UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Repositório De Streaming

Dennis Lemke Green	11219108
Marcus Vinícius Teixeira Huziwara	11834432
Victor Paulo Cruz Lutes	11795512
Vinícius Santo Monteiro	11932463
Pedro Henrique Conrado	11819091

SCC0240 - Bases de Dados
Profª Elaine Sousa

São Carlos, 2022

Sumário

Descrição do Problema e dos Requisitos de Dados

Com o número crescente de plataformas disponíveis para assistir filmes e séries online (as streamings), como Netflix, HBO Max, Disney Plus, entre muitas outras; e cada uma com uma lista diferente de conteúdo disponíveis, julgou-se necessário criar uma plataforma para unificar essas informações e auxiliar os usuários dessa plataforma em seu direcionamento e gerenciamento dos filmes e séries que ele já assistiu ou planeja assistir; assim como auxiliar na utilização de redes privadas virtuais (VPN) para acessar conteúdos por meio de outros países. Os usuários alvo são pessoas que possuem assinaturas em vários streamings e querem gerenciar e/ou organizar as mídias que ele assistiu em quais streamings.

Nosso projeto tem como objetivo: centralizar as informações dos filmes e das séries associando a elas a sua disponibilidade em diferentes plataformas e países para que os usuários possam cadastrar quais plataformas que ele assina e o país onde ele acessa essas plataformas, permitir que os usuários cadastrem quais VPNs ele utiliza, permitir que os usuários listem quais filmes e séries ele possui acesso (referente aos seus respectivos streamings) e fazer uma busca por um filme ou série e descobrir onde ele estará disponível.

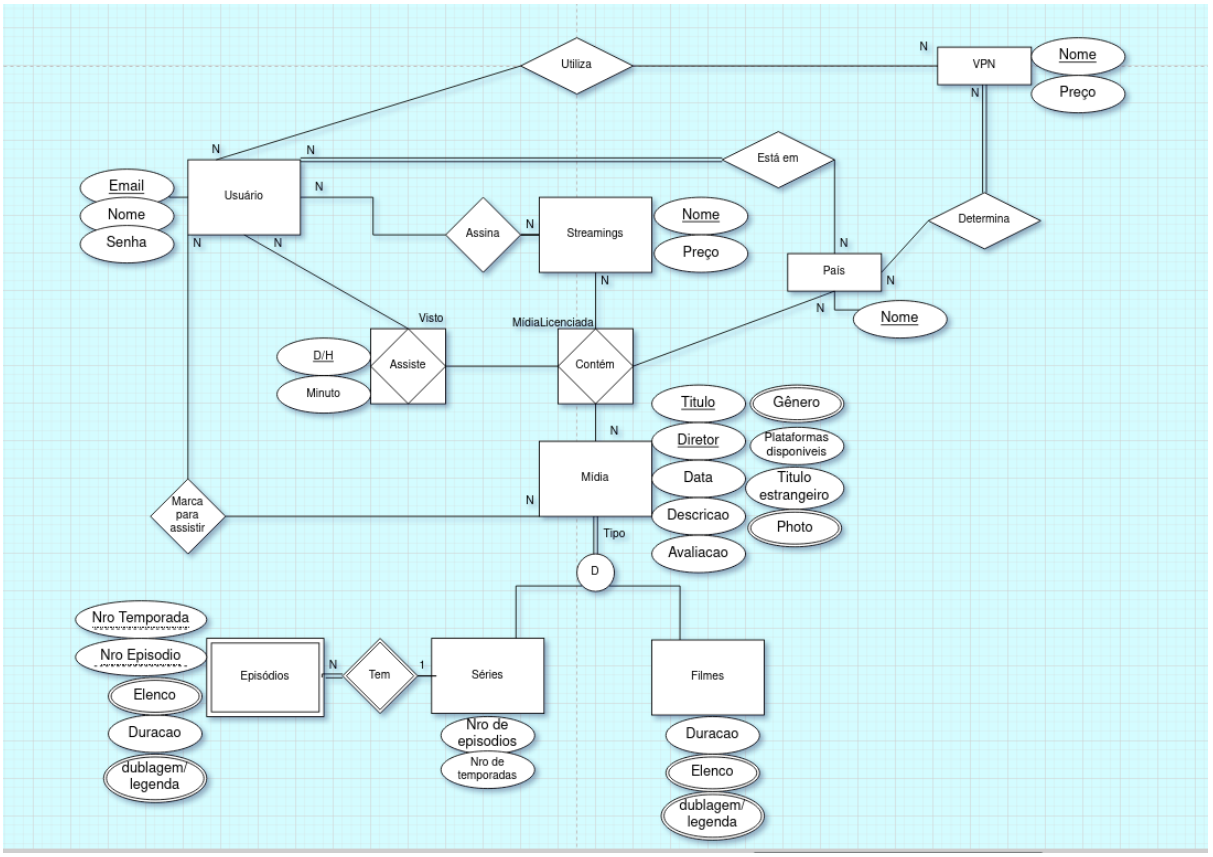
Nesta plataforma teremos os usuários, que podem assinar diferentes serviços de streaming e cadastrá-los no sistema. Cada streaming pode possuir mídias e essas mídias podem estar disponíveis em vários países e podem ser acessadas pelos streamings a partir da utilização (ou não) de VPNs, dependendo de quais países esse serviço de streaming disponibiliza essa mídia. Um mídia pode ser um filme ou uma série, sendo que as séries também possuem episódios associados a elas. Um usuário também pode ter assistido uma mídia em certa plataforma ou pode marcar uma mídia para assistir mais tarde.

