Relatório Projeto BD

Autores:

Alexandre Ferreira 2021236702

Eduardo Pereira 2021233890

João Tinoco 2021223708

Manual de Instalação

Requisitos do Sistema:

- Certifique-se de que o seu sistema atende aos seguintes requisitos:
  - Python instalado
  - Postman instalado (para testar os requests)
  - Dependências do projeto instaladas (Bibliotecas)

Bibliotecas Utilizadas:

- flask: É um framework web em Python que permite criar aplicativos web
- Flask-JWT-Extended: É uma extensão do Flask para lidar com autenticação e autorização usando tokens JWT (JSON Web Tokens).
- psycopg2: É uma biblioteca de adaptação de base de dados PostgreSQL para Python. É usada para conectar e interagir com a base de dados PostgreSQL.
- hashlib: É uma biblioteca em Python que fornece funções de hash criptográfico, como MD5, SHA1, SHA256, entre outros.
- secrets: É uma biblioteca em Python que fornece funções para geração de números aleatórios seguros e segredos criptográficos.
- random: É uma biblioteca em Python que fornece funções para geração de números aleatórios.
- datetime: É uma biblioteca em Python que fornece classes para manipulação de datas e horas.

Configuração do Programa:

- Abra o ficheiro "config.txt" no diretório do projeto.
- Edite o ficheiro e preencha os seguintes parâmetros:
  - **user**: É o nome do utilizador usado para autenticação na base de dados
  - **password**: É a senha associada ao utilizador para autenticação na base de dados.
  - **host**: É o endereço IP ou o nome do host onde a base de dados está hospedada. (Exemplo: “127.0.0.1”, refere-se ao endereço de loopback, indicando que a base de dados está sendo executada localmente na máquina)
  - **port**: É a porta na qual a base de dados está configurada para aceitar conexões. (Exemplo: “5432”, porta padrão para o PostgreSQL)
  - **database**: É o nome da base de dados ao qual se deseja conectar.

Execução do Programa:

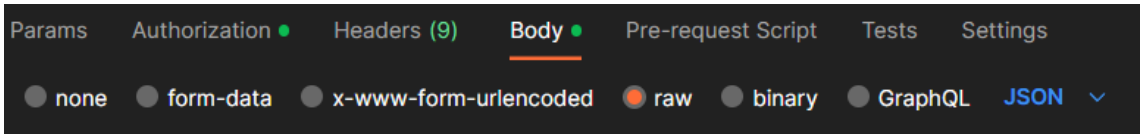
- Após fazer as alterações necessárias, o programa poderá ser executado.
- O servidor será iniciado e estará pronto para receber as requisições.

Utilização do Postman:

- Abra o Postman no seu sistema.
- Para cada uma das ações mencionadas, utilize o método correto e a URL fornecida para realizar os requests.
- Adicione os parâmetros necessários no corpo do request, seguindo a estrutura especificada em cada ação.
- Clique em "SEND" para realizar o request e visualizar a resposta.

Manual de Utilizador

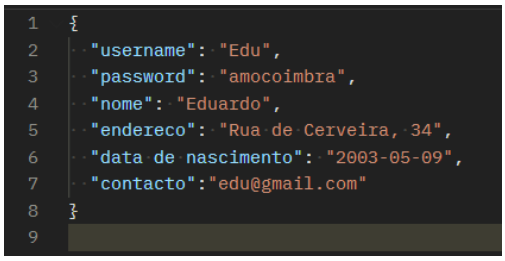
Em qualquer pedido de adição ou alteração, escolha a opção “Body”, “raw” e “JSON”, para que seja possível enviar as informações necessárias à correta execução dos requests.



Para que as funcionalidades do programa sejam utilizadas no seu total, é necessário adicionar à base de dados registos de pelo menos uma gravadora e um administrador.

