# Projeto de Bases de Dados (CC2005) - parte 2

## 1. Elementos do grupo

**Grupo nº** **[34]**

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

## 2. Correção do modelo da BD

* O nosso dataset foi alterado de forma a ir buscar diretamente músicas, artistas e álbuns ao Spotify (como os dados dos utilizadores são confidenciais, foram criados alguns utilizadores-exemplo para demonstrar o funcionamento da base de dados);
* Descrevemos concretamente as características individuais dos atributos dos objetos do universo;
* Adicionamos a justificação das cardinalidades, da participação para os relacionamentos e os requisitos para a entidade tipo;
* Por fim, explicamos a tradução do modelo ER e resolvemos uns problemas

do modelo relacional.

### 2.1. Correção ao universo considerado e fontes de dados

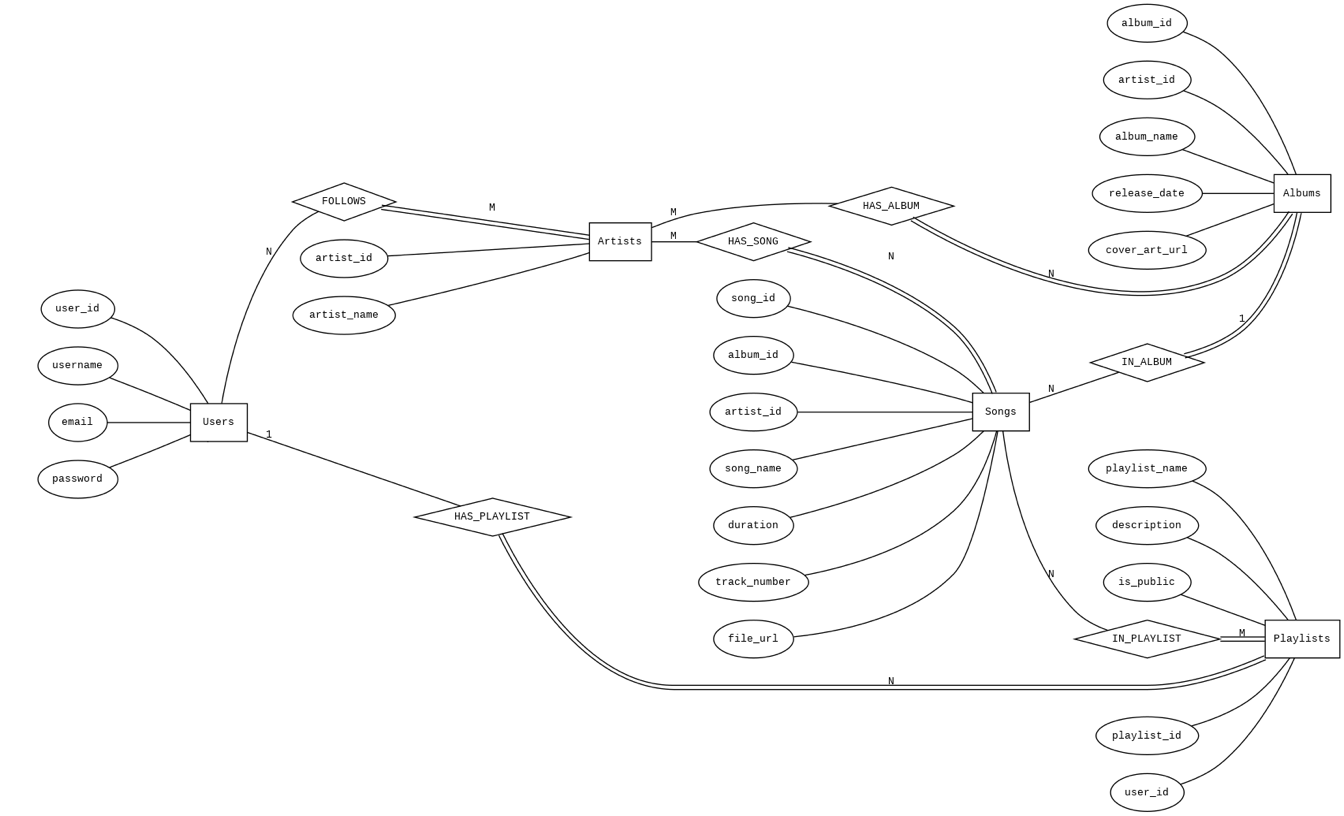
* Como referido anteriormente, alteramos o nosso dataset de modo a ser mais eficiente a busca de músicas artistas e playlists ao Spotify.

