Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Bacharelado em Ciências de Computação - 2022.1
SCC0240 - Bases de Dados

Open Media Database (OMDB)

Sistema colaborativo para catalogação e avaliação de mídias em geral

Docente: Profª Elaine Parros Machado de Sousa
Estagiário PAE: André Moreira Souza

Eduardo Rodrigues Amaral - 11735021
Lucas Xavier Leite - 10783347
Vinícius Stroebel Bertelle - 9266469

São Carlos, 03 de julho de 2022

1. Descrição do problema e dos requisitos de dados

Com o avanço da tecnologia e a facilidade de distribuição de conteúdo em meios digitais, o número de mídias cresce em um ritmo cada vez mais acelerado, criando também a necessidade de catalogação desses conteúdos, com informações relevantes a respeito do conteúdo, participantes e organizações envolvidas. Com este propósito, hoje existem várias plataformas disponíveis na internet, como IMDB e IGN , contendo várias informações a respeito de mídias específicas, assim como plataformas que permitem a avaliação de mídias e interação com outros usuários através de comentários, como MyAnimeList e MyDramaList. No caso destes dois últimos, as informações ainda são editadas de forma colaborativa pelos usuários, de forma similar ao Wikipedia.

As plataformas citadas, contudo, reúnem em sua base de dados informações específicas para cada tipo de mídia, como filmes, jogos, anime (séries ou filmes de animação japonesa), ou dramas (como são popularmente conhecidas as séries de conteúdo asiático), exigindo que os usuários utilizem diversas plataformas diferentes.

Tendo isso em mente, o objetivo desta plataforma é ser uma base de dados centralizada e colaborativa, assim como um sistema de avaliação, seguindo o modelo das plataformas MyAnimeList e MyDramaList, mas com o propósito de reunir as informações de todos os tipos de mídia (filmes, séries, jogos, audiobooks, etc.) em uma única plataforma.

No sistema proposto existem dois tipos de usuários**: comum** e **moderador.** O usuário comum é aquele que irá buscar conteúdo, interagir e colaborar na edição das informações das páginas, enquanto que os moderadores são os responsáveis por moderar esse conteúdo, podendo reverter as ações indevidas e punir os usuários responsáveis, de forma a manter a ordem na plataforma.

Um **usuário comum** pode buscar uma **mídia** no catálogo, fazendo uso de filtros, como mais bem avaliados, gêneros, número de avaliações, etc., assim como cadastrar uma nova página de mídia contendo as informações de cada **participante,** como a natureza da participação: **autor, ator e diretor**; assim como suas indicações e uma descrição, podendo, por exemplo, indicar se o ator tem papel de protagonista ou coadjuvante, ou ainda se o autor é roteirista de cinema ou desenvolvedor de jogo. Também pode informar **sinopse**, **data de lançamento**,

gêneros, franquia, organizações envolvidas: produtora, distribuidora, gravadora e **desenvolvedora;** e **status: em produção, em exibição (específico para séries), finalizado** e **cancelado**. Ele pode especificar um tipo de mídia como: **filme**, **série**, **jogo** e **audiobook**, ou ainda criar um tipo **genérico**. No caso de **séries** (incluindo shows de variedades, etc.), pode incluir informações dos **episódios** e das **temporadas,** enquanto que, para **jogos**, pode indicar a **plataforma**, e para **audiobooks**, o **número de capítulos**. A **duração** pode ser especificada tanto para audiobooks quanto para **filmes**. Para tipos genéricos, esses atributos especiais não estão disponíveis. A página de mídia criada pelo usuário só poderá ser visualizada pelos outros usuários após aprovação do moderador.

O usuário pode avaliar uma mídia, com uma **nota** e um **texto**, uma única vez. Ele também pode fazer **comentários** na página da mídia, assim como comentários em resposta a outros comentários, e também reagir a eles com **votos** positivos ou negativos.

Os usuários também podem adicionar mídias a uma lista de **interesses**.

Um **moderador**, além de poder realizar todas as operações de usuário comum, também deve **aprovar** a criação de uma nova página, para evitar possíveis abusos, e também podem **remover** ou **editar** dados cadastrados pelos usuários que julgarem inapropriados, como comentários, avaliações e páginas de mídia, (no caso deste último item, também pode reverter para uma versão antiga do conteúdo), assim como **penalizar** os usuários responsáveis.

