FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Projeto de Base de Dados 2022/2023

Gustavo Samuel de Alves e Bastos de André e Lima - 2020217743
Nuno Ricardo Rodrigues - 2020246759
Diogo Rafael Melo Tavares - 2020236566

Manual de Instalação

Deverá instalar o Postman de forma a conseguir interagir com a API através de pedidos HTTP, visto que, no manual do utilizador, esta componente será explicada. O download deve ser realizado no site oficial. Se, por algum motivo, não conseguir realizar este passo, há algumas alternativas ao Postman que podem ser utilizadas. Para além disso, é necessário e recomendado instalar o PostgreSQL mais recente diretamente pelo site oficial e o Python.

SETUP

PGADMIN

Criar a base de dados:

Em primeiro lugar, é necessário iniciar sessão no utilizador postgres, ou em qualquer outro utilizador com permissões. Pode fazê-lo executando o seguinte comando:

```
psql -h localhost -p 5432 -d postgres -U postgres
```

Em seguida, é necessário criar a base de dados e o utilizador spotsong. Pode fazê-lo executando os seguintes comandos:

```
CREATE DATABASE dbspotsong;
CREATE USER spotsong PASSWORD 'spotsong';
GRANT ALL ON SCHEMA public TO spotsong;
```

Criar as tabelas:

Agora que criou a base de dados e o utilizador, tem de executar o ficheiro create_tables.sql (geradas pelo diagrama ER que desenvolvemos na meta 1), o ficheiro create_trigger.sql e o ficheiro insert_data.sql. Pode fazê-lo executando os seguintes comandos:

```
psql -h localhost -p 5432 -d dbspotsong -U spotsong -f create_tables.sql
psql -h localhost -p 5432 -d dbspotsong -U spotsong -f create_trigger.sql
psql -h localhost -p 5432 -d dbspotsong -U spotsong -f insert_data.sql
```

Importação de bibliotecas:

Ao nível da importação de bibliotecas, as principais que utilizámos foram o Flask e o Psycopg. Também é necessário executar um dos seguintes comandos:

```
Windows: py -m pip install -r requirements.txt
```

```
Linux/mac: pip install -r requirements.txt
```

Execute o programa. Certifique-se de estar na pasta src do codigo fonte no qual tem o ficheiro main.py e execute o seguinte comando:

```
python3 main.py
```

Manual do Utilizador

Operações Principais:

Administrator

- Criação de contas de artista
- Criação de um número específico de cartões pré-pagos (10, 25 ou 50)
- Criação de etiquetas

Consumer

- Criação de uma conta
- Autenticação de uma conta
- Compra de uma subscrição (mês, trimestre ou semestre) utilizando cartões pré-pagos
- Lista de reprodução com as 10 músicas mais tocadas (últimos 30 dias)
- Comentários às músicas ou a outros comentários
- Acesso a todas as músicas, álbuns e listas de reprodução públicas

Consumer w/ subscription

- Criação de listas de reprodução (públicas ou privadas)

Artist

- Criação de álbuns
- Criação de músicas

POST http://localhost:8080/dbproj/user

Criação/registo de uma conta:

BODY (json): { "username": "nuno", "email": "nuno@uc.pt", "password": "1234", "address": "rua 25 de abril", "contact": "+351 919 949 969", "name":"Nuno Rodrigues" }

RESPONSE (json): { "results": "Inserted user 5", "status": 200 }

PUT http://localhost:8080/dbproj/user

Autenticação de um utilizador:

BODY (json): { "username": "gustavo", "password": "1234" }

RESPONSE (json): { "results": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo3LCJleHAiOjE2ODY5NDA1NzkuMTY2MDEzfQ.BGbdg_qOj9mJXr-_sw8mreUZzKmo-Bg3gjranBtqGwA", "status": 200 }

POST http://localhost:8080/dbproj/song

Adição de uma música:

BODY (json): { "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo3LCJleHAiOjE2ODY5NDA1NzkuMTY2MDEzfQ.BGbdg_qOj9mJXr-_sw8mreUZzKmo-Bg3gjranBtqGwA", "song_name": "Décima Sinfonia", "duration": "04:30", "genre": "funk", "release_date": "2023-06-11 10:30:00", "publisher_id": 1, "other_artists": [3] }

