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

## 1. Elementos do grupo

**Grupo nº [51]**

| **Nº mecanográfico** | **Nome** |
|---|---|
| 202204576 | Gabriel Filipe Lucas Ribeiro Ramos |
| 202109728 | Rui Filipe da Silva Santos |
| | |

## 4. Aplicação Python

1) **Indique os "endpoints" implementados e um sumário da respectiva funcionalidade na tabela abaixo.**

| **"Endpoint"** | **Funcionalidade** |
|---|---|
| ***/*** | Página de entrada |
| */mangas/* | Página com todos os mangas e respetivas informações |
| */mangas/*<int:id>/ | Página com informacao especifica de um manga |
| */mangas/*search/<expr>/ | Página com todos os mangas que assemelham a uma dada expressao |
| */animes/* | Página com todos os animes e respetivas informações |
| */animes/*<int:id>/ | Página com informacao especifica de um anime |
| */animes/*search/<expr>/ | Página com todos os animes que assemelham a uma dada expressao |
| */genres/* | Página com todos os gênero |
| */genres/*<int:id>/ | Página com todos os mangas e animes de um genero |

| | |
|---|---|
| */authors/* | Página com todos os autores de manga e respetivas informações |
| */authors/*<int:id>/ | Página com informacao especifica de um autor de manga |
| */authors*/search/<expr>/ | Página com todos os autores de manga que assemelham a uma dada expressao |
| */studios/* | Página com todos os estudios de anime e respetivas informações |
| */studios/*<int:id>/ | Página com informacao especifica de um estudio de anime |
| */studios*/search/<expr>/ | Página com todos os estudios de anime que assemelham a uma dada expressao |

**2) Indique as consultas mencionadas no ponto 3 para a aplication Python, expondo o código SQL e explicando o seu significado.**

```
@APP.route('/')
def index():
    stats = {}
    x = db.execute('SELECT COUNT(*) AS manga FROM MANGA').fetchone()
    stats.update(x)
    x = db.execute('SELECT COUNT(*) AS anime FROM ANIME').fetchone()
    stats.update(x)
    logging.info(stats)
    return render_template('index.html',stats=stats)
```

*Figura 1 - screenshot de app.py (/)*

1. ***SELECT COUNT(*) AS manga FROM MANGA***

Conta o número de mangas presentes na base de dados (renomeando a contagem com o nome 'manga').

2. ***SELECT COUNT(*) AS anime FROM ANIME***

Conta o número de animes presentes na base de dados (renomeando a contagem com o nome 'anime').

```python
#Manga
@APP.route('/mangas/')
def list_mangas():
    mangas = db.execute(
      '''
      SELECT MangaId,manga_name,chapters,volumes,mangaRating,start_date,end_date,pre_quel
      FROM MANGA
      ORDER BY manga_name
      ''').fetchall()
    return render_template('manga-list.html', mangas=mangas)
```

*Figura 2 - screenshot de app.py (/mangas/)*

1. ***SELECT MangaId,manga_name,chapters,volumes,mangaRating,start_date,end_date ,pre_quel***

   ***FROM MANGA***

   ***ORDER BY manga_name***

Seleciona todos os registos e todos os atributos da tabela 'MANGA' e ordena-os por 'manga_name' , ou seja, por nome da manga.

```python
@APP.route('/mangas/<int:id>/')
def get_manga(id):
    manga = db.execute(
      '''
      SELECT MangaId,manga_name,chapters,volumes,mangaRating,start_date,end_date,pre_quel
      FROM MANGA
      WHERE MangaId = %s
      ''', id).fetchone()
    if manga is None:
        abort(404, 'Manga id {} does not exist.'.format(id))

    genres = db.execute(
      '''
      SELECT GenreId, Label
      FROM MANGA_GENRE NATURAL JOIN GENRE NATURAL JOIN  MANGA
      WHERE MangaId = %s
      ORDER BY Label
      ''', id).fetchall()
    return render_template('manga.html',manga = manga,genres = genres)
```

*Figura 3 - screenshot de app.py (/mangas/<int:id>/)*

1. ***SELECT MangaId,manga_name,chapters,volumes,mangaRating,start_date,end_date, pre_quel***

   ***FROM MANGA***

   ***WHERE MangaId = %s***

Seleciona todos os registos e todos os atributos da tabela 'MANGA' que tem um dado 'MangaId'.