### 2.2. Correção de requisitos

* Adicionamos as características das entidades e atributos e retiramos a possibilidade de que um artista seja também um utilizador.

### 2.3. Correção do modelo ER

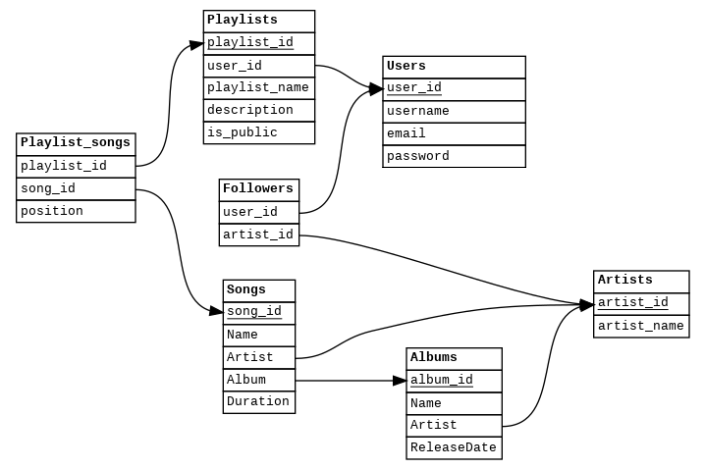
* Melhoramos o nosso modelo entidade-relacionamento de modo a que a sua utilização seja mais prática e adicionamos as suas restrições de cardinalidade.



### 

### ´2.4. Correção do modelo relacional

* Traduzimos do modelo ER para o modelo relacional de maneira apropriada.



## 3.

## Povoamento de tabelas

1. Como fonte das primeiras 4 tabelas, utilizamos diretamente o Spotify (empregando um programa em python que acede ao Spotify utilizando a API oficial). Os dados de utilizadores e as restantes informações diretamente relacionadas com estes (que artistas segue, playlists criadas e músicas nas playlists), como são confidenciais, foram criadas à mão para exemplos demonstrativos.
2. A base de dados foi povoada usando o mesmo programa de Python referido anteriormente e a biblioteca pymysql, realizando vários INSERTS dos dados, depois destes serem extraídos ou criados.

|  |  |
| --- | --- |
| **Nome da tabela** | **Nº de entradas** |
| ARTISTS | 15 |
| SONGS | 773 |
| ALBUMS | 177 |
| USERS | 15 |
| PLAYLISTS | 14 |
| PLAYLIST\_SONGS | 100 |
| FOLLOWERS | 25 |

## 4. Aplicação Python

|  |  |
| --- | --- |
| **“Endpoint”** | **Funcionalidade** |
| **/** | Página de entrada |
| **/songs/** | Listagem de todas as músicas na base de dados. |
| /**songs/<int:id>/** | Apresentação dos dados de uma música dado o seu id. |
| **/albums-list/** | Listagem de todas os álbuns na base de dados. |
| **/albums/<int:id>/** | Apresentação dos dados de álbum dado o seu id. |
| **/artists-list/** | Listagem de todos os artistas na base de dados. |
| **/artist/<int:id>/** | Apresentação dos dados, álbuns e singles de um artista, dado o seu id. |
| **/users-list/** | Listagem de todos os utilizadores na base de dados. |
| **/user/<int:id>/** | Apresentação dos dados, artistas seguidos e playlists de um dado utilizador, dado o seu id. |
| **/playlists-list/** | Listagem de todos as playlists de utilizadores na base de dados. |
| **/playlist/<int:id>/** | Apresentação dos dados e músicas numa playlist, dado o seu id. |

4.2

**/ :**

SELECT COUNT(\*) AS artist1 FROM ARTISTS

SELECT COUNT(\*) AS song1 FROM SONGS

SELECT COUNT(\*) AS album1 FROM ALBUMS

SELECT COUNT(\*) AS user1 FROM USERS

SELECT COUNT(\*) AS playlist1 FROM PLAYLISTS

No index, como listamos a quantidade de entradas de cada tabela, temos uma query para contar o número das respetivas.