Principais Operações

- Usuário
    - Criação, edição e exclusão de conta de usuário;
    - Consulta em qual plataforma o filme está disponível, sendo possível filtrar por VPN;
    - Consulta quais VPNs estão disponíveis, filtrando por valor e localidades disponíveis;
    - Avalia uma mídia assistida (série, filme ou episódio);
    - Consulta filmes, podendo filtrar por avaliação, gênero, data, elenco e direção;
    - Consulta filmes já assistidos;
    - Adiciona e remove que assina novas plataformas de streaming;
    - Adiciona ou remove que usa uma VPN;
    - Adiciona mídias para assistir mais tarde;

- Administrador
    - Inserir uma nova VPN, ou inserir que um país é acessível por uma VPN
    - Inserir um novo serviço de streaming
    - Inserir uma nova mídia
    - Inserir que uma mídia é acessível por um certo país num certo streaming e guardar a data disso

Projeto Conceitual



https://drive.google.com/file/d/1GBAzmTM3zC8knMR-0euB79ejP-L39OPV/view?usp=sharing

Pensamentos durante a modelagem

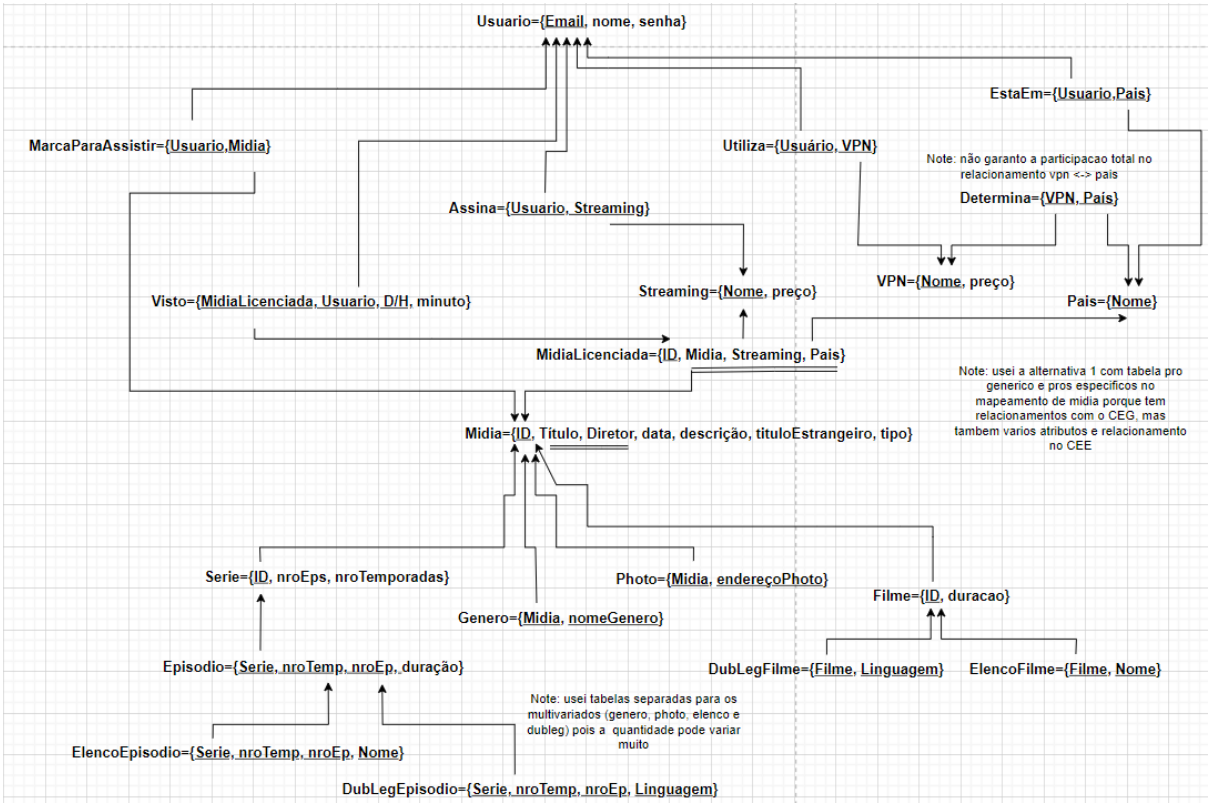
- Tem ciclo entre usuário, pais e vpn, mas não é problema pois o usuário pode acessar seu próprio país usando a vpn

- Pensamos em Mídia ser episodio e filme, para simplificar o modelo na hora de marcar um episódio como assistido, mas teríamos que guardar um título+temporada+episódio para cada episódio, deixando a chave muito longa e teríamos informações repetidas entre episodio e série. Então deixamos o usuário assistindo uma mídia (que eh um filme ou série), mas também guarda o minuto assistido, e com isso podemos guardar até qual episódio o usuário viu, usando a soma da duração dos episódios. (Pensamos também no