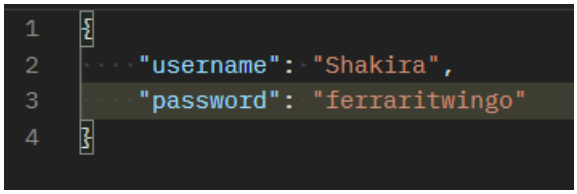
Registo do Consumidor:

- Utilize o método POST e a seguinte URL para realizar o request: “http://localhost:8080/create”
- No corpo do request, adicione os dados do consumidor, como nome, email, senha, etc.
- Clique em "SEND" para realizar o request.



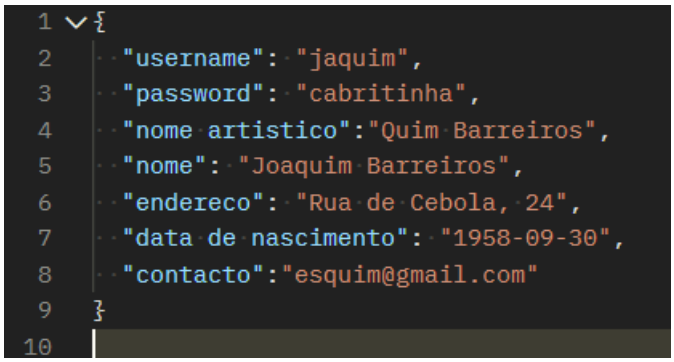
Autenticação de Utilizador:

- Utilize o método PUT e a seguinte URL para realizar o request: “http://localhost:8080/song”
- No corpo do request, adicione o nome e senha do utilizador.
- Clique em "SEND" para realizar o request.
- Se os dados introduzidos estiverem corretos é impresso um token que autoriza as restantes funcionalidades.



Registo do Artista (apenas disponível para administradores**)**:

- Introduza o token previamente obtido na área “Authorization”, com a opção “Bearer Token”
- Utilize o método POST e a seguinte URL para realizar o request: “http://localhost:8080/create”
- No corpo do request, adicione os dados do artista, como nome, nome artístico, email, senha, etc.
- Clique em "SEND" para realizar o request.



Adicionar Música (apenas disponível para artistas**)**:

- Introduza o token previamente obtido na área “Authorization”, com a opção “Bearer Token”
- Utilize o método POST e a seguinte URL para realizar o request: http://localhost:8080/song
- No corpo do request, adicione os detalhes da música, como nome, data de lançamento, editora, artistas adicionais.
- Clique em "SEND" para realizar o request.

Adicionar Álbum (apenas disponível para artistas**)**:

- Introduza o token previamente obtido na área “Authorization”, com a opção “Bearer Token”
- Utilize o método POST e a seguinte URL para realizar o request: “http://localhost:8080/album”
- No corpo do request, adicione os detalhes do álbum, como nome, data de lançamento, editora, músicas (incluindo detalhes das novas músicas ou identificadores das músicas existentes).
- Clique em "SEND" para realizar o request.

Pesquisar Música:

- Introduza o token previamente obtido na área “Authorization”, com a opção “Bearer Token”

- Utilize o método GET e a seguinte URL para realizar o request: “http://localhost:8080/song/{keyword}” substituindo "{keyword}" pela palavra-chave desejada.
- Clique em "SEND" para realizar o request.

Detalhes do Artista:

- Introduza o token previamente obtido na área “Authorization”, com a opção “Bearer Token”
- Utilize o método GET e a seguinte URL para realizar o request: “http://localhost:8080/artist_info/{artist_id}”
  substituindo "{artist_id}" pelo identificador do artista desejado.
- Clique em "SEND" para realizar o request.

Assinar como Premium (apenas disponível para consumidores**)**:

- Introduza o token previamente obtido na área “Authorization”, com a opção “Bearer Token”
- Utilize o método POST e a seguinte URL para realizar o request: “http://localhost:8080/subscription”
- No corpo do requestd, adicione o período desejado ("month", "quarter" ou "semester") e os números dos cartões pré-pagos.
- Clique em "SEND" para realizar o request.