Um usuário pode iniciar uma **promoção**, solicitando uma votação para ser promovido de usuário comum a moderador, que será encerrada após um determinado período de tempo. Se durante o período da promoção, o número de votos for suficiente, seu **status** passa a ser **deferido**, efetivando a promoção. Caso o usuário não alcance o número necessário de votos, seu status assume o valor **indeferido**, porém o usuário pode participar de uma nova promoção após um certo período de tempo, reiniciando todo o processo. O status de moderador só pode ser revertido a usuário comum pelo administrador do sistema.

2. Principais operações

- Usuário comum
  - buscar mídias no catálogo, com uso de filtros

- criar páginas de mídia, que devem ser aprovadas por um moderador
- editar mídias de qualquer usuário
- avaliar mídia, com nota e texto
- comentar em mídias
- responder comentários de outros usuários
- reagir a comentários de outros usuários
- adicionar e remover mídias da lista de interesses
- solicitar uma votação para ser promovido a moderador

- Moderador
  - todas as operações de usuário comum
  - aprovar a criação de uma nova página
  - remover e editar avaliações de qualquer usuário
  - remover e editar comentários de qualquer usuário
  - aplicar uma penalidade ao usuário, após moderação
  - votar na promoção para moderador de qualquer usuário
  - criar, editar e remover páginas de pessoas e organizações

3. Possíveis Problemas/Inconsistências do MER

3.1. Relacionamento ternário “Promoção”, “Usuário”, “Moderador”

Tal relacionamento ternário poderia ser quebrado em dois binários (“Usuário requer Promoção” e “Moderador vota em Promoção”) sem perda de informação semântica. Foi escolhido manter o relacionamento ternário, pois, por mais que entidade fraca “Promoção” seja identificada apenas pela data e pelo usuário que a requereu, o grupo entende que o relacionamento ternário deixa mais claro a interação indireta entre o moderador e o usuário durante a votação. Tal relacionamento ternário gera o ciclo “Usuário” -> “Promoção” -> “Moderador (Usuário)”, porém tal ciclo não é preocupante para o modelo físico pois existe a restrição de que apenas usuários do tipo “usuário” podem solicitar promoções e apenas usuários do tipo

"moderador" podem votar em promoções, o que impossibilita que um usuário vote em sua própria promoção.

3.2. Ciclos de Moderação

Todos relacionamentos de natureza: "Moderador (Usuário)" modera "Algo" (seja "Edição", "Comentário", etc) gera ciclos. Tais ciclos são essenciais para refletir que o relacionamento de moderação gera a entidade agregada "Mod_Algo". Outro motivo que impossibilita a quebra do ciclo é o fato de todas entidades "Algo" serem agregações de relacionamentos de "Usuário" e "Mídia". Dessa maneira o "Usuário" tem de estar conectado às entidades "Algo" realizando a agregação e os "Moderadores (Usuários)" tem de estar conectados às entidades "Algo" às moderando e gerando as agregações "Mod_Algo". Para evitar injustiças de um moderador moderar suas próprias ações basta evitar que "Mod_Algo" seja criado com email do moderador igual ao email do criador de "Algo".

3.3. Inconsistência de data entre "Pessoa" e "Mídia"

Para evitar que uma "Pessoa" com data de nascimento posterior ao ano de lançamento de uma mídia participe de tal "Mídia" alguma verificação deve ser feita, seja ela garantida pelo esquema do banco (por meio de *Checks* ou *Triggers*) ou pela aplicação.

3.4. Inconsistência de data entre "Organização" e "Mídia"

Para evitar que uma "Organização" com data de fundação posterior ao ano de lançamento de uma mídia produza tal "Mídia" alguma verificação deve ser feita, seja ela garantida pelo esquema do banco (por meio de *Checks* ou *Triggers*) ou pela aplicação.

3.5. Inconsistência de data entre "Mod_Algo" e "Algo"

Para evitar que uma moderação ("Mod_Algo") seja feita com data prévia à data de criação de "Algo" (tal como "Comentário", "Edição") alguma verificação deve ser feita, seja

ela garantida pelo esquema do banco (por meio de *Checks* ou *Trigger's*) ou pela aplicação.

3.6. Inconsistência de data entre "Edição" e "Cadastro"

Para evitar que uma "Edição" seja feita com data prévia à data de "Cadastro" da "Mídia" editada, alguma verificação deve ser feita, seja ela garantida pelo esquema do banco (por meio de *Checks* ou *Triggers*) ou pela aplicação.

3.7. Inconsistência de data entre "Comentário" e "Cadastro"

Para evitar que um "Comentário" seja feito com data prévia à data de "Cadastro" da "Mídia" comentada alguma verificação deve ser feita, seja ela garantida pelo esquema do banco (por meio de *Checks* ou *Triggers*) ou pela aplicação.

