Faculdade de Ciências da Universidade do Porto

Base de Dados (CC2005) 2021/22

Relatório do 2º Projeto de Base de Dados

Hugo Silva up202004447
João Malheiro up202004449
Tomás Ferreira up202006594

1. Introdução

Este relatório foi realizado no âmbito da continuidade do trabalho prático da unidade curricular de Base de Dados. O novo foco agora seria de conceber uma aplicação web simples que faça interface com a base de dados (BD) que já havíamos desenvolvido: uma que simula um sistema de streaming como o Spotify.

Esta aplicação deveria ser escrita em Python 3 e usar as bibliotecas Flask e PyMySQL, de acordo com o ambiente de código previamente fornecido pelos docentes. Tivemos também o acesso a uma BD de exemplo, a 'MovieStream', tendo esta sido consolidada nas aulas teóricas.

2. Pequena Introdução à nossa BD

Como já referido antes, a nossa base de dados tem como intuito simular o comportamento da plataforma de streaming Spotify, de forma mais simplificada do que o site original. Tendo a fonte SQL feita na parte 1 do nosso projeto, bastou apenas fazer pequenas alterações ao código SQL em si (de forma a tornar a melhor experiência de navegação pelo site) e por fim criar um site adaptado a tal código através das templates e de material já dado.

Tables	Total
Users	10
Artists	10
Songs	70
Genres	5

3. Pequenas adaptações da conversão para a web

Para facilitar a navegação e tornar a experiência mais agradável pelo site as seguintes alterações foram feitas:

1. Artista (ARTIST) agora é uma tabela totalmente independente, ou seja, artistas já não são considerados users (USERS), eliminando assim a necessidade da tabela ARTIST ser definida por dois Id's. A tabela ARTIST é constituída por:
    - Login (ALogin) - **Chave primária da tabela**;
    - Name;
    - Verified.
2. Mudança de alguns nomes na BD para facilitar a busca de dados:
    - SONG.Id —> SId;
    - USER.Login —> ULogin;
    - ARTIST.Login —> ALogin;
    - PLAYLIST.Id —> PId;
    - ALBUM.Id —> AId.

    Por consequência, as tabelas secundárias ALBUM_SONG, PLAYLIST_SONG, ALBUM_SONG, SONG_LIKED, PLAYLIST_LIKED, ALBUM_LIKED e SONG_GENRE passam a ter a PRIMARY KEY com o mesmo nome das FOREIGN KEYS para onde aponta, podendo dar mais uso ao NATURAL JOIN para pesquisa de dados, tornando o processo mais fácil.

3. Acréscimo da tabela ARTIST_FOLLOWER que tem como PRIMARY KEY (ALogin, ULogin) e permite saber quais USERS seguem cada ARTIST.

4. Inserção de novos dados na Base de Dados.

4. Funcionalidades

A nossa app web foca-se em 4 entidades, que podemos encontrar na página de entrada:

- USER
- ARTIST
- SONG
- GENRE

A partir destas entidades podemos aceder a tabelas de outras, por exemplo, ao aceder à info de uma música podemos ter acesso ao ALBUM associado à música (se esta estiver associada a um álbum), PLAYLIST associada à música (se esta estiver associada a uma playlist) e GENRE associado à música (se esta estiver associada a um género).

A página inicial está estruturada em 4 vertentes de funcionalidades:

1. Identificação do grupo responsável pela criação da app web;

2. Créditos para a app em que nos inspiramos: Spotify (com hiperligação para a webpage do sistema de streaming);

3. Todo o volume de dados que possuímos;

```python
# Start page
@APP.route('/')
def index():
    stats = {}
    x = db.execute('SELECT COUNT(*) AS users FROM USER').fetchone()
    stats.update(x)
    x = db.execute('SELECT COUNT(*) AS songs FROM SONG').fetchone()
    stats.update(x)
    x = db.execute('SELECT COUNT(*) AS artists FROM ARTIST').fetchone()
    stats.update(x)
    x = db.execute('SELECT COUNT(*) AS genres FROM GENRE').fetchone()
    stats.update(x)
    logging.info(stats)
    return render_template('index.html',stats=stats)
```

Tables	Total
Users	10
Artists	10
Songs	70
Genres	5

4. Um centro de pesquisa para cada uma das entidades principais, com 3 opções de pesquisa:
   4.1. Aceder a todos os dados da entidade em causa, definindo um endpoint "/<entidade>" para a aplicação Flask e disponibilizando uma tabela com todos os atributos pertencentes. Para cada entidade, existe por sua vez uma ligação para a sua página estrita, mostrando todas as relações que lhe dizem respeito ("/<entidade>/<Id de entidade>"), ao que se chama uma instrução parametrizada de parâmetro %s (o id da entidade escolhida);