2. ***SELECT***

   ***GenreId, Label***

   ***FROM MANGA_GENRE NATURAL JOIN GENRE NATURAL JOIN MANGA***

   ***WHERE MangaId = %s***

   ***ORDER BY Label***

Fazemos a junção natural entre as tabelas 'MANGA_GENRE' e 'Genre' (através do atributo 'GenreId') e fazemos novamente outra junção natural com a tabela 'MANGA'(através do atributo 'MangaId').Depois selecionamos o 'GenreId' e o 'Label' de uma determinada Manga ('MangaId=%s').Por fim ordenamos por ordem crescente nome do género ('Label').

```
@APP.route('/mangas/search/<expr>/')
def search_manga(expr):
  search = { 'expr': expr }
  expr = '%' + expr + '%'
  mangas = db.execute(
      '''
      SELECT MangaId, manga_name
      FROM MANGA
      WHERE manga_name LIKE %s
      ''', expr).fetchall()
  return render_template('manga-search.html',
                         search=search,mangas=mangas)
```

*Figura 4 - screenshot de app.py (/mangas/search/<expr>/)*

1. ***SELECT MangaId, manga_name***

   ***FROM MANGA***

   ***WHERE manga_name LIKE %s***

Selecione o ‘MangaId’ e ‘manga_name’ da tabela ‘MANGA’ onde o nome da manga (‘manga_name’) assemelhe-se ou iguale uma dada expressão.

```python
#Anime
@APP.route('/animes/')
def list_animes():
    animes = db.execute(
      '''
      SELECT AnimeId,animeName,episode_number,animeRating,MangaId,StudioId,start_date,end_date
      FROM ANIME
      ORDER BY animeName
      ''').fetchall()
    return render_template('anime-list.html', animes=animes)
```

*Figura 5 - screenshot de app.py (/animes/)*

1. ***SELECT AnimeId,animeName,episode_number,animeRating,MangaId,StudioId,start_date,end_date***

   ***FROM ANIME***

   ***ORDER BY animeName***

Seleciona todos os registos e todos os atributos da tabela ‘ANIME’ e ordena-os por ‘animeName’(ordem crescente).

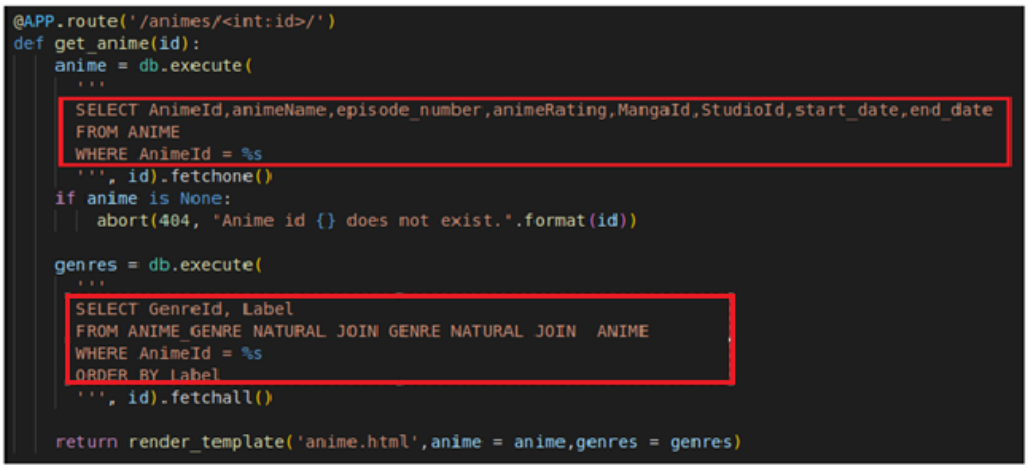


*Figura 6 - screenshot de app.py (/animes/<int:id>)*

1. ***SELECT AnimeId,animeName,episode_number,animeRating,MangaId,StudioId,start_date,end_date***

   ***FROM ANIME***

   ***WHERE AnimeId = %s***

Seleciona todos os registos e todos os atributos da tabela 'ANIME' que tenha um dado 'AnimeId'.