3.8. Inconsistência de data entre "Comentário" e "Comentário"

Para evitar que um "Comentário" seja comentado por outro "Comentário" de data prévia a ele, alguma verificação deve ser feita, seja ela garantida pelo esquema do banco (por meio de *Checks* ou *Triggers*) ou pela aplicação.

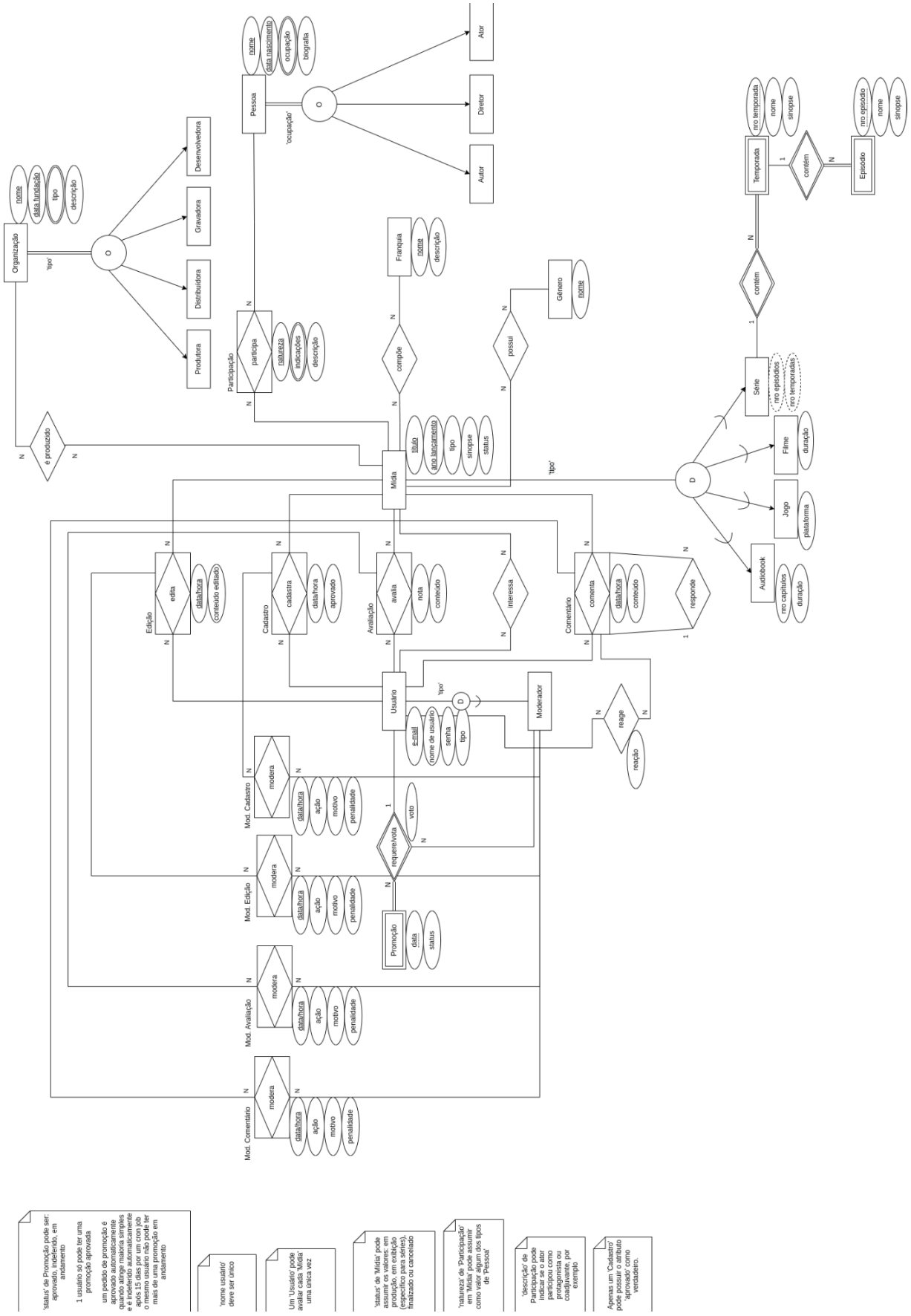
3.9. "Mídia" sem "Cadastro" aprovado, sem "Autor (Pessoa)" ou "Organização" que produziu

Por mais que pareça um problema existir uma "Mídia" sem "Cadastro" referente a ela aprovado, ou sem "Autor (Pessoa)", ou sem "Organização" que produziu, tal situação é válida para "Mídias fantasmas" que relacionam-se com outras entidades. Tal situação é comum em sistemas do tipo "Wiki" e é comumente chamada de *Red Links*.

