Projeto de Base de Dados

LEI 2022/2023

Realizado por:

- Frederico Ferreira - 2021217116
- Gonçalo Monteiro - 2021217127
- Guilherme Almeida - 2019224555

Introduction

The following project was developed with the main objective to create a database and it’s respective API, to manage a music streaming platform like the users, songs and subscription plans.

To do this we used *PostgreSQL* and *pgAdmin* to manage the database and *Python* to develop the REST API, using the *flask* library.

Installation Manual

1.1 Pre-installation Steps DB:

- If PostgreSQL is not already installed, follow the instructions below to install it:
  - For Windows:
    a. Download the PostgreSQL installer from the official website: https://www.postgresql.org/download/windows/
    b. Run the installer and follow the on-screen instructions.
    c. Choose the desired installation location and components (including pgAdmin).
    d. Complete the installation process.
  - For Ubuntu:
    a. Open a terminal.
    b. Execute the following commands:

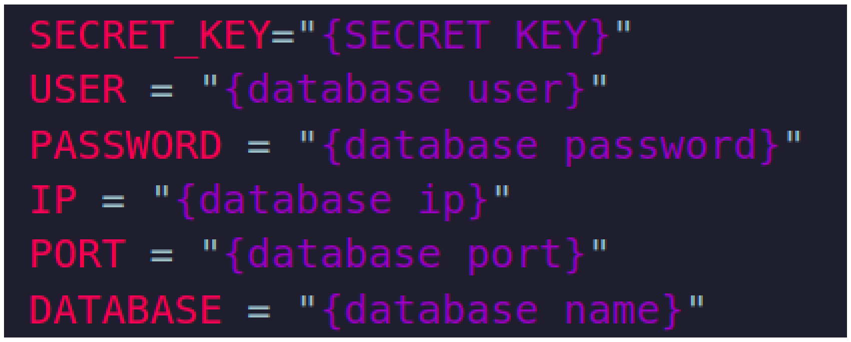
```
$sudo apt update

$ sudo apt install
postgresql

$ sudo apt install
```

  - For macOS:
    a. Download the PostgreSQL installer for macOS from the official website: https://www.postgresql.org/download/macosx/
    b. Run the installer and follow the on-screen instructions.
    c. Choose the desired installation location and components (including pgAdmin).
    d. Complete the installation process.

1.2 Database Setup:

- Once PostgreSQL is installed, open pgAdmin to manage the database.
- Create new user if needed.
- Create a new PostgreSQL database:
- Launch pgAdmin.
- Connect to the PostgreSQL server using the appropriate credentials.
- Right-click on the "Databases" node and select "Create" > "Database".

- Enter a name for the database and click "Save".
- After creating the database the user and defining the IP and the port store all of this values in a .env file with the following format:



```
SECRET_KEY="{SECRET KEY}"
USER = "{database user}"
PASSWORD = "{database password}"
IP = "{database ip}"
PORT = "{database port}"
DATABASE = "{database name}"
```

1.3 Table Creation and Population:

- Use the provided script "create_tables.sql" for this step.
- Execute the SQL script using the following steps:
- Open pgAdmin and connect to the database created in the previous step.
- Right-click on the database and select "Query Tool".
- Open the SQL script "create_tables.sql" in the query editor.
- Execute the script by clicking the "Execute" button or using the appropriate shortcut.

2 Installation Process API

2.1 Environment Setup:

- Ensure that you have Python installed on your system. You can download Python from the official website: https://www.python.org/downloads/
- Verify that pip, the package installer for Python, is also installed. You can check if pip is installed by running the following command in your terminal:

```
$ pip --version
```

- If pip is not installed, you can install it by following the instructions provided on the official Python website.

2.3 Install Dependencies:

- Open your terminal, navigate to the project directory, and execute the following command to install the required dependencies:

```
$ pip install -r requirements.txt
```

This command will install all the dependencies listed in the requirements.txt file.

- Make sure the .env referred above is created as instructed.
- Open your terminal, navigate to the project directory, and execute the following command to start the Flask development server:

```
$ python bdproj.py
```

- The Flask development server will start, and your REST API will be accessible at the specified URL (e.g., http://localhost:8080/).