Criar Playlist (apenas disponível para consumidores**)**:

- Introduza o token previamente obtido na área “Authorization”, com a opção “Bearer Token”
- Utilize o método POST e a seguinte URL para realizar o request: “http://localhost:8080/playlist”
- No corpo do request, adicione o nome da playlist, visibilidade ("public" ou "private") e os identificadores das músicas desejadas.
- Clique em "SEND" para realizar o request.

Reproduzir Música (apenas disponível para consumidores**)**:

- Introduza o token previamente obtido na área “Authorization”, com a opção “Bearer Token”
- Utilize o método PUT e a seguinte URL para realizar o request: “http://localhost:8080/{song_id}”
  substituindo "{song_id}" pelo identificador da música desejada.
- Clique em "SEND" para realizar o request.

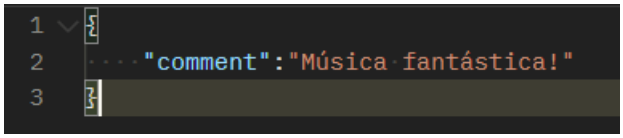
Gerar Cartões Pré-pagos:

- Introduza o token previamente obtido na área “Authorization”, com a opção “Bearer Token”
- Utilize o método POST e a seguinte URL para realizar a requisição: “http://localhost:8080/card”
- No corpo do request, adicione o número de cartões desejado e o valor do cartão ("10", "25" ou "50")
- Clique em "SEND" para realizar o request.

Deixar Comentário/Feedback:

- Introduza o token previamente obtido na área “Authorization”, com a opção “Bearer Token”
- Utilize o método POST
- Se quiser comentar um comentário utilize a seguinte URL: “http://localhost:8080/comments/{song_id}/{parent_comment_id}”, substituindo "{song_id}" pelo identificador da música desejada e {parent_comment_id} pelo identificador do comentário que irá comentar.
- Se quiser comentar uma música utilize a seguinte URL: “http://localhost:8080/comments/{song_id}”, substituindo "{song_id}" pelo identificador da música desejada.