3.10. Participação inválida de "Pessoa" em "Mídia"

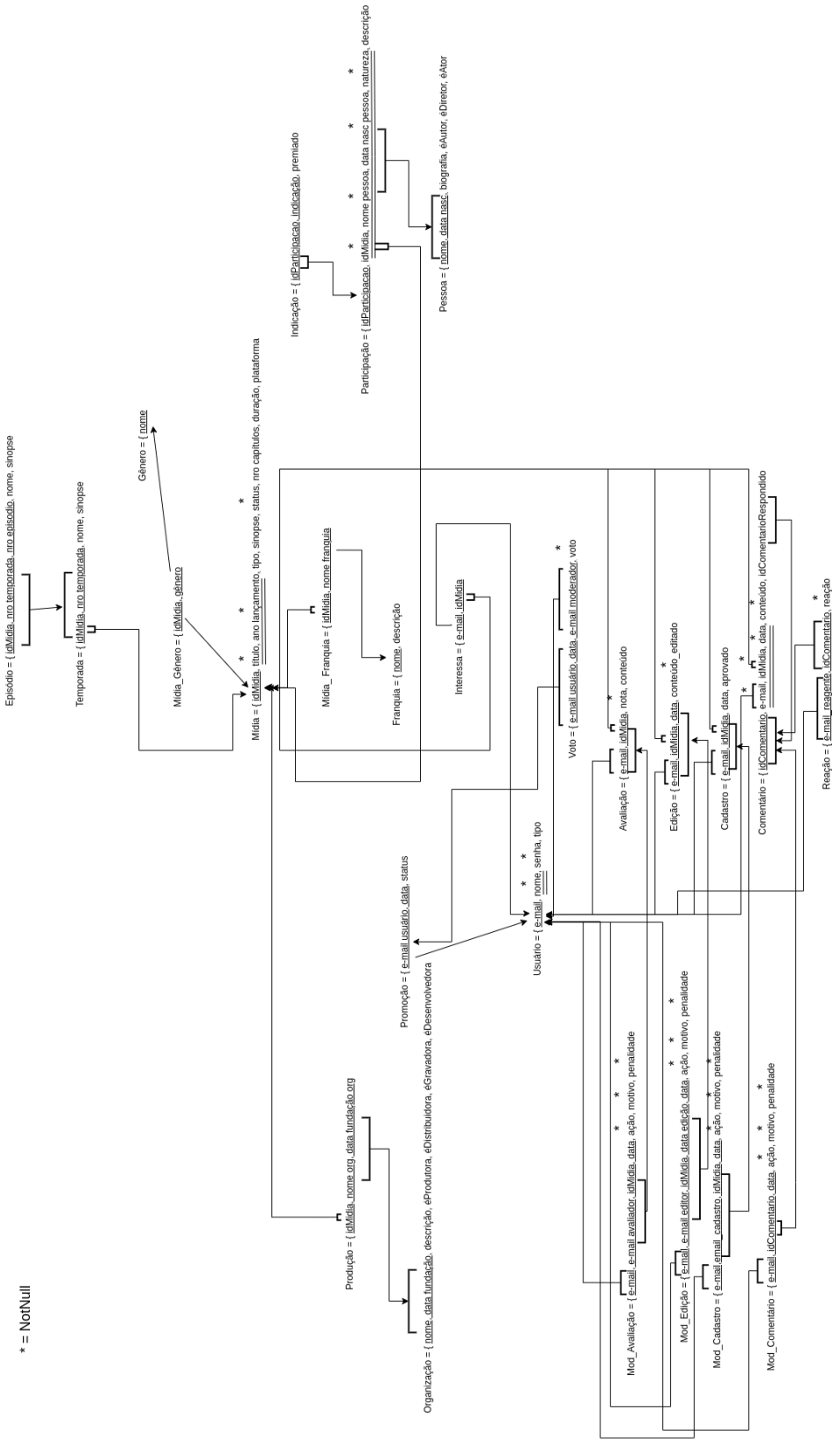
Para evitar que uma "Pessoa" não participe em uma mídia com uma natureza de especialização que não possui (pessoa atua sem ser "Ator") alguma verificação deve ser feita, seja ela garantida pelo esquema do banco (por meio de *Checks* ou *Trigger's*) ou pela aplicação.