usuário marcar assistido diretamente o episódio, mas aí teríamos que dividir entre filme e episódio e criaríamos outro ciclo e precisaríamos de mais relações)

- Esse mapeamento gera um ciclo onde o usuário pode assistir uma mídia e marcar ela para ser assistida, mas no nosso projeto, isso não é um problema, pois consideramos que um usuário pode querer deixar como marcado para assistir algum filme ou série que ele já viu, caso queira ver outro episódio ou rever.

Modelo Relacional



[Link para a página do modelo relacional em alta escala]

Pensamentos durante a modelagem

- A participação total entre VPN, país não é garantida nesse modelo e deve ser tratada na próxima fase.

- Usamos uma tabela Mídia (o CEG) e uma tabela para Filme e uma para Série (os CEE) já que tem relacionamentos tanto com o CEG e com os CEEs que também tem vários atributos.

- Usamos tabelas separadas para os atributos multivalorados: Photo, Gênero, Dub/Leg e Elenco pois são todos atributos que podem variar muito entre um filme/episódio e outro e não tem um número fixo que a maioria vai ter.

Mudanças entre as entregas

- Na primeira entrega, havia a contextualização do projeto e dos requisitos de dados, as principais operações dos usuários e administradores e o projeto conceitual do projeto. No projeto conceitual havia quais seriam os conjuntos de entidades, os conjuntos e relacionamento.

- Na segunda entrega, adicionamos o modelo relacional do projeto e fizemos algumas pequenas alterações na descrição do problema e dos requisitos de dados.

- Para a terceira entrega, houve algumas mudanças no modelo relacional em relação ao VPN, a entidade mídia licenciada e a entidade visto.

