Relatório do Trabalho Final de Banco de Dados

Gustavo de Mendonça Freire
123102270

João Victor Lopez Pereira
123317370

William Victor Quintela Paixão
123089993

Yuri Rocha de Albuquerque
123166143

29 de dezembro de 2024

Rio de Janeiro - RJ

Sumário

Capítulo 1

Introdução

O *website YouTube*, de propriedade da *Google*, consiste em uma plataforma *online* de compartilhamento de vídeos. Diariamente, milhões de usuários publicam e assistem vídeos de assuntos variados no *site* através de seus computadores, dispositivos móveis ou mesmo *smart TVs* de vários países ao redor do mundo. O *YouTube*, por meio de um algoritmo de recomendação, seleciona, para um dado usuário, conteúdo voltado aos seus interesses. Como resultado desse processo, certos vídeos se popularizam de maneira notória, passando a participar de uma seção especial na plataforma, denominada "em alta". Os vídeos dessa seção variam de acordo com o país, sendo portanto, adequada à realidade do usuário que a acessa.

O tema do nosso trabalho é a análise de tendências no *YouTube*, com foco em compreender quais tipos de vídeos estão em alta na plataforma, identificar os canais mais influentes e observar o engajamento que esses vídeos recebem, medido em termos de curtidas, comentários e visualizações. Através dessa análise, buscamos extrair *insights* que permitam caracterizar o que define um vídeo popular, os padrões de conteúdo que se destacam, e a relação entre a popularidade e o país de origem dos vídeos. Para isso, estamos nos guiando através das seguintes principais questões: "Quais categorias de vídeos estão em alta?", "Quais tipos de canais produzem o conteúdo mais relevante e visualizado?", "Como as tendências variam de acordo com as regiões?" e "Qual a relação entre quantidade de comentários e *likes* e a classificação de um vídeo?". Essas perguntas, além de norteadoras, nos motivaram a escolher esse tema. Por meio desta análise, buscamos não apenas identificar padrões regionais e globais, mas também entender como os algoritmos da plataforma podem influenciar o comportamento do público e o sucesso de certos tipos de conteúdo.

Consultas realizadas, assim como a modelagem física em `SQL`, a modelagem conceitual e lógica utilizando o `brModelo`, e a coleta de dados implementada em `Python`, podem ser encontrados em [5].

Capítulo 2

Datasets

Os conjuntos de dados utilizados estão sendo extraídos por meio de um programa em `Python` que na verdade é o `YouTube Data Scraper` do usuário `mitchelljy`, publicado em seu *GitHub*, incrementado com diversas modificações que nos permitem extrair dados públicos de nosso interesse. A chave `YouTube Data API v3` que está sendo utilizada foi adquirida por meio do *YouTube* em uma de nossas próprias contas. Os dados extraídos pelo programa são armazenados em arquivos *CSV* que o próprio programa gera. É importante mencionar que a página de `mitchelljy` do *GitHub* foi encontrada por meio do *site Kaggle*[4].

Os dados contidos no *CSV* e seus respectivos tipos são:

- `video_id`: contém, em cada registro, o *ID* do vídeo gerado pelo próprio *YouTube*. Esse *ID* é utilizado para, além de mera caracterização de vídeos únicos na plataforma, a geração de suas *URL*s, por exemplo. Cada *ID* é uma sequência alfanumérica de símbolos, sendo mais apropriado modelá-lo como cadeia de caracteres (*string*) em nosso sistema;
- `title`: refere-se ao título do vídeo em sua língua original (em que foi publicado). Suporta caracteres de *UTF-8*. É, naturalmente, do tipo textual, sendo, portanto, claramente representável por valores do domínio de *strings*;
- `publishedAt`: corresponde à data e hora de publicação do vídeo (com relação ao Tempo Universal Coordenado *UTC*). O valores desse atributo são resgatados no formato *ISO 8601* e podem ser mapeados para o domínio *DATETIME* do *SGBD*;
- `channelId`: análogo ao *ID* gerado para o vídeo, este campo armazena o *ID* criado pelo *site* para cada canal, neste caso, o canal que publicou o vídeo. É uma cadeia alfanumérica e, consequentemente, está no domínio de *strings*;
- `channelTitle`: também análogo ao título do vídeo, é relativo ao título do canal que postou o determinado vídeo. É do tipo textual e mapeado no tipo *string*;
- `categoryId`: corresponde ao identificador da categoria atribuída ao vídeo pelo canal que o publicou. No guia de referências da *API* consta que é um dado do tipo *string*, porém, após coleta da listagem de categorias disponíveis, percebemos que todas se tratam de inteiros de 1 a 44 (pulando alguns). Portanto, consideramos que a atribuição ao domínio dos inteiros seja a melhor alternativa;
- `trending_date`: contém a data em que o vídeo está em alta, ou seja, equivale à data em que o arquivo *CSV* foi criado. Está em um formato diferente do utilizado pela *API* do *YouTube*,

pois é uma informação adicionada artificialmente no *script*. Consiste do ano, dia e mês de coleta dos dados, nesta ordem, separados por ponto final (AA.DD.MM). É naturalmente associável ao domínio *DATE*;