4. Projeto conceitual



5. Projeto lógico

5.1. Modelo relacional



5.2. Discussões e justificativas

5.2.1. Generalização de Organização e Pessoa

Em função do pequeno número de atributos, relacionamentos e especializações optamos por mapear as especializações de Organização — Produtora, Distribuidora, Gravadora e Desenvolvedora — e Pessoa — Autor, Ator, Diretor — em uma única tabela para cada entidade com atributos booleanos identificando as generalizações. Tal mapeamento permite refletir a sobreposição de especializações facilmente e permite garantir a restrição de totalidade facilmente por meio de *Checks* no modelo físico.

5.2.2. Generalização de Mídia

As entidades especializadas de Mídia — Filme, Série, Audiobook e Jogo — não possuem muitos atributos ou relacionamentos, tornando o mapeamento em uma única tabela uma opção viável, com a vantagem de agilizar as consultas à base de dados, uma das principais operações que serão realizadas na aplicação.

Para o relacionamento da entidade Série com Temporada e Episódios, é possível realizar o uso de *constraints* na implementação em SGBD para verificar se o tipo de mídia é uma série.

5.2.3. Agregações de Avaliação e Cadastro

As agregações das entidades Avaliação e Cadastro ocorrem apenas uma vez para cada par Usuário/Mídia, portanto não é necessário o uso de uma chave adicional.

5.2.4. Atributo Multivalorado Indicações

O atributo multivalorado “indicações” foi mapeado para uma tabela auxiliar pois além de uma participação poder possuir N indicações, as indicações também possuem o atributo “premiado” que informa se a indicação se concretizou em uma premiação de fato.

5.2.5. Atributos Multivalorados Tipo e Ocupação

Os atributos “tipo” (de “Organização) e “ocupação” (de “Pessoa”) foram mapeados por meio de atributos booleanos em suas respectivas entidades, pois, além de possuírem um número pequeno e específico de valores fazem parte de generalizações totais sobrepostas e tal mapeamento facilita refletir as generalizações.

5.2.6. Atributos Derivados

Os atributos derivados “número de episódios” e “número de temporadas” não foram mapeados para atributos físicos e mantiveram-se como atributos calculados, pois são advindos de relacionamentos e facilmente calculados. Torná-los físicos introduziria uma grande chance de inconsistência e não traria benefícios consideráveis em questão de performance.

5.3. Observações

É importante notar que, devido à presença de chave composta por muitos atributos, um identificador sintético na entidade “Participação” foi utilizado visando melhoria de performance. Outra entidade que foi identificada sinteticamente por motivos relacionados à performance foi a entidade “Mídia” que possui diversos relacionamentos que seriam afetados pela presença da chave natural composta por um número inteiro e por uma *string* de tamanho variável. Por fim, a entidade “Comentário” também recebeu um identificador sintético devido à possibilidade de um alto número de relacionamentos nas auto-relações de resposta.

6. Descrição da implementação

A aplicação que implementa o projeto OMDB foi implementada em Python (3.10.5) e o SGBD utilizado foi o PostgreSQL (14.3). A biblioteca utilizada para o conector foi a Psycopg2. Outras bibliotecas como prettytable (para exibição do resultado das consultas) e python-dotenv (para leitura do arquvo de credenciais) foram utilizadas também. Para utilizar a aplicação modifique o arquivo *.env* (presente na raiz do projeto) com as credenciais de acesso ao banco de dados, instrale as dependências executando o comando *pip install -r requirements.txt* e execute o arquivo *main.py* no interpretador Python.