Aplicação

Nossa aplicação foi implementada em Python com Postgre usando uma base de dados online com o ElephantSQL.

Todos os arquivos podem ser encontrados no link https://github.com/VictorLutes/TrabalhoBD.

Ela pode ser executada apenas rodando o comando “python3 main.py”.

Criamos dois arquivos com nosso esquema adaptado do modelo relacional, “Esquema.sql” usa a sintaxe da Oracle e “EsquemaPostgre.sql” com a sintaxe do Postgre e que é usado na nossa aplicação para recarregar o esquema.

Criamos também dois arquivos com a inserção de dados nas diferentes tabelas, “Dados.sql” usa a sintaxe da Oracle e “DadosPostgre.sql” com a sintaxe do Postgre e que é usado na nossa aplicação para recarregar os dados na base de dados, caso o usuário deseje remover tudo que inseriu.

As ações que podem ser realizadas na nossa aplicação são enumeradas de 0 a 7 quando o programa é executado:

0-Sair:

```
conn.close()
print('Database connection closed.')
```

1-apagar todas as tabelas e recarregar o esquema e os dados dos arquivos sql (usado se o usuário quiser apagar todas as inserções que foram feitas e recomeçar apenas com os dados no arquivo Dados.sql):

na função `dropTables` na linha 210:

```
cur.execute("SELECT table_schema,table_name FROM information_schema.tables WHERE table_schema = 'public' ORDER BY table_schema,table_name")
rows = cur.fetchall()
for row in rows:
    if(row[1]!="pg_stat_statements"):
        print("dropping table: ", row[1])
        cur.execute("drop table " + row[1] + " cascade")
conn.commit()
```

2-buscar pela plataforma com mais filmes ou séries marcadas para assistir de um usuário em um país:

na função `buscarMarcados` na linha 139:

```
print("Digite o email de um usuario para saber quantas midias que ele marcou para assistir tem em cada plataforma (se houver pelo menos uma) em um determinado pais")
email=input("digite um email: ")
pais=input("digite o pais: ")
```

```
#estou selecionando o nome da streaming e o numero de midias licenciadas que ela tem que contem alguma midia que o usuario passado marcou para assistir
sql="SELECT S.nome, count(MPA.midia) FROM Streaming S JOIN MidiaLicenciada ML ON S.nome=ML.streaming JOIN MarcaParaAssistir MPA ON MPA.midia=ML.midia WHERE MPA.usuario = %s AND ML.pais= %s GROUP BY S.nome HAVING count(MPA.midia)>0 ORDER BY count(MPA.midia) DESC;"
try:
   cur.execute(sql, [email, pais.upper()])
except(Exception, psycopg2.DatabaseError) as error:
    print(error)
    print("select failed")
    return
res = cur.fetchall()
print("Nome das streamings e numero de midias que elas tem em algum pais que o usuario "+email+" marcou para assitir: ")
for row in res:
    print("\t", end='')
    print(row)
```

3-buscar quais plataformas tem um filme ou série em um país:
na função `buscarMidia` na linha 158:

```
print("Descubra quais plataformas tem um filme ou serie em um determinado país")
pais=input("digite o pais: ")
nome=input("digite o nome: ")
diretor=input("digite o nome do diretor: ")
#estou selecionando o nome das Streamings que tem uma midia licenciada do filme com o nome e diretor passado pelo usuario no pais passado pelo usuario
sql="SELECT S.nome FROM Streaming S JOIN MidiaLicenciada ML ON S.nome=ML.streaming JOIN Midia M ON M.id=ML.midia WHERE ML.pais=%s AND M.titulo=%s AND M.diretor=%s;"
try:
   cur.execute(sql, [pais.upper(), nome.upper(), diretor.upper()])
except(Exception, psycopg2.DatabaseError) as error:
   print(error)
   print("select failed")
   return
res = cur.fetchall()
print("Nome das streamings com a midia "+nome+" no "+pais+": ")
```

```
for row in res:
   print("\t", end='(')
   print(row[0], end=')\n')
```