2. ***SELECT***

   ***GenreId, Label***

   ***FROM ANIME_GENRE NATURAL JOIN GENRE NATURAL JOIN ANIME***

   ***WHERE AnimeId = %s***

   ***ORDER BY Label***

Fazemos a junção natural entre as tabelas 'ANIME_GENRE' e 'Genre' (através do atributo 'GenreId') e fazemos novamente outra junção natural com a tabela 'ANIME'(através do atributo 'AnimeId').Depois selecionamos o 'GenreId' e o 'Label' de uma determinada Manga ('MangaId=%s').Por fim ordenamos por ordem crescente nome do género ('Label').

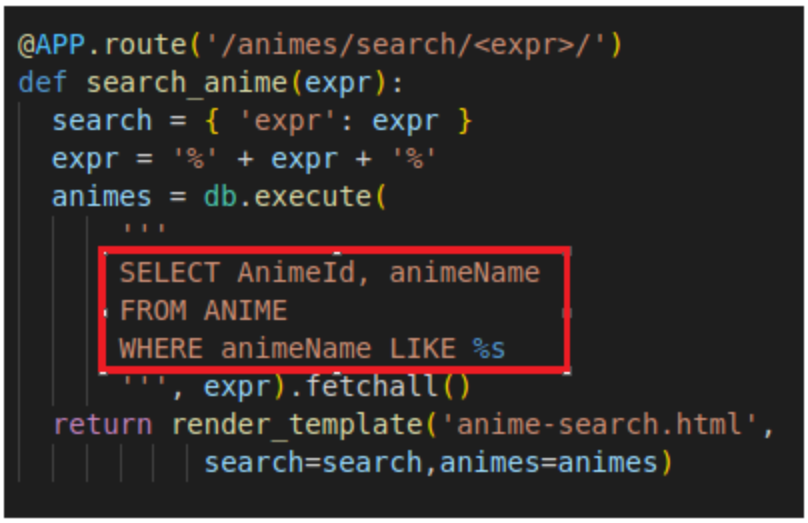


*Figura 7 - screenshot de app.py (/animes/search/<expr>/)*

1. ***SELECT AnimeId, animeName***

   ***FROM ANIME***

   ***WHERE animeName LIKE %s***

Selecione o 'AnimeId' e 'animeName' da tabela 'ANIME' onde o nome do anime ('animeName') assemelhe-se ou iguale uma dade expressão.

```python
#Genres
@APP.route('/genres/')
def list_genres():
    genres = db.execute('''
        SELECT GenreId, Label
        FROM GENRE
        ORDER BY Label
    ''').fetchall()
    return render_template('genre-list.html', genres=genres)
```

*Figura 8 - screenshot de app.py (/genres/)*

1. ***SELECT GenreId, Label***

   ***FROM GENRE***

   ***ORDER BY Label***

Seleciona todos os registos e todos os atributos da tabela 'GENRE' e ordena-os por forma crescente do nome do género ('Label').