User’s Manual

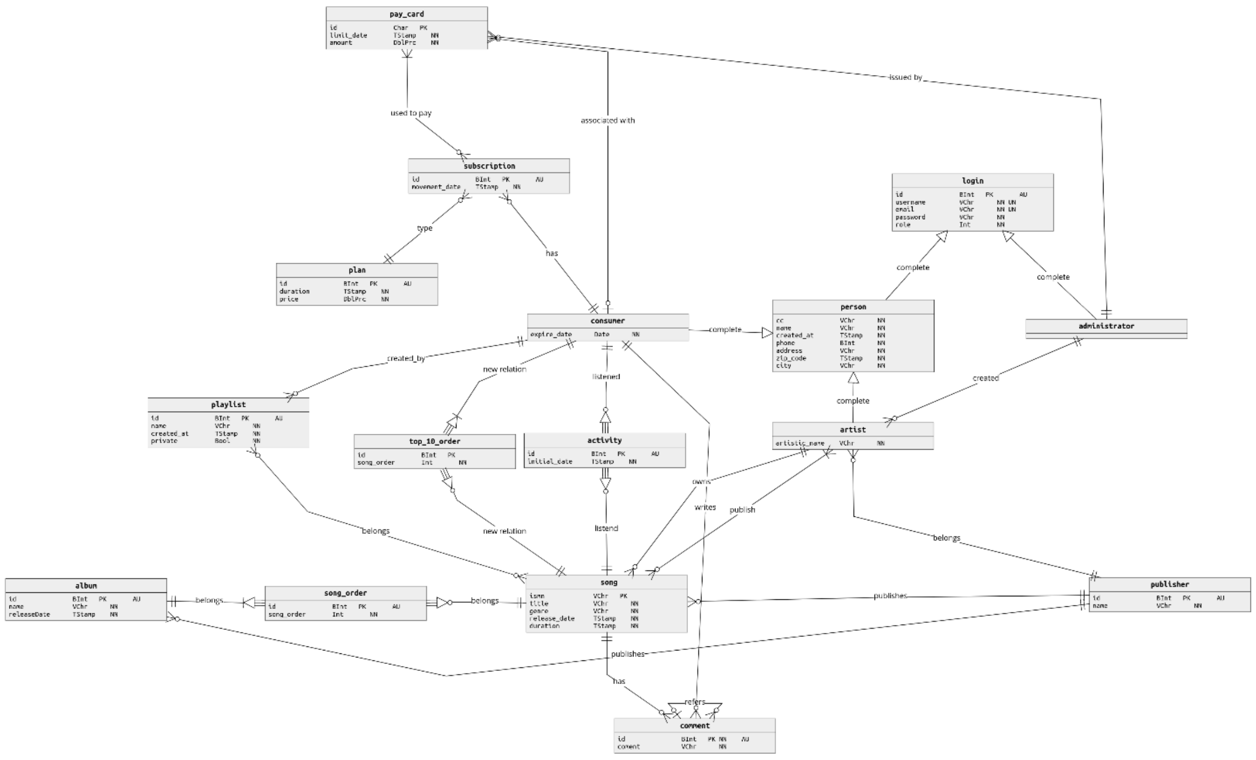
To test this application, after following the installation manual it is possible to use the API.

To test the API a postman JSON was provided with an example for all of the endpoints, every endpoint except for the login and the register requires the use of a token acquired by logging in with an account with the correct permissions.