4-Listar todas as VPNs (comando que adicionamos para facilitar a verificação de que as inserções estavam sendo realizadas corretamente, mas é um select muito simples):

na função `buscarVPNs` na linha 178:

```
#estou selecionando todas as linhas da tabela VPN
sql="SELECT * FROM VPN;"
try:
    cur.execute(sql)
except(Exception, psycopg2.DatabaseError) as error:
    print(error)
    print("select failed")
    return
res = cur.fetchall()
print("Dados de VPN:")
for row in res:
    print("\t", end='')
    print(row)
```

5-Listar todas as Midias (comando que adicionamos para facilitar a verificação de que as inserções estavam sendo realizadas corretamente, mas é um select muito simples):

na função `buscarMidias` na linha 194:

```
#estou selecionando todas as linhas da tabela Midia
sql="SELECT * FROM Midia;"
try:
    cur.execute(sql)
except(Exception, psycopg2.DatabaseError) as error:
    print(error)
    print("select failed")
    return
res = cur.fetchall()
print("Dados de Midias:")
for row in res:
    print("\t", end='')
    print(row)
```

6-Inserir dados em uma tabela:

na função `inserir` na linha 28:

O usuário pode escolher entre 5 tabelas para inserir dados (e se ele quiser inserir dados em mídia tem que haver uma inserção casada em filme ou série e é feito rollback caso isso não ocorra). Para todas as inserções é feito um rollback se o comando SQL retorna erro, se não, é feito commit.

```
def inserir():
    print("Tabelas: 1-VPN, 2-Streaming, 3-Pais, 4-Usuario, 5-Midia (e Filme ou Serie)")
    tabela=int(input("digite o numero da tabela que voce quer inserir: "))
   if(tabela==1):#insere em VPN
       nome=input("digite o nome da VPN: ")
       preco=int(input("digite o preco da VPN: "))
       sql="INSERT INTO VPN (nome, preco) VALUES (%s, %s);"
       try:
           cur.execute(sql, [nome.upper(), preco])
       except(Exception, psycopg2.DatabaseError) as error:
           print(error)
           print("insert failed")
           conn.rollback()
           return
       conn.commit()

   elif(tabela==2):#insere em Streaming
       nome=input("digite o nome da Plataforma de Streaming: ")
       preco=input("digite o preco da Plataforma de Streaming: ")
       sql="INSERT INTO Streaming (nome, preco) VALUES (%s, %s);"
       try:
           cur.execute(sql, [nome.upper(), preco])
       except(Exception, psycopg2.DatabaseError) as error:
           print(error)
           print("insert failed")
           conn.rollback()
           return
       conn.commit()

   elif(tabela==3):#insere em pais
```

```
        nome=input("digite o nome do pais: ")
        sql="INSERT INTO Pais (nome) VALUES (%s);"
        try:
            cur.execute(sql, [nome.upper()])
        except(Exception, psycopg2.DatabaseError) as error:
            print(error)
            print("insert failed")
            conn.rollback()
            return
        conn.commit()

    elif(tabela==4):#insere em usuario
        email=input("digite o email do Usuario: ")
        nome=input("digite o nome do Usuario: ")
        senha=input("digite o senha do Usuario: ")
          sql="INSERT INTO Usuario (email, nome, senha) VALUES (%s, %s,
%s);"
        try:
            cur.execute(sql, [email, nome, senha])
        except(Exception, psycopg2.DatabaseError) as error:
            print(error)
            print("select failed")
            conn.rollback()
            return
        conn.commit()

    elif(tabela==5):#faz insercao casada midia e filme ou serie
          print("Digite os dados da mídia e depois os dados do filme ou
serie correspondente")
        titulo=input("digite o titulo da Midia: ")
        diretor=input("digite o diretor da Midia: ")
            data=input("digite a data da Midia (no formato YYYY/MM/DD
HH24:MI:SS): ")
        descricao=input("digite a descricao da Midia: ")
        tituloEstrangeiro=input("digite o tituloEstrangeiro da Midia: ")
        tipo=input("digite o tipo da Midia (filme ou serie): ")
            sql="INSERT INTO Midia (titulo, diretor, data, descricao,
tituloEstrangeiro, tipo) VALUES (%s, %s, TO_DATE(%s, 'YYYY/MM/DD
HH24:MI:SS'), %s, %s, %s);"
        try:
                cur.execute(sql, [titulo.upper(), diretor.upper(), data,
descricao, tituloEstrangeiro.upper(), tipo.upper()])
```

```
        except(Exception, psycopg2.DatabaseError) as error:
            print(error)
            print("insert failed")
            conn.rollback()
            return
          #agora vou buscar o id da linha que acabei de inserir para
inserir com o mesmo id a especializacao serie ou filme
        sql="SELECT id FROM Midia WHERE titulo=%s AND diretor=%s;"
        try:
            cur.execute(sql, [titulo.upper(), diretor.upper()])
        except(Exception, psycopg2.DatabaseError) as error:
            print(error)
            print("insert failed")
            conn.rollback()
            return
        res = cur.fetchall()
        ident=res[0][0]

        if(tipo.upper()=='SERIE'):#insere com o mesmo id em serie
            print("Agora insira os dados dessa Serie:")
            nroTemp=input("digite numero de temporadas da  Serie: ")
                 nroEp=input("digite numero episodios por temporada da
Serie: ")
              sql="INSERT INTO Serie (id, nroTemporadas, nroEps) VALUES
(%s, %s, %s);"
            try:
                cur.execute(sql, [int(ident), int(nroTemp), int(nroEp)])
            except(Exception, psycopg2.DatabaseError) as error:
                print(error)
                print("insert failed")
                conn.rollback()
                return
        else:#insere com o mesmo id em filme
            print("Agora insira os dados desse Filme:")
            dur=input("digite a duracao do Filme: ")
            sql="INSERT INTO Filme (id, duracao) VALUES (%s, %s);"
            try:
                cur.execute(sql, [int(ident), int(dur)])
            except(Exception, psycopg2.DatabaseError) as error:
                print(error)
                print("insert failed")
                conn.rollback()
```

```
                return

        conn.commit()
```