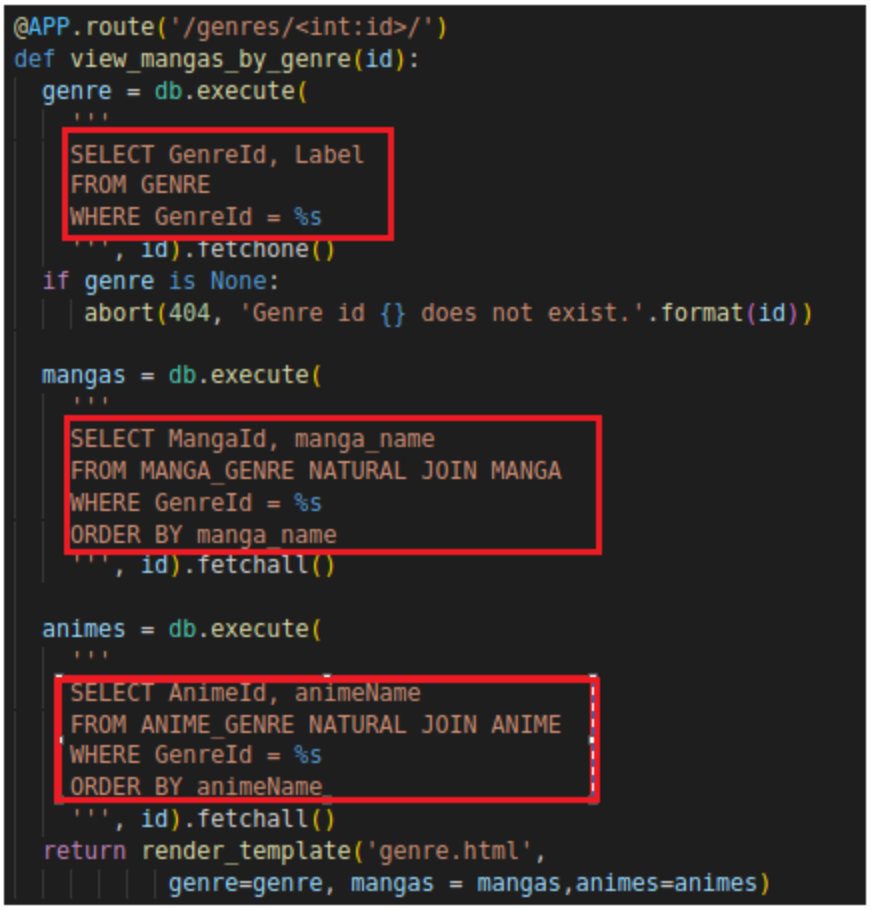


*Figura 9 - screenshot de app.py (/genres/<int:id>/)*

1. ***SELECT GenreId, Label***

   ***FROM GENRE***

   ***WHERE GenreId = %s***

Selecione todos os registos e todos os atributos da tabela 'GENRE' que tenham um dado 'GenreId'.

2. ***SELECT MangaId, manga_name***

   ***FROM MANGA_GENRE NATURAL JOIN MANGA***

   ***WHERE GenreId = %s***

   ***ORDER BY manga_name***

Fazemos junção natural das tabelas 'MANGA_GENRE' e 'MANGA' (através do atributo 'MangaId') e selecionamos os atributos 'MangaId' e 'manga_name' que tenham um dado 'GenreId'. Por fim ordenamos por ordem crescente por nome do manga ('manga_name').

3. ***SELECT AnimeId, animeName***

   ***FROM ANIME_GENRE NATURAL JOIN ANIME***

   ***WHERE GenreId = %s***

   ***ORDER BY animeName***

Fazemos junção natural das tabelas 'ANIME_GENRE' e 'ANIME' (através do atributo 'AnimeId') e selecionamos os atributos 'AnimeId' e 'animeName' que tenham um dado 'GenreId'. Por fim ordenamos por ordem crescente por nome do anime ('animeName').

```
#Authors
@APP.route('/authors/')
def list_authors():
    authors = db.execute('''
        SELECT AuthorId, authorName ,nationality,date_of_birth,date_of_death
        FROM AUTHORS
        ORDER BY authorName
    ''').fetchall()
    return render_template('author-list.html', authors=authors)
```

*Figura 10 - screenshot de app.py (/authors/)*

1. ***SELECT AuthorId, authorName ,nationality,date_of_birth,date_of_death***

   ***FROM AUTHORS***

   ***ORDER BY authorName***

Seleciona todos os registos e todos os atributos da tabela 'AUTHORS' e ordena por ordem crescente pelo nome do autor da manga 'authorName'.

```python
@APP.route('/authors/<int:id>/')
def view_manga_by_author(id):
    author = db.execute(
        '''
        SELECT AuthorId, authorName ,nationality,date_of_birth,date_of_death
        FROM AUTHORS
        WHERE AuthorId = %s
        ''', id).fetchone()

    if author is None:
        abort(404, 'Author Id {} does not exist.'.format(id))

    mangas = db.execute(
        '''
        SELECT AuthorId, manga_name
        FROM MANGA_AUTHOR NATURAL JOIN AUTHORS NATURAL JOIN MANGA
        WHERE AuthorId = %s
        ORDER BY manga_name
        ''', id).fetchall()
    return render_template('author.html', author=author,mangas=mangas)
```

*Figura 11 - screenshot de app.py (/authors/<int:id>/)*

1. **SELECT AuthorId, authorName ,nationality,date_of_birth,date_of_death**

   **FROM AUTHORS**

   **WHERE AuthorId = %s**

Seleciona todos os registos e todos os atributos da tabela 'AUTHORS' de um dado autor de manga.

