# Projeto de Bases de Dados (CC2005) - parte 1

## 1. Elementos do grupo

**Grupo nº** **34**

|  |  |
| --- | --- |
| **Nº mecanográfico** | **Nome** |
| 202107268 | Inês José Cardoso |
| 202108383 | Pedro Rafael Castro Sousa |

## 2. Universo considerado

Neste trabalho escolhemos como universo uma plataforma de streaming de música. Para desenvolvermos este universo, criamos um programa em python que vai buscar diretamente ao Spotify os dados necessários (artistas, músicas, álbuns), como os dados dos utilizadores são confidenciais, criámos alguns “utilizadores exemplo” para demonstrar o funcionamento da nossa base de dados.

## 3. Requisitos

Esta base de dados consiste em seis tabelas: USERS, ARTISTS, ALBUMS, SONGS, PLAYLISTS e FOLLOWERS.

Tabela USERS: armazena informações sobre os utilizadores do serviço de streaming de música e possui os seguintes atributos:

* **user\_id**: é um identificador único de cada utilizador que é gerado e incrementado automaticamente sempre que um novo usuário é adicionado à tabela (PRIMARY KEY);
* **username**: nome do utilizador;
* **e-mail**: endereço de e-mail do usuário, tem que ser único;
* **senha**: palavra-passe escolhida pelo utilizador.

Tabela ARTISTS: armazena as informações sobre os artistas e possui os seguintes atributos:

* **artist\_id**: identificador único de cada artista, que é gerado e incrementado automaticamente sempre que um novo artista é adicionado à tabela (PRIMARY KEY);
* **artist\_name**: o nome do artista.

Tabela ALBUNS: armazena as informações sobre álbuns e possui os seguintes atributos:

* **album\_id**: identificador único de cada álbum, é gerado automaticamente e incrementado sempre que um novo álbum é adicionado à tabela (PRIMARY KEY);
* **Nome**: nome do álbum;
* **Artista**: nome do artista que fez o álbum (FOREIGN,faz referência ao atributo artist\_id na tabela ARTISTS);
* **ReleaseDate**: a data em que o álbum foi lançado.

Tabela SONGS: armazena informações sobre músicas e possui os seguintes atributos:

* **song\_id**: identificador único para cada música, que é gerado e incrementado automaticamente sempre que uma nova música é adicionada à tabela (PRIMARY KEY);
* **Nome**: o nome da música.
* **Artista**: o artista que fez a música (FOREIGN KEY, faz referência ao atributo artist\_id na tabela ARTISTS);
* **Álbum**: o álbum a que a música pertence (FOREIGN KEY estrangeira que faz referência ao atributo album\_id na tabela ALBUMS);
* **Duraçã**o: a duração da música em segundos.

A tabela PLAYLISTS armazena informações sobre as listas de reprodução criadas pelos usuários e possui os seguintes atributos:

* **playlist\_id**: identificador único para cada playlist, que é gerado e incrementado automaticamente sempre que uma nova playlist é adicionada à tabela (PRIMARY KEY);
* **user\_id**: o utilizador que criou a playlist,(FOREIGN KEY, faz referência ao atributo user\_id na tabela USERS);
* **playlist\_name**: nome da lista de reprodução;
* **descriçã**o: uma descrição da lista de reprodução;
* **is\_public**: um valor booleano que indica se a playlist é pública ou não;

A tabela SEGUIDORES armazena informações sobre usuários e possui os seguintes atributos:

* **user\_id**: utilizador que segue o artistas em questão (FOREIGN KEY, faz referência ao atributo user\_id na tabela USERS);
* **artist\_id**: o artistas que está a ser seguido (FOREIGN KEY, faz referência ao atributo artist\_id na tabela ARTISTS).

## 4. Modelo ER

### 4.1. Mapeamento de requisitos em modelo ER

As nossas identidades-tipo vão ser os Usuários/Utilizadores, Artistas, Músicas, Álbuns e Playlists e os nossos relacionamentos serãoo as músicas estão em playlists criadas por utilizadores, que playlists pertencem a cada utilizador, a que artistas os álbuns pertencem, a que álbum as músicas pertencem e a que artista a músicas pertencem.

Logo, dado estas informações, podemos descrever as restrições de cardinalidade entre cada relação:

1. Cada utilizador pode seguir zero ou mais artistas, e cada artista poderá ser seguido por zero ou mais utilizadores, logo será uma relação do tipo N:M.
2. Cada playlist terá exatamente um utilizador, e cada utilizador poderá criar quantas playlists quiser, logo será 1:N.
3. Cada artista tem que ter pelo menos uma música, e cada música tem de ter exatamente um artista, logo 1:N.
4. Cada artista pode ter mais do que um álbum, no entanto, cada álbum só poderá pertencer exatamente a um, logo 1:N.
5. Várias músicas podem estar contidas num álbum, mas só poderão estar contidas a um único álbum (caso estejam), logo será do tipo N:1.
6. Cada playlist pode conter uma quantidade qualquer de músicas (mesmo repetidas), contendo pelo menos uma, mas nem todas as músicas têm de pertencer a playlists, logo M:N.