6.1. Trechos do código da aplicação

6.1.1. Inserção de nova mídia no banco de dados

A rotina *insertMedia* presente no arquivo *media.py* é responsável pela insersão de novas mídias.

```
def insertMedia(cursor, connection):
    clearScreen()
    print("Inserindo nova mídia...")
    platform = duration = chapterCount = None
    title = readString("Título: ", 255)
    year = readMediaReleaseDate("Data de Lançamento: ")
    mediaType = readMediaType("Tipo de Mídia: ")
    sinopsis = readString("Sinopse: ")
    status = readMediaStatus("Status: ", mediaType)
    if mediaType == 'JOGO':
        platform = readString("Plataforma: ", 255)
    elif mediaType == 'FILME':
        duration = readMediaDuration("Duração da mídia:
")
    elif mediaType == 'AUDIOBOOK':
        duration = readMediaDuration("Duração da mídia:
")
        chapterCount = readInt("Número de capítulos: ")

    print("Inserindo nova mídia...")
    try:
        cursor.execute("INSERT INTO Midia(titulo,
data_lancamento, tipo, sinopse, status, nro_capitulos,
duracao, plataforma) VALUES (%s, %s, %s, %s, %s, %s,
%s, %s) RETURNING id_midia",
```

```
                    (title, year, mediaType,
sinopsis, status, chapterCount, duration, platform))
        connection.commit()
    except psycopg2.errors.UniqueViolation:
        connection.rollback()
        print("Mídia já existe!")
        return
    except Exception as e:
        print(e)
        print("Erro ao inserir nova mídia!")
        return False

    insertedId = cursor.fetchone()[0]

    print("Mídia inserida com sucesso!")

    while readYesOrNo("Você deseja inserir um gênero
para a mídia? (S/N) "):
        if insertMediaGenero(cursor, connection,
insertedId) == False:
            return False

    while readYesOrNo("Você deseja inserir uma franquia
para a mídia? (S/N) "):
        if insertMidiaFranchise(cursor, connection,
insertedId) == False:
            return False

    if mediaType == 'SERIE':
        while readYesOrNo("Você deseja inserir uma
temporada para a mídia? (S/N) "):
            season = insertSerieSeason(cursor,
connection, insertedId)
            if season == False:
                return False
            while readYesOrNo(f"Você deseja inserir um
episódio para a temporada {season}? (S/N) "):
                if insertSerieEpisode(
                        cursor, connection, insertedId,
season) == False:
                    return False
```

As rotinas *insertMediaGenero, insertMediaFranchise, insertSerieSeason* e *insertSerieEpisode*, presentes no arquivo *media.py* são responsáveis por inserir possíveis gêneros, fraquias, temporadas e episódios da nova mídia.

```
def insertMediaGenero(cursor, connection, mediaId):
    genero = readString("Gênero: ", 50)
    try:
        cursor.execute("SELECT * FROM Genero WHERE nome
= %s", (genero,))
        if cursor.rowcount == 0:
            cursor.execute("INSERT INTO Genero(nome)
VALUES (%s)", (genero,))
        cursor.execute(
            "INSERT INTO Midia_Genero(id_midia, genero)
VALUES (%s, %s)", (mediaId, genero))
        connection.commit()
    except psycopg2.errors.UniqueViolation:
        connection.rollback()
        print("Gênero já existe para mídia!")
    except Exception as e:
        print(e)
        print("Erro ao inserir gênero!")
        return False


def insertMediaFranchise(cursor, connection, mediaId):
    franchise = readString("Nome da Franquia: ", 255)
    try:
        cursor.execute("SELECT * FROM Franquia WHERE
nome = %s", (franchise,))
        if cursor.rowcount == 0:
            description = readString("Descrição da
franquia: ")
            cursor.execute("INSERT INTO Franquia(nome,
descricao) VALUES (%s, %s)",
                           (franchise, description))
        cursor.execute(
            "INSERT INTO Midia_Franquia(id_midia, nome)
VALUES (%s, %s)", (mediaId, franchise))
        connection.commit()
    except psycopg2.errors.UniqueViolation:
```

```
        connection.rollback()
        print("Franquia já existe para mídia!")
    except Exception as e:
        print(e)
        print("Erro ao inserir franquia!")
        return False


def insertSerieSeason(cursor, connection, mediaId):
    season = readInt("Temporada: ")
    name = readString("Nome da temporada: ", 255)
    sinopsis = readString("Sinopse da temporada: ")
    try:
        cursor.execute("INSERT INTO Temporada(id_midia,
nro_temporada, nome, sinopse) VALUES (%s, %s, %s,
%s)",
                       (mediaId, season, name,
sinopsis))
        connection.commit()
        return season
    except psycopg2.errors.UniqueViolation:
        connection.rollback()
        print("Temporada já existe para mídia!")
        return season
    except Exception as e:
        print(e)
        print("Erro ao inserir temporada!")
        return False


def insertSerieEpisode(cursor, connection, mediaId,
season):
    episode = readInt("Episódio: ")
    name = readString("Nome do episódio: ", 255)
    sinopsis = readString("Sinopse do episódio: ")
    try:
        cursor.execute("INSERT INTO Episodio(id_midia,
nro_temporada, nro_episodio, nome, sinopse) VALUES
(%s, %s, %s, %s, %s)",
                       (mediaId, season, episode, name,
sinopsis))
        connection.commit()
    except psycopg2.errors.UniqueViolation:
```

```
        connection.rollback()
        print("Episódio já existe para mídia!")
    except Exception as e:
        print(e)
        print("Erro ao inserir episódio!")
        return False
```