2. ***SELECT AuthorId, manga_name***

   ***FROM MANGA_AUTHOR NATURAL JOIN AUTHORS NATURAL JOIN MANGA***

   ***WHERE AuthorId = %s***

   ***ORDER BY manga_name***

Fazemos a junção natural entre as tabelas 'MANGA_AUTHOR' e 'AUTHORS' (através do atributo 'AuthorId') e fazemos novamente outra junção natural com a tabela 'MANGA'(através do atributo 'MangaId').Depois selecionamos o 'AuthorId' e 'manga_name' de um determinado autor ('AuthorId=%s').Por fim ordenamos por ordem crescente nome do manga ('manga_name').

```python
@APP.route('/authors/search/<expr>/')
def search_author(expr):
    search = { 'expr': expr }
    expr = '%' + expr + '%'
    authors = db.execute(
        '''
        SELECT AuthorId, authorName
        FROM AUTHORS
        WHERE authorName LIKE %s
        ''', expr).fetchall()
    return render_template('author-search.html',
                           search=search,authors=authors)
```

*Figura 12 - screenshot de app.py (/authors/search/<expr>/)*

1. ***SELECT AuthorId, authorName***

   ***FROM AUTHORS***

   ***WHERE authorName LIKE %s***

Selecione o 'AuthorId' e 'authorName' da tabela 'AUTHORS' onde o nome de autor do manga ('authorName') assemelhe-se ou iguale uma dada expressão.

```python
#Studio
@APP.route('/studios/')
def list_studios():
    studios = db.execute('''
        SELECT StudioId, studioName ,studioName,studioOwner,LCountry,LCity
        FROM STUDIOS
        ORDER BY studioName
    ''').fetchall()
    return render_template('studio-list.html', studios =studios)
```

*Figura 13 - screenshot de app.py (/studios/)*

1. ***SELECT StudioId, studioName ,studioName,studioOwner,LCountry,LCity***

   ***FROM STUDIOS***

   ***ORDER BY studioName***

Seleciona todos os registos e todos os atributos da tabela 'STUDIOS' e ordena por ordem crescente usando nome de estúdio ('studioName').

```
@APP.route('/studios/<int:id>/')
def view_animes_by_studio(id):
    studio = db.execute(
        '''
        SELECT StudioId, studioName,studioOwner,LCountry, LCity
        FROM STUDIOS
        WHERE StudioId = %s
        ''', id).fetchone()

    if studio is None:
        abort(404, 'Studio id {} does not exist.'.format(id))

    animes = db.execute(
        '''
        SELECT AnimeId, animeName
        FROM ANIME NATURAL JOIN STUDIOS
        WHERE StudioId = %s
        ORDER BY animeName
        ''', id).fetchall()

    return render_template('studio.html',
                           studio=studio, animes=animes)
```

*Figura 14 - screenshot de app.py (/studios/<int:id>/)*

1. **SELECT StudioId, studioName,studioOwner,LCountry, LCity**

   **FROM STUDIOS**

   **WHERE StudioId = %s**

Seleciona todos os registos e todos os atributos da tabela 'STUDIOS' que tenham um dado 'StudioId'.

2. ***SELECT AnimeId, animeName***

   ***FROM ANIME NATURAL JOIN STUDIOS***

   ***WHERE StudioId = %s***

   ***ORDER BY animeName***

Fazemos junção natural das tabelas 'ANIME' e 'STUDIOS' (através do atributo 'StudioId') e selecionamos os atributos 'AnimeId' e 'animeName' que tenham um dado 'StudioId'. Por fim ordenamos por ordem crescente por nome do anime ('animeName').

```
@APP.route('/studios/search/<expr>/')
def search_studio(expr):
  search = { 'expr': expr }
  expr = '%' + expr + '%'
  studios = db.execute(
      '''
      SELECT StudioId, studioName
      FROM STUDIOS
      WHERE studioName LIKE %s
      ''', expr).fetchall()
  return render_template('studio-search.html',
                         search=search,studios=studios)
```

*Figura 15 - screenshot de app.py (/studios/search/<expr>/)*

1. ***SELECT StudioId, studioName***

   ***FROM STUDIOS***

   ***WHERE studioName LIKE %s***

Selecione o 'StudioId' e 'studioName' da tabela 'STUDIOS' onde o nome do estudio ('studioName') assemelhe-se ou iguale uma dada expressão.