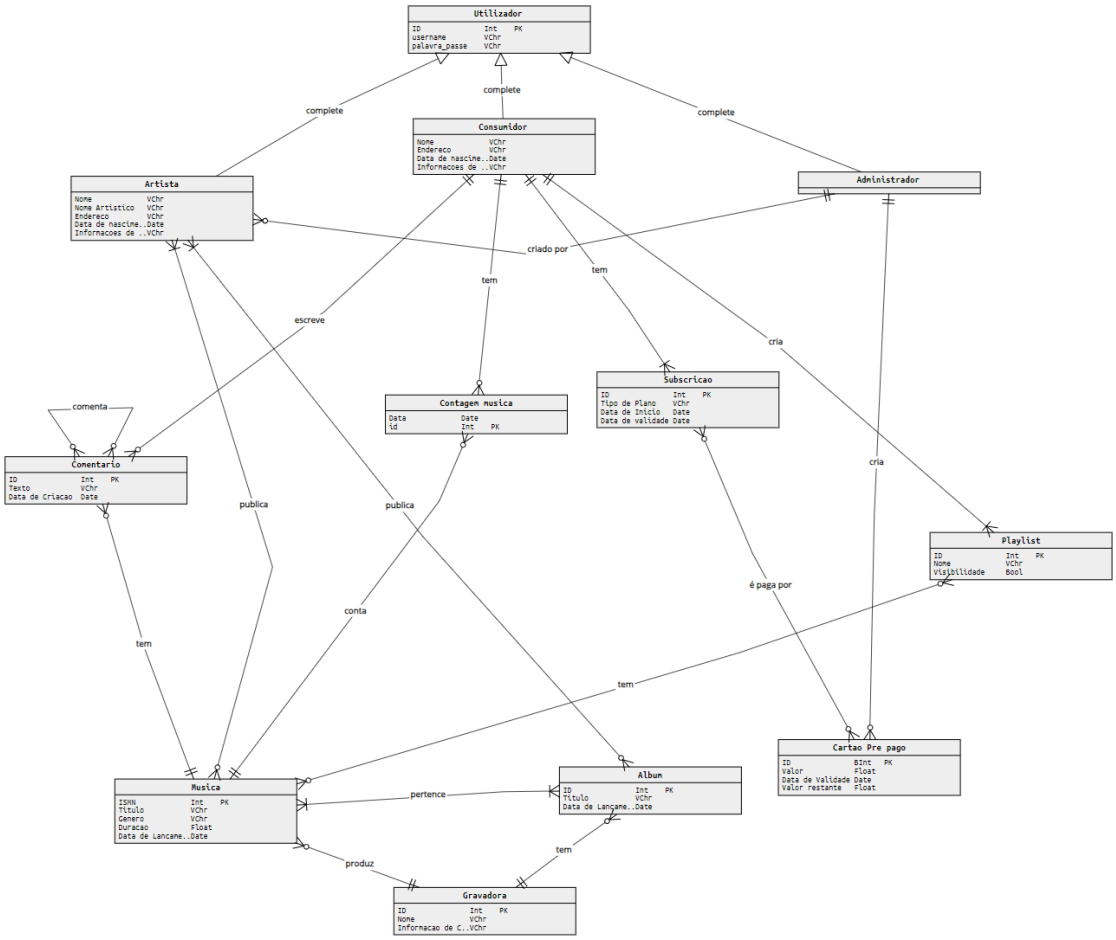
- No corpo do request, adicione o conteúdo do comentário/feedback.
- Clique em "SEND" para realizar o request.



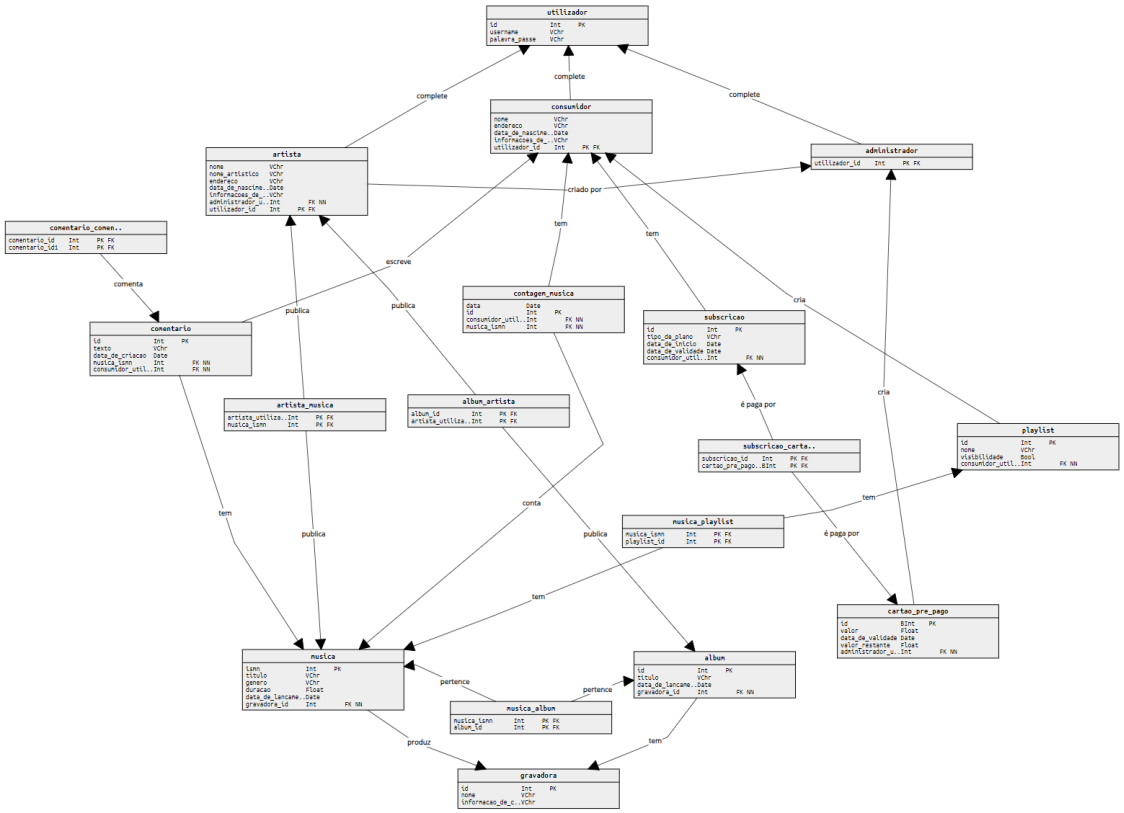
Gerar Relatório Mensal:

- Introduza o token previamente obtido na área “Authorization”, com a opção “Bearer Token”
- Utilize o método GET e a seguinte URL para realizar o request: “http://localhost:8080/report/{year-month}”, substituindo "year" pelo ano e "month" pelo mês que deseja obter o relatório mensal com o número de músicas reproduzidas por mês e por gênero nos últimos 12 meses.
- Clique em "SEND" " para realizar o request

ER Conceptual


Utilizador
ID Int PK
username VChr
palavra_passe VChr
complete
complete
complete
Consumidor
Nome VChr
Endereco VChr
Data de nascime..Date
Informacoes de ..VChr
Artista
Nome VChr
Nome Artistico VChr
Endereco VChr
Data de nascime..Date
Informacoes de ..VChr
Administrador
criado por
tem
tem
escreve
cria
Contagem musica
Data Date
id Int PK
Subscricao
ID Int PK
Tipo de Plano VChr
Data de Inicio Date
Data de validade Date
comenta
Conentario
ID Int PK
Texto VChr
Data de Criacao Date
publica
publica
cria
Playlist
ID Int PK
Nome VChr
Visibilidade Bool
é paga por
conta
tem
tem
Cartao Pre pago
ID BInt PK
Valor Float
Data de Validade Date
Valor restante Float
Musica
ISRN Int PK
Titulo VChr
Genero VChr
Duracao Float
Data de Lancame..Date
pertence
Album
ID Int PK
Titulo VChr
Data de Lançame..Date
produz
tem
Gravadora
ID Int PK
Nome VChr
Informacao de C..VChr

ER Physical



Plano de desenvolvimento

Alexandre (15h):

- Add Song
- Add Album
- Detail artist
- Generate a monthly report
- Relatório

Eduardo(15h):

- Transaction Control
- Concurrency
- Security
- Relatório
- ER

João(15h):

- User Registration
- User Authentication
- Subscribe to Premium
- Create Playlist
- Play song
- Generate pre-paid cards
- Leave comment/feedback
- Search Song

Construção da Aplicação

Registo do Utilizador: O endpoint do registo de utilizador tem duas variações conforme a utilização do JWT Token, se for colocado um token a aplicação assume que o objetivo é adicionar um artista, apenas permitido ao administrador, logo faz uma verificação. Cria a entrada na tabela "utilizador" (codificando a palavra-passe em SHA256) e na tabela "artista" com as informações correspondentes. Se não for colocado nenhum token, a aplicação interpreta o objetivo de criar um consumidor, ou seja, adiciona uma entrada na tabela "utilizador" e na tabela "consumidor", com os devidos dados e a palavra-passe encriptada. Cria uma entrada na tabela "subscricao" com o estado da conta do utilizador, premium ou regular. Além disto, cria também uma entrada na tabela "playlist", que representa uma playlist acessada apenas pelo programa para manter registo do TOP10 de músicas mais ouvidas.