6.1.2. Recuperação de todas as mídias presentes no banco de dados

A rotina *listMedia* presente no arquivo *media.py* é responsável pela recuperação e por mostrar todas as mídias presentes na base de dados

```
def listMedia(cursor):
    clearScreen()
    cursor.execute(
        'SELECT M.titulo as "Título",
M.data_lancamento as "Ano de Lançamento", M.tipo
as "Tipo", SUBSTRING(M.sinopse,1,20) as
"Sinopse", M.status as "Status",
COALESCE(CAST(M.nro_capitulos as VARCHAR),
\'-\') as "Número de Capítulos",
COALESCE(CAST(M.duracao as VARCHAR), \'-\') as
"Duração", COALESCE(M.plataforma, \'-\') as
"Plataforma", CASE WHEN UPPER(M.tipo) =
\'SERIE\' THEN CAST(COALESCE(T.nro_temporadas,0)
AS VARCHAR) ELSE \'-\' END as "Número de
Temporadas", CASE WHEN UPPER(M.tipo) = \'SERIE\'
THEN CAST(COALESCE(E.nro_episodio,0) AS VARCHAR)
ELSE \'-\' END as "Número de Episódios"  FROM
Midia as M LEFT JOIN (SELECT id_midia, COUNT(*)
as nro_temporadas FROM Temporada GROUP BY
id_midia) as T ON M.id_midia = T.id_midia LEFT
JOIN (SELECT id_midia, COUNT(*) as nro_episodio
FROM Episodio GROUP BY id_midia) as E ON
M.id_midia = E.id_midia ORDER BY M.titulo')
    print("Mídias cadastradas:")
    table = from_db_cursor(cursor)
    print(table)
    input("Pressione ENTER para fechar...")
```

6.1.3. Recuperação de todos atores, e suas possíveis premiações, de um dado filme

A rotina *listActorFromMovie* presente no arquivo *media.py* é responsável pela recuperação e por mostrar todos atores, e suas possíveis premiações, de um dado filme

```
def listActorFromMovie(cursor):
   clearScreen()
   movieName = readString("Título do filme: ",
255)
   try:
       cursor.execute(
           "SELECT id_midia FROM Midia WHERE
titulo = %s AND UPPER(tipo) = 'FILME' ",
(movieName,))
       if cursor.rowcount == 0:
           print(f"Filme {movieName} não
encontrado!")
           return
       cursor.execute(
           'SELECT Participacao.nome_pessoa as
"Nome",
TO_CHAR(Participacao.data_nasc_pessoa,\'DD/MM/YY
YY\') as "Data de Nascimento",
COALESCE(Indicacao.indicacao,\'-\') as
"Indicação", CASE WHEN Indicacao.premiado = TRUE
THEN \'Sim\' WHEN Indicacao.premiado = FALSE
THEN \'Não\' ELSE \'-\' END as "Premiado" FROM
(SELECT id_midia FROM Midia WHERE Midia.titulo =
%s) M JOIN Participacao ON (M.id_midia =
Participacao.id_midia AND Participacao.natureza
= \'ATOR\') LEFT JOIN Indicacao ON
(Participacao.id_participacao =
Indicacao.id_participacao) ORDER BY
Participacao.nome_pessoa', (movieName,))
       print(f"Atores do Filme {movieName}:")
       table = from_db_cursor(cursor)
       print(table)
       input("Pressione ENTER para fechar...")
   except Exception as e:
```

```
        print(e)
        print(f"Erro ao buscar atores do filme
{movieName}!")
        return False
```

7. Principais Consultas

7.1.

Consultar todas as mídias que possuem participação de atores específicos e sejam feitas por uma organização específica e sejam lançadas em um período específico, organizadas pela média das notas das avaliações dos usuários.

```
SELECT
   Midia.titulo,
   Midia.data_lancamento,
   A.media_aval,
   Midia_Franquia.nome AS franquia,
   Producao.nome_organizacao AS produtora
FROM
   Midia
   JOIN Participacao ON (
       Midia.id_midia = Participacao.id_midia
       AND Participacao.nome_pessoa IN ('Robert Downey Jr.',
'Ícaro Silva')
       AND Participacao.natureza = 'ATOR'
   )
   LEFT JOIN (
       SELECT
           id_midia,
           AVG(nota) AS media_aval
       FROM
           Avaliacao
       GROUP BY
           id_midia
       HAVING
           AVG(nota) > 2
   ) A ON (Midia.id_midia = A.id_midia)
   JOIN Producao ON (
       Midia.id_midia = Producao.id_midia
```

```
        AND Producao.nome_organizacao IN ('Marvel Studios',
'Storytel')
    )
    LEFT JOIN Midia_Franquia ON (Midia.id_midia =
Midia_Franquia.id_midia)
WHERE
    DATE_PART('year', Midia.data_lancamento) BETWEEN 2018
    AND 2020
ORDER BY
    A.media_aval DESC;
```

7.2.