**/songs/ :**

SELECT song\_id, Name, Artist, Album, Duration

      FROM SONGS

      ORDER BY song\_id

Nas páginas de listagem das tabelas, nós fazemos uma query com todos os dados importantes para a listagem dos items. Por conveniência, listamos todos os items por ordem de id.

Consoante isto, podemos ver as seguintes:

**/albums-list/ :**

SELECT album\_id, Name, Artist

      FROM ALBUMS

      ORDER BY album\_id

**/artists-list/ :**

SELECT artist\_id, artist\_name

      FROM ARTISTS

      ORDER BY artist\_id

**/users-list/ :**

SELECT user\_id, username, email, password

      FROM USERS

      ORDER BY user\_id

**/playlists-list/ :**

SELECT playlist\_id, playlist\_name, user\_id, description, is\_public

      FROM PLAYLISTS

      ORDER BY playlist\_id

**/songs/<int:id>/ :**

Na página de cada música, nós listamos os dados os dados da música, o seu artista (por nome), e o álbum a que pertence (caso haja), logo, temos 3 queries:

SELECT song\_id, Name, Artist, Duration , Album

      FROM SONGS

      WHERE song\_id = %s

Nesta primeira query, o %s representa o id passado pelo endpoint, os dados da query são guardados na variável “song”.

SELECT artist\_id, artist\_name

      FROM ARTISTS

      WHERE artist\_id = %s

SELECT album\_id, Name, Artist, ReleaseDate

        FROM ALBUMS

        WHERE album\_id = %s

Nestas queries, o %s representa, respetivamente, o valor na variável song para o id do artista (song[“Artist”]) e do álbum (song[“Album”]).

Para os restantes endpoints em que são descritos os dados de um item de uma entidade em específico, são feitas queries semelhantes, como as seguintes:

/albums/<int:id>/ :

SELECT album\_id, Name, Artist, ReleaseDate

        FROM ALBUMS

        WHERE album\_id = %s

SELECT song\_id, Name, Artist, Duration , Album

        FROM SONGS

        WHERE Album = %s

SELECT artist\_id, artist\_name

      FROM ARTISTS

      WHERE artist\_id = %s

Na visualização de um álbum em específico, além das informações deste, fazemos uma query com o id do álbum para visualizarmos as músicas todas pertencentes a este, e mais uma query para irmos buscar o nome do artista consoante o seu id.

/artist/<int:id>/ :

SELECT artist\_id, artist\_name

      FROM ARTISTS

      WHERE artist\_id = %s

SELECT album\_id, Name, Artist

      FROM ALBUMS

      WHERE Artist = %s

SELECT song\_id, Name, Duration, Album

      FROM SONGS

      WHERE Artist = %s AND Album IS NULL

SELECT COUNT(\*) AS numfollowers FROM FOLLOWERS where artist\_id = %s

Ao inspecionarmos um artista, além de fazermos queries aos seus atributos, fazemos duas queries: uma de modo a obtermos todos os álbuns desse, e a seguinte para visualizarmos as músicas que não estão em álbuns, os “singles”. Por fim, para estatística, temos uma query para retornar o número de seguidores que cada artista tem.

/playlist/<int:id>/ :

SELECT playlist\_id, playlist\_name, user\_id, description, is\_public

      FROM PLAYLISTS

      WHERE playlist\_id = %s

SELECT playlist\_id, song\_id, position, Name, Duration

        FROM PLAYLIST\_SONGS NATURAL JOIN SONGS

        WHERE playlist\_id = %s

SELECT user\_id, username

      FROM USERS

      WHERE user\_id = %s

Ao listarmos uma playlist de um user, além de listarmos todas as músicas pertencentes à sua playlist (fazendo uma query aonde é dado um natural join às músicas com o respetivo playlist\_id), listamos o nome do utilizador que a criou e o nome da playlist.

/user/<int:id>/ :

SELECT playlist\_id, playlist\_name, user\_id, description, is\_public

      FROM PLAYLISTS

      WHERE user\_id = %s

SELECT user\_id, artist\_id, artist\_name

      FROM FOLLOWERS NATURAL JOIN ARTISTS

      WHERE user\_id = %s

SELECT user\_id, username, email, password

      FROM USERS

      WHERE user\_id = %s

Na visualização de um utilizador, listamos os seus dados pessoais, os seus artistas seguidos, e as suas playlists criadas.