RESPONSE (json): { "results": "Inserted song 2", "status": 200 }

POST http://localhost:8080/dbproj/album

Adição de um álbum:

BODY (json): { "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo3LCJleHAiOjE2ODY5NDA4OTIuNDQ0NjcxfQ.7avhDZ8BcwZQ6uRm90vC2wzG6cg9QXWT81qbZ2yZkRA", "release_date": "2023-06-11 10:30:00", "publisher_id": 1, "album_name": "Oceano 404", "songs": [2, { "song_name": "Guitarra azul", "duration": "2023-06-11 10:30:00", "genre": "classica", "release_date": "2023-06-11 10:30:00", "publisher_id": 1, "other_artists": [3] }] }

RESPONSE (json): { "results": "Inserted album 3", "status": 200 }

GET http://localhost:8080/dbproj/song/{keyword}

Procura de uma música:

BODY (json): {
"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo1LCJleHAiOjE2ODY5NDA5NjUuMTIwODQ0fQ.SFsnPMIAIxiQYndqqKuFEinTfVc3lyAu-wds32NIC1M"
}

RESPONSE (json): { "results": [{ "artists": ["artisteiro"], "title": "opa ganda style" }, { "artists": ["MC Gustavo", "bluedays"], "title": "Décima Sinfonia" }, { "artists": ["bluedays", "MC Gustavo"], "title": "Guitarra azul" }, { "albuns": [1, 3] }], "status": 200 }

GET http://localhost:8080/dbproj/artist_info/{artist_id}

Lista de informações relevantes de um artista específico:

BODY (json): {
"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo1LCJleHAiOjE2ODY5NDMyNDguMjAyOTQ1fQ.uUozkfpqAzZwa86V-OvopSPD5mX15cTH0PGLxq_iDQo" }

RESPONSE (json): { "results": { "albuns": [3], "name": "MC Gustavo", "playlists": [3], "songs": [2, 3] }, "status": 200 }

POST http://localhost:8080/dbproj/subscription

Criação de uma subscrição:

BODY (json): { "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo1LCJleHAiOjE2ODY5NDIxNDguODU2MDA2fQ.9FzTnp5JcSS-CP1WwhrDrM_rprYEkffOMgNt99e7vVk",
"period": "month", "cards": ["UGLFZ0568EB60W44", "1KUHKC6CA6GTE0ZX"] }

RESPONSE (json): { "results": "subscription 1 created", "status": 200 }

POST http://localhost:8080/dbproj/playlist

Criação de uma playlist:

BODY (json): { "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo1LCJleHAiOjE2ODY5NDE1MzcuNTE1OTk5fQ.NXEwcDHzNgXo5sPaCDSTvyaQhaxcPQaxxxF51r-ZFAs",
"playlist_name": "minhas Favoritas", "visibility": "public", "songs": [1,2,3] }

RESPONSE (json): { "results": "Inserted playlist 3", "status": 200 }

PUT http://localhost:8080/dbproj/{song_id}

Ouvir uma música (dar view):

BODY (json): {
"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo1LCJleHAiOjE2ODY5NDIzMDkuNjQ3OTE1fQ.fHJe7FrHQQwXuV6jdX9jA_kOkFfe_XDTleO5WsvTe6U" }

RESPONSE (json): { "results": { "message": "Song view added successfully", "view_id": 52 }, "status": 200 }

POST http://localhost:8080/dbproj/card

Geração de um cartão pré-pago:

BODY (json): {
"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoxLCJleHAiOjE2ODY5NDE1NzYuNTY4MTkyfQ.5QubMoRZleD1ThH62311ssybi47fJ1Ef2Z23vUsgXHI", "number_cards":"10", "card_price":50 }

RESPONSE (json): { "results": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], "status": 200 }

POST http://localhost:8080/dbproj/comments/{song_id}

Comentário a uma música:

BODY (json): { "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo1LCJleHAiOjE2ODY5NDA5NjUuMTIwODQ0fQ.SFsnPMIAIxiQYndqqKuFEinTfVc3lyAu-wds32NIC1M",
"comment": "Estou a comentar uma música" }

RESPONSE (json): { "results": "Inserted comment 1", "status": 200 }

POST http://localhost:8080/dbproj/comments/{song_id}/{parent_comment_id}

Comentário em resposta a um comentário de uma música:

BODY (json): { "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo1LCJleHAiOjE2ODY5NDA5NjUuMTIwODQ0fQ.SFsnPMIAIxiQYndqqKuFEinTfVc3lyAu-wds32NIC1M", "comment": "Estou a comentar um comentário" }

RESPONSE (json): { "results": "Inserted comment 2", "status": 200 }

GET http://localhost:8080/dbproj/report/{year-month}

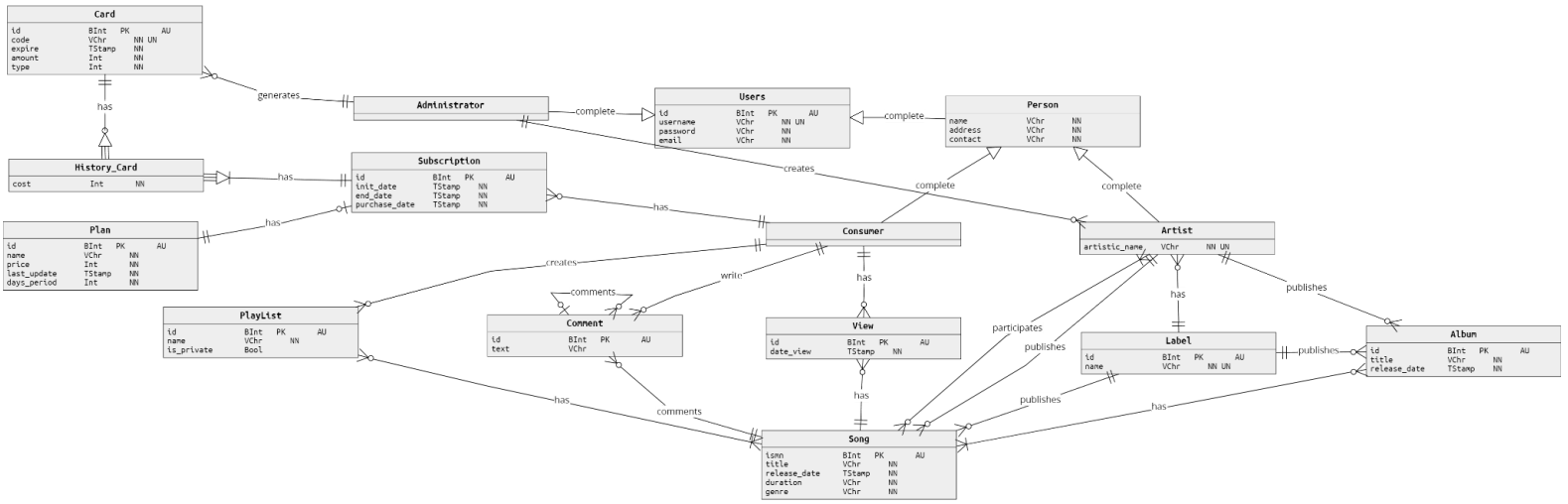
Lista de músicas tocadas por mês e género:

BODY (json): { "token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjo1LCJleHAiOjE2ODY5NDI5MzYuMjU1OTY3fQ.QTznB7mjPO1XoIZz8F-yOk3Mrgv7FjoqDO-SZMARHbc" }