Consultar todos os atores que participaram de um certo filme e todas as indicações que eles receberam, se houver alguma.

```
SELECT
    Participacao.nome_pessoa,
    Participacao.data_nasc_pessoa,
    Indicacao.indicacao,
    Indicacao.premiado
FROM
    (
        SELECT
            id_midia
        FROM
            Midia
        WHERE
            Midia.titulo = 'Avatar'
    ) M
    JOIN Participacao ON (
        M.id_midia = Participacao.id_midia
        AND Participacao.natureza = 'ATOR'
    )
    LEFT JOIN Indicacao ON (
        Participacao.id_participacao =
Indicacao.id_participacao
    );
```

7.3.

Consultar as mídias de uma franquia, ordenadas por franquia e por nota média dada pelos usuários.

```
SELECT
   Midia_franquia.nome,
   AVG(A.media_aval) AS media_aval_franquia,
   SUM(A.nro_aval) AS qtd_aval_franquia
FROM
   Midia
   JOIN (
       SELECT
           id_midia,
           COUNT(*) AS nro_aval,
           AVG(nota) AS media_aval
       FROM
           Avaliacao
       GROUP BY
           id_midia
   ) A ON (Midia.id_midia = A.id_midia)
   JOIN Midia_Franquia ON (Midia.id_midia =
Midia_Franquia.id_midia)
GROUP BY
   Midia_franquia.nome
ORDER BY
   media_aval_franquia;
```

7.4.

Consultar o número de vezes que cada usuário teve um conteúdo editado por um moderador e o número de penalidades que ele sofreu.

```
SELECT
   Usuario.email,
   Usuario.nome,
   COUNT(Mod_Cadastro) AS mod_Cadastro,
   SUM(Mod_Cadastro.penalidade) AS penalidade_cadastro,
   COUNT(Mod_Edicao) AS mod_edicao,
   SUM(Mod_Edicao.penalidade) AS penalidade_edicao,
   COUNT(Mod_Comentario) AS mod_comentario,
   SUM(Mod_Comentario.penalidade) AS penalidade_comentario,
   COUNT(Mod_Avaliacao) AS mod_avaliacao,
   SUM(Mod_Avaliacao.penalidade) AS penalidade_avaliacao
FROM
   Usuario
   LEFT JOIN Mod_Cadastro ON (Usuario.email =
Mod_Cadastro.email_cadastrante)
```

```
   LEFT JOIN Mod_Edicao ON (Usuario.email =
Mod_Edicao.email_editor)
   LEFT JOIN (
       Mod_Comentario
       JOIN Comentario ON (
           Mod_Comentario.id_comentario =
Comentario.id_comentario
       )
   ) ON (Usuario.email = Comentario.email)
   LEFT JOIN Mod_Avaliacao ON (Usuario.email =
Mod_Avaliacao.email_avaliador)
GROUP BY
   Usuario.email,
   Usuario.nome;
```

7.5.

Consultar todos Usuários que avaliaram todas as Mídias de uma dada Franquia.

```
SELECT
   email, nome
FROM
   Usuario AS U
WHERE
   NOT EXISTS (
       (
           SELECT
               Midia.id_midia
           FROM
               Midia,
               Midia_Franquia
           WHERE
               Midia.id_midia = Midia_Franquia.id_midia
               AND Midia_Franquia.nome = 'Vingadores'
       )
       EXCEPT
           (
               SELECT
                   id_midia
               FROM
                   Avaliacao
               WHERE
                   email = U.email
```

```
            )
  );
```

8. Conclusão

O projeto como um todo foi interessante e ajudou bastante no entendimento de banco de dados, porém foi muito trabalhoso, especialmente na parte 1, que foi bastante desafiadora, pois requer a escolha de um tema que fosse suficientemente complexo e concomitantemente factível no período disponível. Outro ponto de dificuldade foi a implementação, que é trabalhosa e ocorre no fim do semestre, período bastante ocupado devido à provas e outros trabalhos.