Buscas e Esquema

No arquivo Esquema.sql estão os comandos de criação de tabela. Foi modificado em EsquemaPostgre.sql para funcionar no Postgre, apenas com Number trocado por Integer e Varchar2 por Varchar.

As Tabelas criadas são: Assina, Determina, DubLegEpisodio, DubLegFilme, ElencoEpisodio, ElencoFilme, Episodio, EstaEm, Filme, Genero, MarcaParaAssistir, Midia, MidiaLicenciada, Pais, Photo, Serie, Streaming, Usuario, Utiliza, Visto e VPN.

No arquivo Buscas.sql apresentamos os diferentes selects que implementamos para o trabalho (dois deles foram usados na aplicação):

Retorna todos os nomes e diretores das mídias que têm algum gênero específico

```
/*Filtrar por genero:*/
SELECT M.titulo, M.diretor FROM
    Midia M JOIN Genero G ON G.midia=M.id WHERE G.nomeGenero='genero
buscado';
```

Retorna todos os nomes e diretores dos Filmes ou Séries que estão dublados ou legendados em alguma linguagem específica:

```
/*filtrar por linguagem:*/
SELECT M.titulo, M.diretor FROM
        Midia M JOIN DubLegEpisodio D ON D.serie=M.id WHERE
D.linguagem='linguagem buscada'
   UNION
SELECT M.titulo, M.diretor FROM
         Midia M JOIN DubLegFilme D ON D.filme=M.id WHERE
D.linguagem='linguagem buscada';
```

Retorna todos os nomes e diretores dos Filmes ou Séries que tem um ator específico no elenco:

```
/*buscar filmes e series com um ator:*/
SELECT M.titulo, M.diretor FROM
    Midia M JOIN ElencoEpisodio E ON E.serie=M.id WHERE E.nome='ator
buscado'
   UNION
SELECT M.titulo, M.diretor FROM
```

```
    Midia M JOIN ElencoFilme E ON E.filme=M.id WHERE E.nome='ator
buscado';
```