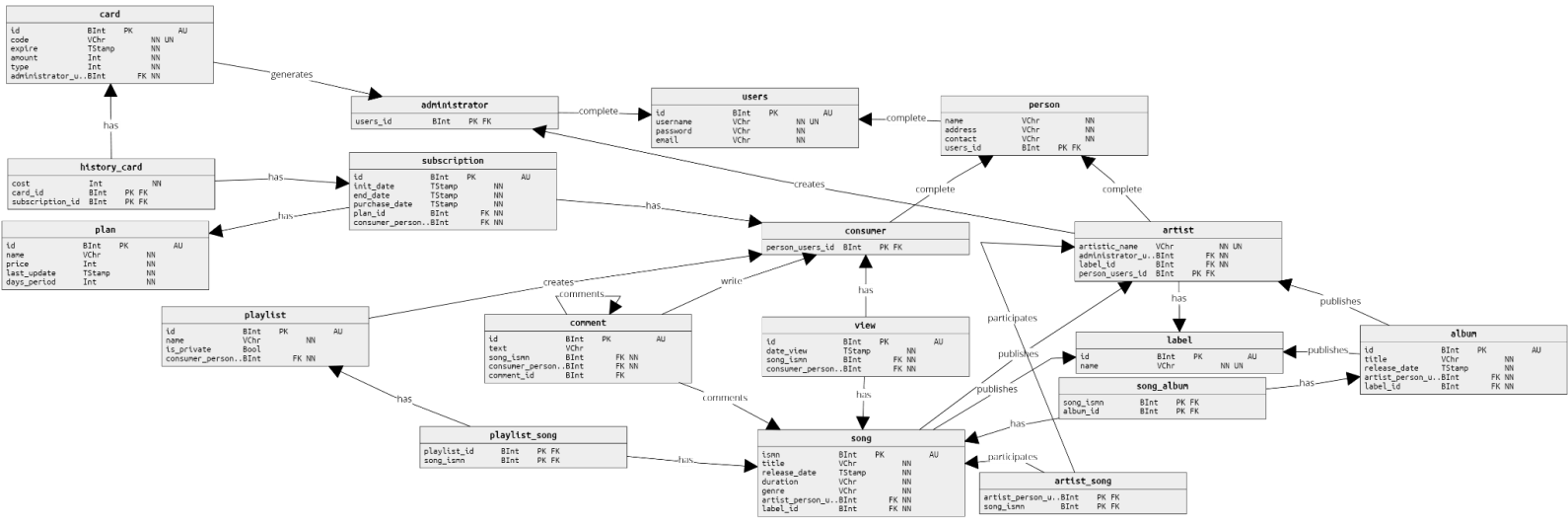
RESPONSE (json): { "results": [{ "genre": "classica", "month": "1", "playbacks": 2 }, { "genre": "funk", "month": "1", "playbacks": 2 }, { "genre": "funk", "month": "3", "playbacks": 7 }, { "genre": "funk", "month": "5", "playbacks": 3 }, { "genre": "rap", "month": "5", "playbacks": 4 }, { "genre": "classica", "month": "7", "playbacks": 1 }, { "genre": "classica", "month": "9", "playbacks": 3 }, { "genre": "classica", "month": "10", "playbacks": 7 }], "status": 200 }

ER Final e Modelo de dados relacional

Diagrama ER Final:



Modelo de Dados Relacional:



De notar que realizámos várias alterações do diagrama ER em relação à meta 1, nomeadamente na lógica das subscrições e “plays” de uma música, de modo a conseguir melhorar e simplificar, não somente o diagrama, como também algumas funcionalidades subjacentes.

Plano de Desenvolvimento

O projeto foi elaborado em simultâneo, através de reuniões online. O ER desenvolvido na meta 1 foi discutido novamente e aprimorado. De notar que houve articulação de todos os elementos do grupo para a realização das funcionalidades, visto que realizámos várias reuniões de modo a discutir o que era necessário melhorar no diagrama ER, e a implementação das funcionalidades e realização de testes e pedidos/respostas no POSTMAN.

Detalhes e informações adicionais relevantes

Utilizámos um trigger para atualizar a tabela de playlists (playlist "Top10") sempre que ouvimos uma música. Esse trigger vai verificar se essa playlist já existe, e em caso negativo, cria uma nova playlist com as novas músicas que estamos a ouvir. Em caso afirmativo, vamos contar as views que cada música tem para o utilizador atual nos últimos 30 dias, e atualizamos a tabela no caso do top10 alterar. Este trigger é chamado sempre que um utilizador ouve uma música.

Conclusão

Em conclusão, o nosso projeto focou-se no desenvolvimento de uma aplicação de música, envolvendo uma Base de Dados que armazena e processa todos os dados relacionados com a mesma. Este trabalho foi importante uma vez que nos permitiu aprimorar os conceitos relacionados com Base de Dados e a perceber melhor como um sistema de base de dados funciona, bem como a programação em PL/pgSQL.

Para garantir a robustez e eficiência da aplicação, utilizamos o PostgreSQL com um DBMS relacional transacional. Implementámos uma arquitetura de aplicação de base de dados distribuída, fornecendo uma API REST, e permitindo a comunicação entre o cliente e o servidor. Utilizámos ainda a ferramenta POSTMAN, que funciona como um cliente, enviando pedidos HTTP para o nosso servidor, para processarmos a informação corretamente.