Autenticação de Utilizador: O endpoint de autenticação recebe o username e a password e procura um utilizador com o username dado. Se encontar, faz o hash da password dada (SHA256) e compara com o hash da base de dados, se coincidir é gerado o token, com o seu tipo de utilizador e o seu id na identidade, para dar acesso às restantes funcionalidades, caso contrário é mostrada uma mensagem de erro.

Adicionar Música: O endpoint "add_song" tem a funcionalidade de adicionar uma música de um artista à base de dados. Todas as informações da música são adicionadas e as tabelas de junção relacionadas à entidade "musica" são atualizadas.

Adicionar Álbum: Da mesma forma, que o endpoint "add_song", o endpoint "add_album" adiciona um álbum à base de dados. Todas as informações do álbum são adicionadas e as tabelas de junção relacionadas à entidade "album" são atualizadas.

Pesquisar Música: O endpoint "search song" obtém as informações das músicas que correspondem à palavra-chave fornecida. A consulta utiliza junções (LEFT JOIN) entre as tabelas "musica", "artista_musica", "artista", "musica_album" e "album" para obter os dados relacionados. A cláusula "WHERE m.titulo LIKE %s" filtra as músicas cujo título contenha a palavra-chave, sendo que o parâmetro %s é substituído pelo valor da palavra-chave.

O resultado da consulta é recuperado usando cur.fetchall() e armazenado na variável consulta. A função formata_search_song(consulta) é chamada para formatar os resultados da consulta em um formato desejado. Ela itera sobre os resultados e cria uma estrutura de dados que agrupa as informações por título da música, artistas e álbuns. O resultado final é uma lista de dicionários, onde cada dicionário contém informações sobre uma música, seus artistas associados e os álbuns em que a música está presente.

Detalhes do Artista: O endpoint “detail_artist” utiliza a tabela "artista" como tabela principal e é referenciada como "a" na query.

São realizados vários "LEFT JOIN" para relacionar outras tabelas com a tabela de artista:

"artista_musica" é relacionada através da coluna "artista_utilizador_id" para obter as músicas associadas ao artista, "musica" é relacionada através da coluna "ismn" para obter informações sobre as músicas do artista, "album_artista" é relacionada através da coluna "artista_utilizador_id" para obter os álbuns associados ao artista, "album" é relacionada através da coluna "id" para obter informações sobre os álbuns do artista, "musica_playlist" é relacionada através da coluna "musica_ismn" para obter as playlists em que as músicas do artista estão presentes e a tabela "playlist" é relacionada através da coluna "id" para obter informações sobre as playlists do artista.

É utilizada a função "ARRAY_AGG" para agrupar e obter uma lista distinta de músicas, álbuns e playlists relacionados ao artista, seguidamente, a cláusula "GROUP BY" é usada para agrupar os resultados pelo identificador único do artista, a cláusula "WHERE" filtra os resultados pelo identificador do artista, e por fim, acláusula "p.visibilidade=true" é adicionada ao "LEFT JOIN" com a tabela "playlist" para garantir que apenas as playlists públicas sejam consideradas.

Assinar como Premium: Para um consumidor o endpoint recebe o período desejado e os identificadores dos cartões para pagar a modalidade. Os cartões são verificados, tanto quanto ao valor como à data de validade e são utilizados para pagar a subscrição. Se os cartões permitirem a compra do pacote premium, é inserida uma entrada na tabela “subscricao” com o período de premium (data de início e data de validade) para o consumidor em questão. É também feita a associação entre os cartões utilizados e a subscrição feita através da tabela “subscricao”. Por fim se o valor do pacote premium for diferente do valor dos cartões, o valor é descontado e estes podem ser utilizados noutras compras.