Retorna todos os nomes das plataformas e o numero de midias que elas têm, em ordem das plataformas que licenciam o maior número de mídias em um país específico que um usuário específico marcou para assistir:

```
/*Mostrar as plataformas com mais shows que ele marcou para assistir em
um pais:*/
SELECT S.nome, count(MPA.midia) FROM Streaming S
   JOIN MidiaLicenciada ML ON S.nome=ML.streaming
   JOIN MarcaParaAssistir MPA ON MPA.midia=ML.midia
   WHERE MPA.usuario='email buscado' AND ML.pais='pais buscado'
   GROUP BY S.nome HAVING count(MPA.midia)>0
   ORDER BY count(MPA.midia) DESC;
```

Retorna todos os nomes das plataformas que tem um filme ou série especifico em um país específico.

```
/*selecionar todas as plataformas que tem um filme ou serie num
determinado pais:*/
SELECT S.nome FROM Streaming S
   JOIN MidiaLicenciada ML ON S.nome=ML.streaming
   JOIN Midia M ON M.id=ML.midia
     WHERE ML.pais='pais buscado' AND M.titulo='titulo buscado' AND
M.diretor='diretor buscado';
```

Retorna todos os nomes das plataformas que tem um filme ou série especifico, buscando pelo seu título em portugues.

```
/*selecionar streaming que possui a mídia buscada pelo usuário:*/
SELECT str.nome, ML.pais FROM Streaming str
   INNER JOIN MidiaLicenciada ML ON str.nome = ML.streaming
   INNER JOIN Midia M ON M.id = ML.midia
   WHERE M.tituloEstrangeiro = 'tituloBuscado';
```

Retorna todos os usuarios e o numero de midias que assistiu em ordem decrescente

```
/*Mostrar os usuários com mais shows assistidos:*/
SELECT usuario, COUNT(midiaLicenciada) as nroMidia
   FROM Visto
   GROUP BY usuario
```

```
   ORDER BY nroMidia DESC;
```

Retorna todas as plataformas que licenciam em qualquer país todos os filmes que algum usuário marcou para assistir.

```
/*Mostrar as plataformas todos os shows que um usuario marcou para assistir:*/
/*se for testar no Postgre usar EXCEPT ao inves de MINUS*/
SELECT S.nome FROM Streaming S
   WHERE NOT EXISTS(\
               (SELECT MPA.midia FROM MarcaParaAssistir MPA WHERE MPA.usuario='')
       MINUS
            (SELECT ML.midia FROM MidiaLicenciada ML WHERE S.nome = ML.streaming));
```

Busa qual usuario mais assistiu algum gênero específico

```
-- Buscar qual usuario mais assistiu algum genero
SELECT usuario, COUNT(midiaLicenciada) as nroMidia
    FROM Visto
    JOIN Genero G ON G.midia=Visto.midiaLicenciada on G.nomeGenero='genero buscado'
    GROUP BY usuario ON
    ORDER BY nroMidia DESC
    LIMIT 1;
```

Busca qual usuário mais assistiu alguma plataforma

```
-- Busca qual usuario mais assistiu alguma plataforma
SELECT usuario, COUNT(midiaLicenciada) as nroMidia
    FROM Visto
    JOIN Streaming S ON S.nome=Visto.midiaLicenciada on S.nome='plataforma buscada'
    GROUP BY usuario ON
    ORDER BY nroMidia DESC
    LIMIT 1;
```

Conclusão

O projeto foi uma maneira muito legal de unir todas as partes que estudamos na matéria (modelo conceitual, modelo relacional e SQL). Também foi interessante criar nossa ideia de projeto do zero e tentar implementar ela com o que vimos em aula. Uma dificuldade que tivemos foi de tentar conectar nossa aplicação em python com o Oracle, que acabamos não conseguindo fazer, mas com Postgre o processo foi mais simples. Fora esse problema a tradução do esquema relacional para o esquema SQL não foi tão trabalhoso e foi interessante pensar em diferentes buscas que poderíamos fazer nos dados e depois implementar elas com o select.

Esse projeto ajudou muito a fixar o conteúdo da disciplina e foi muito engajante a oportunidade de implementar nossa própria ideia de projeto.