Artists

Name	Verified
Glacier	Yes
Hugo Silva	No
Joao Malheiro	No
Kendrick Lamar	Yes
Kupla	Yes
Linkin Park	Yes
Quim Barreiros	Yes
Rodrigo Devesa	No
Rosinha	Yes
Virtual Riot	Yes

Ordered by Artist Name

```python
@APP.route('/artists/')
def list_artists():
    artists = db.execute(
        '''
        SELECT A.ALogin, A.Name, A.Verified
        FROM ARTIST A
        ORDER BY A.Name
        '''
    ).fetchall()

    return render_template('artists-list.html', artists=artists)
```

Virtual Riot

Artist info:

Login	Name	Verified
5	Virtual Riot	Yes

Albums created

- Still Kids

Songs Created

- Everyday
- Kingdoms & Castles
- Lost It
- Still Kids

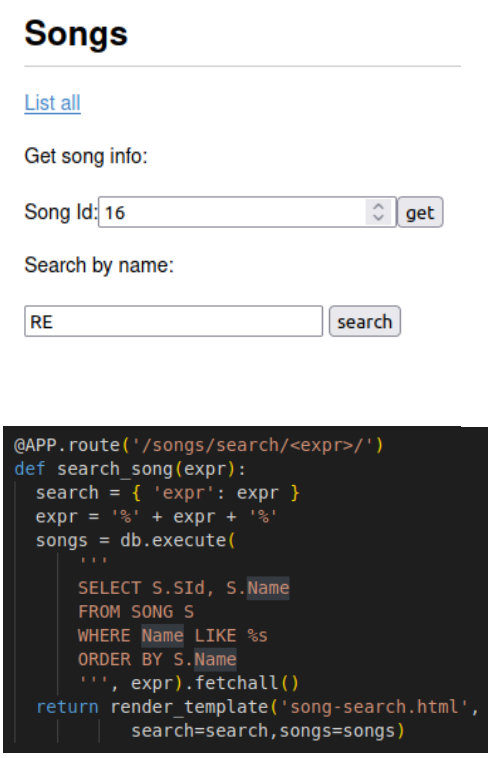
Followed by

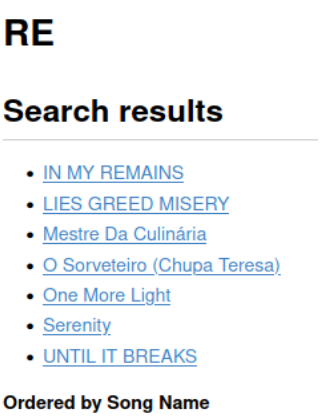
- Diogo Ribeiro

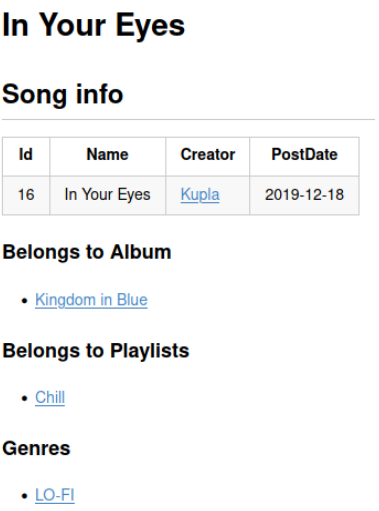
```python
@APP.route('/artists/<int:id>/')
def get_artist(id):
    artist = db.execute(
        '''
        SELECT A.ALogin, A.Name, A.Verified
        FROM ARTIST A
        WHERE A.ALogin = %s
        ''', id).fetchone()

    if artist is None:
        abort(404, 'Artist Id {} not found.'.format(id));

    songs = db.execute(
        '''
        SELECT S.SId, S.Name
        FROM SONG S JOIN ARTIST A ON(S.Creator = A.ALogin)
        WHERE A.ALogin = %s
        ORDER BY S.Name
        ''', id).fetchall()

    albums = db.execute(
        '''
        SELECT A.AId, A.Name
        FROM ALBUM A JOIN ARTIST ON(A.Creator = ALogin)
        WHERE ALogin = %s
        ORDER BY A.Name
        ''', id).fetchall()

    followers = db.execute(
        '''
        SELECT F.ULogin, F.Name
        FROM USER F
        JOIN ARTIST_FOLLOWER ON(Follower = F.ULogin)
        JOIN ARTIST A ON(Artist = A.ALogin)
        WHERE A.ALogin = %s
        ORDER BY F.Name
        ''', id).fetchall()

    return render_template('artist.html',
    artist=artist, songs=songs, albums=albums, followers=followers)
```