### 4.2. Diagrama ER

#### 4.2.1. Código dbdia

Users (user\_id,username,email,password)

Artists (artist\_id,artist\_name)

Albums (album\_id,artist\_id,album\_name,release\_date)

Songs (song\_id,album\_id,artist\_id,song\_name,duration)

Playlists (playlist\_id,user\_id,playlist\_name,description,is\_public)

Users --- N --- <FOLLOWS> === M === Artists

Users --- 1 --- <HAS\_PLAYLIST> === N === Playlists

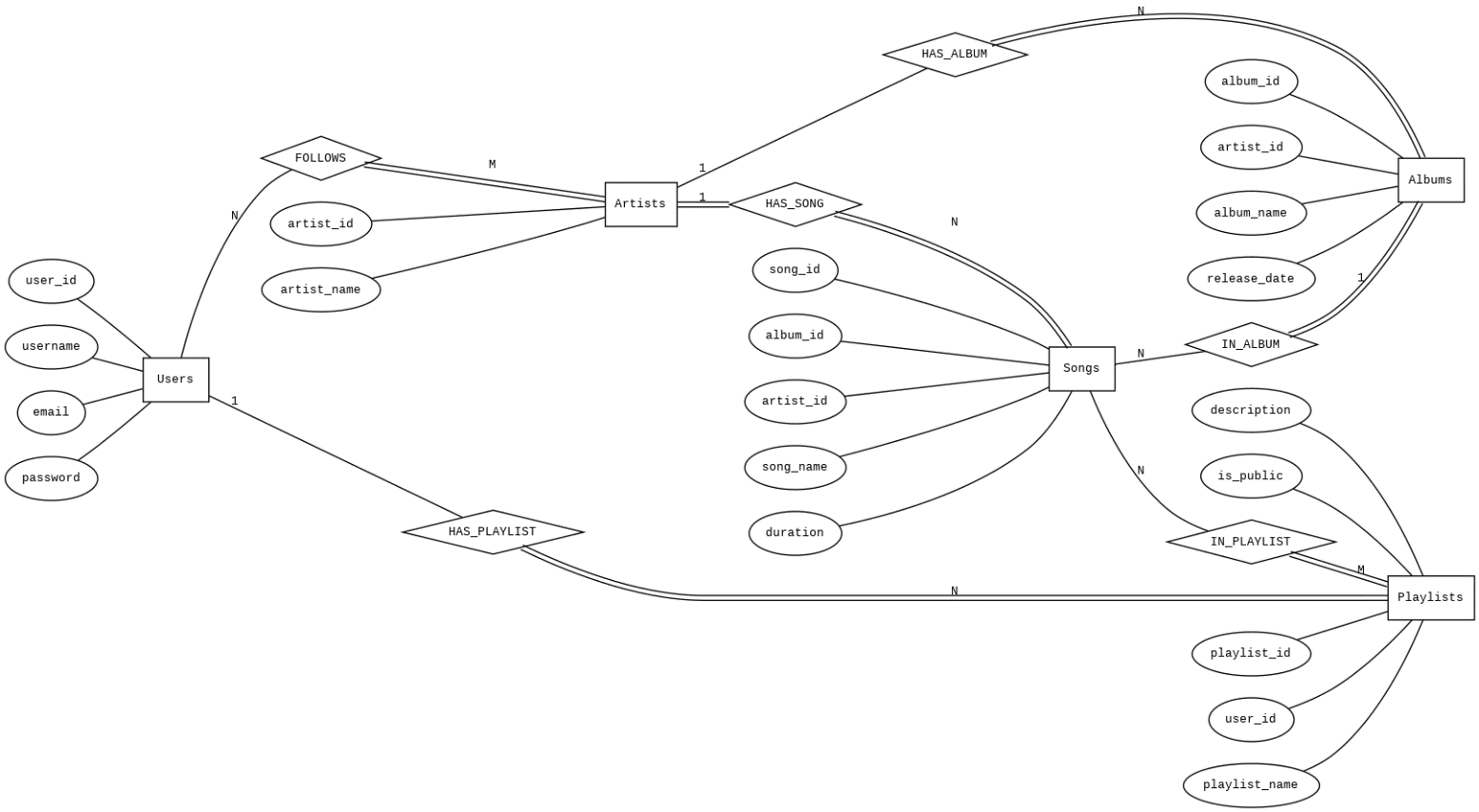
Artists === 1 === <HAS\_SONG> === N === Songs

Songs --- N --- <IN\_ALBUM> === 1 === Albums

Songs --- N --- <IN\_PLAYLIST> === M === Playlists

Artists --- 1 --- <HAS\_ALBUM> === N === Albums

#### 4.2.2. Imagem do diagrama



## 5. Modelo relacional

### 5.1. Mapeamento do modelo ER para relacional

Nós, para cada relação do tipo M:N, criamos uma tabela que relaciona as chaves primárias de cada entidade participante na relação, para as restantes relações de cardinalidade 1:N e N:1, tornamos como atributo o elemento único entre entidades. De resto, criamos uma tabela para cada entidade, também criando um ID exclusivo para cada elemento.

### 5.2. Diagrama para modelo relacional

#### 5.2.1. Código dbdia

table Users(\_user\_id\_,username,email,password,)

table Artists(\_artist\_id\_,artist\_name)

table Albums(\_album\_id\_,Name,Artist --> Artists.artist\_id,ReleaseDate)

table Songs(\_song\_id\_,Name,Artist --> Artists.artist\_id,Album --> Albums.album\_id,Duration)

table Playlists(\_playlist\_id\_,user\_id --> Users.user\_id,playlist\_name,description,is\_public)

table Playlist\_songs(playlist\_id --> Playlists.playlist\_id,song\_id --> Songs.song\_id,position)

table Followers(user\_id --> Users.user\_id,artist\_id --> Artists.artist\_id)

#### 5.2.2. Imagem do diagrama