Criar Playlist: O endpoint “create_playlist”, apenas disponível para consumidores premium, executa uma consulta para obter a data de validade da assinatura do consumidor específico, identificado por “consumidor_utilizador_id”. E é armazenado o resultado na variável “consulta”.

Utiliza a função “procura_tipo_de_plano()” para extrair o tipo de plano (nesse caso, "Premium") e a data de validade da variável “consulta”.

Verifica se o tipo de plano é "Premium", se for, prossegue com a criação da playlist.

Insere uma nova playlist na tabela “playlist” com o nome, visibilidade e ID do consumidor fornecidos. O comando RETURNING id retorna o ID gerado para a playlist recém-criada, que é armazenado em “id_playlist”.

Itera sobre as músicas fornecidas em “payload['songs']” e insere cada uma delas na tabela “musica_playlist”, associando-as à playlist recém-criada.

Reproduzir Música: Ao reproduzir uma música, é adicionada uma entrada na tabela “contagem_musica” com o utilizador atual e a data da reprodução. Ao inserir nesta um trigger é disparado e a playlist TOP10 é atualizada. A função presente no trigger elimina todas as entradas da tabela “musica_playlist” correspondentes à playlist TOP10 do utilizador atual. E de seguida procura na tabela “contagem_musica” pelas 10 músicas que mais ocorrem para o utilizador atual. Após isto, essas músicas são colocadas de volta na tabela “musica_playlist”, mas tendo em conta as novas contagens.

```sql
CREATE OR REPLACE FUNCTION atualizar_top_10()
RETURNS TRIGGER AS $$
BEGIN
    -- Limpar as entradas anteriores na tabela musica_playlist
    DELETE FROM musica_playlist
    WHERE playlist_id IN (
        SELECT id
        FROM playlist
        WHERE nome = 'TOP10' AND consumidor_utilizador_id = NEW.consumidor_utilizador_id
    );

    -- Selecionar as 10 músicas mais frequentes para o consumidor
    WITH top_10 AS (
        SELECT musica_ismn
        FROM contagem_musica
        WHERE consumidor_utilizador_id = NEW.consumidor_utilizador_id
        GROUP BY musica_ismn
        ORDER BY COUNT(*) DESC
        LIMIT 10
    )
    -- Inserir as entradas na tabela musica_playlist
    INSERT INTO musica_playlist (playlist_id, musica_ismn)
    SELECT p.id, t.musica_ismn
    FROM playlist p
    CROSS JOIN top_10 t
    WHERE p.nome = 'TOP10' AND p.consumidor_utilizador_id = NEW.consumidor_utilizador_id;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

Gerar Cartões Pré-pagos: O endpoint “generate_card”, apenas disponível para adminitradores, gera um identificador único para cada cartão pré-pago usando a função “generate_random_sequence()”. O código verifica se o identificador gerado já existe na tabela “cartao_pre_pago”. Se existir, gera um novo identificador até encontrar um único.

Insere o novo cartão pré-pago na tabela “cartao_pre_pago” com o identificador, preço do cartão, data de validade (1 ano após a data atual), valor restante (mesmo que o preço do cartão) e ID do administrador fornecidos em values.

Deixar Comentário/Feedback: O endpoint “make_comment”, apenas disponível para consumidores, verifica se o “parent_id_comment” (ID do comentário pai) não é nulo. Se não for nulo, verifica se o comentário referenciado pelo “parent_id_comment” pertence à música especificada “song_id”. Se não pertencer, retorna uma resposta de erro indicando que o comentário fornecido não se refere à música fornecida.

Insere o novo comentário na tabela “comentário” com o texto do comentário, a data de criação (data atual), o ID da música e o ID do consumidor fornecidos em values. O comando RETURNING id retorna o ID do novo comentário inserido.

Se o “parent_id_comment” não for nulo, insere uma entrada na tabela “comentario_comentario” para estabelecer a relação de comentário pai-filho. O novo comentário é definido como filho do comentário referenciado pelo “parent_id_comment”.