4.2. Aceder aos dados de uma entidade com base numa pesquisa por um dado id (“/<entidade>/<Id_da_entidade>") ou por uma expressão ‘expr’ (“/<entidade>/search/<expr>/”). Esta forma de pesquisa, no entanto, não procede de uma instrução parametrizada: ao invés, o código SQL requerido é gerado dinamicamente em função do valor fornecido (expr).

Assim como no caso anterior, haverá também a disponibilização de hiperligações para entidades que correspondam com o que queiramos pesquisar: quer seja um id específico ou mesmo uma expressão. Em caso de o valor fornecido não existir dentro da base de dados, ou aparecerá uma mensagem de erro revelando que o id da entidade não existe ou a pesquisa não concluirá nenhum resultado;








```
@APP.route('/songs/search/<expr>/')
def search_song(expr):
  search = { 'expr': expr }
  expr = '%' + expr + '%'
  songs = db.execute(
      '''
      SELECT S.SId, S.Name
      FROM SONG S
      WHERE Name LIKE %s
      ORDER BY S.Name
      ''', expr).fetchall()
  return render_template('song-search.html',
          search=search,songs=songs)
```

Outros serviços proporcionados pela web app incluem mecanismos como:

- a consulta de um álbum ou playlist seguida por um determinado user;

MyPlaylist

Playlist info:

Id	Name	Creator	CreationDate
4	MyPlaylist	Tomás Ferreira	2020-11-23

Tracklist

- LOST IN THE ECHO
- BURN IT DOWN
- CASTLE OF GLASS
- Nobody Can Save Me
- Talking to Myself
- Battle Symphony

Liked by Users

- Manuela Santos
- Tomás Ferreira

```
playlists = db.execute(
    '''
    SELECT PId, P.Name
    FROM PLAYLIST P JOIN USER ON(Creator = ULogin)
    WHERE ULogin = %s
    ORDER BY P.Name
    ''', id).fetchall()
```

- a verificação do número de users que seguem um certo artista, os álbuns criados e outras infos globais ao universo da BD;

Kendrick Lamar

Artist info:

Login	Name	Verified
8	Kendrick Lamar	Yes

Albums created

- DAMN

Songs Created

- BLOOD
- DNA
- DUCKWORTH
- ELEMENT
- FEAR
- FEEL
- GOD
- HUMBLE
- LOVE
- LOYALTY
- LUST
- PRIDE
- XXX
- YAH

Followed by

- Tomás Ferreira

```
@APP.route('/artists/<int:id>/')
def get_artist(id):
    artist = db.execute(
        '''
        SELECT A.ALogin, A.Name, A.Verified
        FROM ARTIST A
        WHERE A.ALogin = %s
        ''', id).fetchone()

    if artist is None:
        abort(404, 'Artist Id {} not found.'.format(id));

    songs = db.execute(
        '''
        SELECT S.SId, S.Name
        FROM SONG S JOIN ARTIST A ON(S.Creator = A.ALogin)
        WHERE A.ALogin = %s
        ORDER BY S.Name
        ''', id).fetchall()

    albums = db.execute(
        '''
        SELECT A.AId, A.Name
        FROM ALBUM A JOIN ARTIST ON(A.Creator = ALogin)
        WHERE ALogin = %s
        ORDER BY A.Name
        ''', id).fetchall()

    followers = db.execute(
        '''
        SELECT F.ULogin, F.Name
        FROM USER F
        JOIN ARTIST_FOLLOWER ON(Follower = F.ULogin)
        JOIN ARTIST A ON(Artist = A.ALogin)
        WHERE A.ALogin = %s
        ORDER BY F.Name
        ''', id).fetchall()

    return render_template('artist.html',
    artist=artist, songs=songs, albums=albums, followers=followers)
```

- (...)

5. Considerações finais

Assim termina o nosso trajeto pelo desenvolvimento de uma base de dados funcional. Ficaram ainda algumas funcionalidades por falar, para as quais aconselhamos vivamente de desvendar e usufruir desta 'Spotify-type Database'. Como grupo, não poderíamos estar mais satisfeitos com o trabalho efetuado: foi uma tarefa desafiante à qual traduzimos num protótipo funcional, compacto e eficaz.