMARY KEY(musico_musico_id,gr_musical_grupo_id)
);

CREATE TABLE album_musica (
album_album_id  bigint,
musica_musica_id int,
PRIMARY KEY(album_album_id,musica_musica_id)
);

ALTER TABLE musica ADD CONSTRAINT gen_cr CHECK (genero='Rock' or genero='Pop' or
genero='Hip Hop' or genero='Folk' or genero='Jazz' or genero='Funk' or
genero='Opera' or genero='R&B' or genero='Soul' or genero='Classic' or
genero='Country' or genero='Electronic' or genero='House' or genero='Religious' or
genero='Kuduro' or genero='Other');
ALTER TABLE musica ADD CONSTRAINT date_cr CHECK
(data_lancamento >= date '1000-01-01');
ALTER TABLE album ADD CONSTRAINT album_fk1 FOREIGN KEY (gr_musical_grupo_id)
REFERENCES gr_musical(grupo_id);
ALTER TABLE album ADD CONSTRAINT album_fk2 FOREIGN KEY (musico_musico_id)
REFERENCES musico(musico_id);
ALTER TABLE album ADD CONSTRAINT date_cr CHECK
(data_lancamento >= date '1000-01-01');
ALTER TABLE musico ADD CONSTRAINT ddn CHECK (ddn >= date '1000-01-01');
ALTER TABLE playlist ADD CONSTRAINT playlist_fk1 FOREIGN KEY (utilizador_user_id)
REFERENCES utilizador(user_id);
ALTER TABLE critica ADD CONSTRAINT critica_fk1 FOREIGN KEY (utilizador_user_id)
REFERENCES utilizador(user_id);
ALTER TABLE critica ADD CONSTRAINT critica_fk2 FOREIGN KEY (album_album_id)
REFERENCES album(album_id);
ALTER TABLE critica ADD CONSTRAINT score_cr CHECK (pontuacao between 1 and 10);
ALTER TABLE utilizador ADD CONSTRAINT ddn_cr CHECK (ddn >= date '1900-01-01' );
ALTER TABLE concerto ADD CONSTRAINT concerto_fk1 FOREIGN KEY (musico_musico_id)
REFERENCES musico(musico_id);
ALTER TABLE concerto ADD CONSTRAINT concerto_fk2 FOREIGN KEY (gr_musical_grupo_id)
REFERENCES gr_musical(grupo_id);
ALTER TABLE concerto ADD CONSTRAINT data_cr CHECK (data >= date '1000-01-01' );
ALTER TABLE concerto ADD CONSTRAINT dura_lota_cr CHECK
(duracao_min > 1 and lotacao > 1);
ALTER TABLE compositor ADD CONSTRAINT ddn_cr CHECK (ddn >= date '1000-01-01');
ALTER TABLE periodo ADD CONSTRAINT periodo_fk1 FOREIGN KEY (gr_musical_grupo_id)
REFERENCES gr_musical(grupo_id);
ALTER TABLE periodo ADD CONSTRAINT inicio_cr CHECK (inicio >= date '1000-01-01');
ALTER TABLE periodo ADD CONSTRAINT fim_cr CHECK (fim >= date '1000-01-01' );
```

```
ALTER TABLE periodo ADD CONSTRAINT itof_cr CHECK (inicio <= fim);
ALTER TABLE partilha ADD CONSTRAINT partilha_fk1 FOREIGN KEY (utilizador_user_id)
REFERENCES utilizador(user_id);
ALTER TABLE partilha ADD CONSTRAINT rec_item_cr CHECK
(receptor_id >= 0 and item_id >=0);
ALTER TABLE utilizador_musica ADD CONSTRAINT utilizador_musica_fk1
FOREIGN KEY (utilizador_user_id) REFERENCES utilizador(user_id);
ALTER TABLE utilizador_musica ADD CONSTRAINT utilizador_musica_fk2
FOREIGN KEY(musica_musica_id) REFERENCES musica(musica_id);
ALTER TABLE musico_musica ADD CONSTRAINT musico_musica_fk1
FOREIGN KEY (musico_musico_id) REFERENCES musico(musico_id);
ALTER TABLE musico_musica ADD CONSTRAINT musico_musica_fk2
FOREIGN KEY (musica_musica_id) REFERENCES musica(musica_id);
ALTER TABLE musico_editora ADD CONSTRAINT musico_editora_fk1
FOREIGN KEY (musico_musico_id) REFERENCES musico(musico_id);
ALTER TABLE musico_editora ADD CONSTRAINT musico_editora_fk2
FOREIGN KEY (editora_editora_id) REFERENCES editora(editora_id);
ALTER TABLE compositor_musico ADD CONSTRAINT compositor_musico_fk1
FOREIGN KEY (compositor_compositor_id) REFERENCES compositor(compositor_id);
ALTER TABLE compositor_musico ADD CONSTRAINT compositor_musico_fk2
FOREIGN KEY (musico_musico_id) REFERENCES musico(musico_id);
ALTER TABLE compositor_musica ADD CONSTRAINT compositor_musica_fk1
FOREIGN KEY (compositor_compositor_id) REFERENCES compositor(compositor_id);
ALTER TABLE compositor_musica ADD CONSTRAINT compositor_musica_fk2
FOREIGN KEY (musica_musica_id) REFERENCES musica(musica_id);
ALTER TABLE gr_musical_editora ADD CONSTRAINT gr_musical_editora_fk1
FOREIGN KEY (gr_musical_grupo_id) REFERENCES gr_musical(grupo_id);
ALTER TABLE gr_musical_editora ADD CONSTRAINT gr_musical_editora_fk2
FOREIGN KEY (editora_editora_id) REFERENCES editora(editora_id);
ALTER TABLE playlist_musica ADD CONSTRAINT playlist_musica_fk1
FOREIGN KEY (playlist_playlist_id) REFERENCES playlist(playlist_id);
ALTER TABLE playlist_musica ADD CONSTRAINT playlist_musica_fk2
FOREIGN KEY (musica_musica_id) REFERENCES musica(musica_id);
ALTER TABLE musico_gr_musical ADD CONSTRAINT musico_gr_musical_fk1
FOREIGN KEY (musico_musico_id) REFERENCES musico(musico_id);
ALTER TABLE musico_gr_musical ADD CONSTRAINT musico_gr_musical_fk2
FOREIGN KEY (gr_musical_grupo_id) REFERENCES gr_musical(grupo_id);
ALTER TABLE album_musica ADD CONSTRAINT album_musica_fk1 FOREIGN KEY
(album_album_id) REFERENCES album(album_id);
ALTER TABLE album_musica ADD CONSTRAINT album_musica_fk2 FOREIGN KEY
(musica_musica_id) REFERENCES musica(musica_id);
```