Gerar Relatório Mensal: O endpoint “generate_monthly_report” divide a string “year_month” em duas partes separadas pelo caractere '-', obtendo o ano e o mês correspondentes.

Calcula a data inicial e final de um período de 12 meses com base no ano e mês fornecidos. A data inicial é definida como o primeiro dia do mês especificado, e a data final é definida como o primeiro dia do mês um ano antes.

Constrói uma consulta SQL que procura o número de reproduções de músicas por mês e género dentro do período de 12 meses. A consulta junta as tabelas “contagem_musica” e “musica” usando o campo “musica_ismn” como chave de ligação. A cláusula WHERE filtra as reproduções de música que ocorrem entre a data inicial e final do período. A consulta agrupa os resultados por mês e género, e os ordena por mês e número de reproduções em ordem descendente.

Formata os resultados em uma lista de dicionários, onde cada dicionário representa um resultado com as chaves 'month' (mês), 'genre' (género) e 'playbacks' (número de reproduções).

Detalhes e Decisões de Construção da Aplicação

Triggers: Nesta aplicação é apenas utilizado um trigger (anteriormente descrito com mais detalhe) para atualizar a playlist TOP10 de cada utilizador quando este reproduz uma música.

Transações: De forma a garantir a integridade e consistência dos dados durante as realizações das várias operações na base de dados, foi necessário bloquear as tabelas em modo exclusivo. No caso do endpoint “add_song”, a transação é iniciada com a linha “cur.execute('BEGIN')”, indicando o início da transação. Em seguida, há um bloqueio na tabela "artista_musica" e "musica" usando “cur.execute('LOCK TABLE artista_musica,musica IN EXCLUSIVE MODE')”. Isso é feito para garantir que nenhum outro processo acesse ou modifique essas tabelas enquanto a transação estiver em andamento. Os restantes endpoints, caso estes impliquem a modificação dos dados da base de dados, utilizam a mesma lógica.

Se todas as operações forem executadas com sucesso, a transação é confirmada usando “conn.commit()”. No entanto, se ocorrer algum erro durante a execução da transação, o bloco except será acionado, onde o código realiza o rollback da transação usando “conn.rollback()”. Isso desfaz todas as operações realizadas na transação e repõe a base de dados ao estado anterior à transação.

Portanto, a transação garante que todas as operações sejam tratadas como uma unidade atômica, permitindo que todas sejam confirmadas ou desfeitas em conjunto, garantindo a integridade e consistência dos dados.

Proteções: Para garantir a proteção e a integridade da base de dados, foi adotada a prática de parametrização dos dados nas consultas SQL. Além disso, foram implementadas verificações adicionais para identificar a presença do caractere ";" nos dados. Essas verificações são realizadas pelas funções auxiliares "check_payload" e "ponto_virgula_recursivo". Essas medidas têm como objetivo prevenir a ocorrência de injeção de comandos SQL maliciosos.

Id’s sequenciais: De forma a que o programa execute corretamente, a base de dados necessita de configurações extra. Os ID’s de várias tabelas são identificadores únicos gerados automaticamente e sequencialmente pela base de dados à medida que novos registros são inseridos nas tabelas. O script que atende às necessidades dos id´s sequenciais encontra-se identificado no ficheiro “comandosBaseDados.txt”.

Decisões: Quando um consumidor é adicionado é também criado uma entrada na tabela “playlist” com uma playlist “TOP10” para armazenar as 10 músicas mais ouvidas. Além disto, é também criada uma entrada na tabela “subscricao” que representa o estado da assinatura do utilizador, que por default, é indicada como “regular”. Em relação à

reprodução de músicas, a tabela “contagem_musicas” contém todos os registos de reproduções com a data de reprodução, o id da música e o id do utilizador.