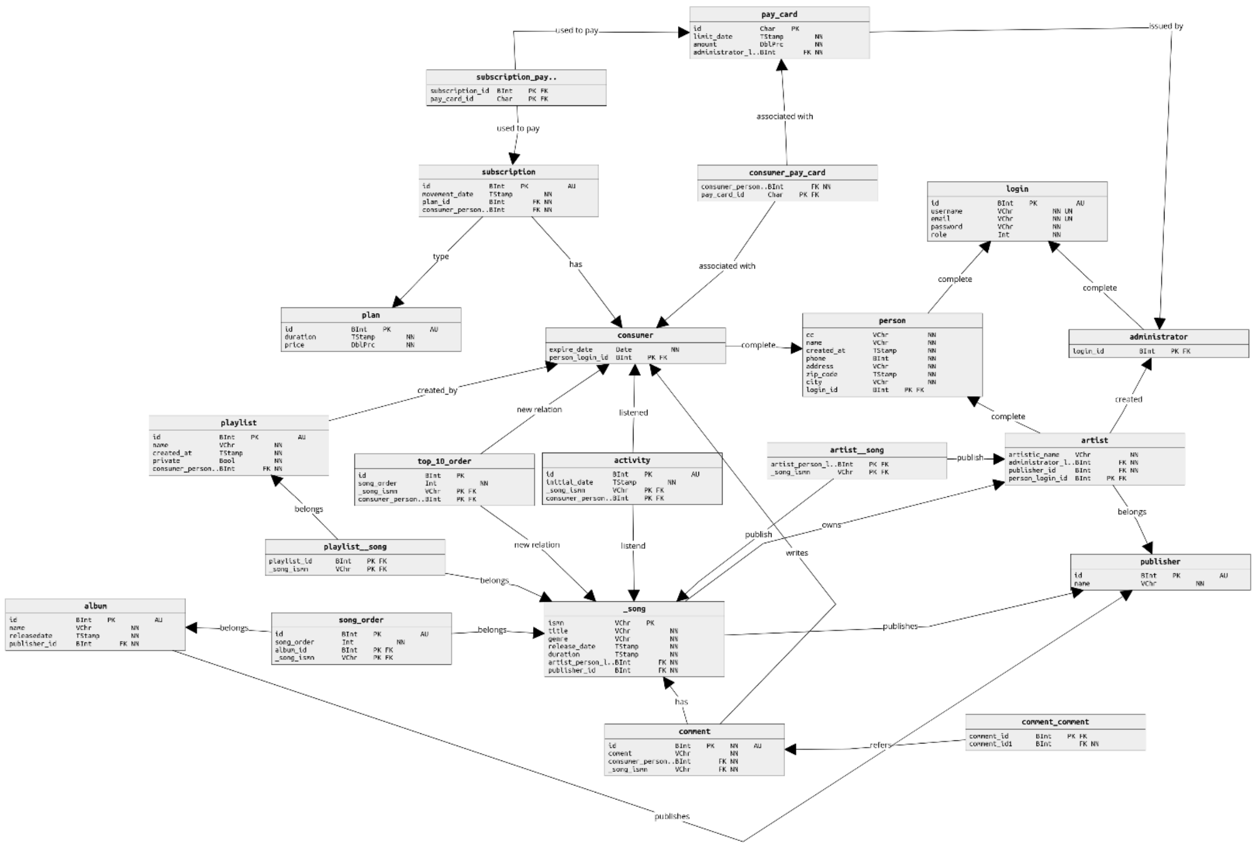
This token should be passed in the header with the name “Authorization”

Diagram

In order to produce the database the tool ONDA, found in https://onda.dei.uc.pt/v4/ was used, the following ER diagram was used as a structure for the database:



This ER resulted in the following physical diagram:



Despite ONDA providing us with functional SQL code for the table creation some changes were made, such as removing the comment_comment table and substituting it by an attribute comment_id in the table comment which is a foreign key of the own table.

The ENUM type is used for the different types of roles, types of subscription plans and values for the pay cards.

Development Plan

- ER Diagram construction and SQL script correction: 8.5 h
    - Frederico: 4 h
    - Gonçalo: 2 h
    - Guilherme: 3.5 h

- Project setup and research: 2 h
    - Frederico: 2 h
    - Gonçalo: 2 h
    - Guilherme: 2 h

- Endpoints development: 20 h
    - Frederico: 15 h
    - Gonçalo: 10 h
    - Guilherme: 10 h

- Trigger implementation: 3 h
    - Goncalo: 3 h

Extra Information

To test this database and use our PL/pgSQL knowledge we used psycopg2 which allows us to query directly the database using python variables as values for the queries, all the endpoints where build with performance in mind making the minimum amount of queries while keeping all the process safe and bug free, in order to achieve better performance some level of redundancy was added to the database, one such example would the be the addition of the column “user_privilege” in the login table, allowing us to know the role of one user with only one query instead of three. This was also done as explained above in in the comment_comment table.

To validate if a user has logged in we used a jwt token and stored inside of it the of the user and its role which allows the user to using only the token access to all the endpoints that it has access to.