- `tags`: corresponde a uma lista de *tags* (etiquetas) atribuídas ao vídeo, usadas como auxílio para o algoritmo de recomendações da plataforma. São pequenas porções de texto (ocorrências de `tag`) separadas por vírgulas (','). É permitido que a `tag` contenha espaços. Cada `tag` encaixa-se no domínio *string*;

- `view_count`: é o número total de visualizações que o vídeo teve desde sua publicação até o instante da captura dos dados. Consiste de um inteiro não-negativo (logicamente) e, por isso, pode ser modelado por um inteiro longo sem sinal;

- `likes`: é uma métrica numérica, assim como a contagem de visualizações, referente ao número de curtidas (*likes*) que o vídeo obteve até a coleta dos dados. É inteiro longo maior ou igual a zero;

- `comment_count`: novamente, é um campo numérico que retrata a quantidade de comentários que o vídeo recebeu até a captura dos dados. Pertence ao domínio de inteiros longos sem sinal;

- `thumbnail_link`: é a *URL* da *thumbnail* (imagem de capa) do vídeo. Por ser um *link*, que é composto por vários caracteres, alfanuméricos e símbolos especiais, em sequência, escolhemos o tipo *string* para armazená-los;

- `comments_disabled`: indica se o canal que publicou o vídeo optou por desativar a postagem de comentários. Nos arquivos *CSV*, esse campo é assinalado por `True` ou `False`, devendo, evidentemente, ser tratado como um valor de domínio *booleano*;

- `ratings_disabled`: informa se o canal desativou a opção dos espectadores curtirem (ou descurtirem) o vídeo. Assim como `comments_disabled`, somente assume valores `True` e `False` e, por isso, é traduzido para valores do espectro *booleano*;

- `description`: é a descrição do vídeo fornecida pelo seu publicador na língua original. É um texto de tamanho completamente variável (no máximo 5000 caracteres) e pode conter símbolos da codificação *UTF-8*. Logo, é do tipo *string*.

- `channel_creation_date`: diz respeito à data e hora (no mesmo formato `ISO 8601` do campo `published_at`) de criação do canal que publicou o vídeo. É mapeado no domínio *DATETIME* do `SGBD`;

- `channel_subscriber_count`: corresponde ao número de inscritos que o canal publicador possui no momento da coleta de dados. Assim como as métricas do vídeo, é um dado altamente mutável e que assume valores inteiros potencialmente grandes. Por esse motivo, optamos por usar o domínio *BIGINT*;

- `channel_video_count`: é a quantidade de vídeos publicados, até então, pelo canal responsável pelo vídeo em questão. Dificilmente passará a marca de $2^{31} - 1$, então assumimos tipo inteiro de 4 *bytes*;

- `video_url`: condiz com a *URL* gerada pelo *YouTube* para o vídeo (que foi discutida na descrição de `video_id`). É obtida, no código de coleta e organização dos dados, a partir da concatenação da *URL* padrão para redirecionamento em vídeo com o *ID* do vídeo. Por corresponder a uma sequência de caracteres, é melhor modelado por uma *string*;

- `channel_description`: refere-se a uma descrição textual do canal dada por seu criador, similar à descrição do vídeo. Escolheu-se o tipo *TEXT* para armazená-la do banco de dados;

- `channel_image`: relativo ao ícone do canal. Seu formato é de uma *URL* que redireciona para a imagem em questão. Por isso, decidimos tratar como *string*;

- `channel_country`: configura o código do país ao qual o canal publicador está associado. É determinado por uma *string* de exatamente dois caracteres (ou três para `N/A`, se o canal não o possui) e *supomos* que equivale ao padrão `ISO 3166-1`[6]

  (não encontramos evidências explícitas sobre isso na documentação da *API*, apenas conferimos por inspeção). Portanto, é uma *string* de tamanho máximo 3;

- `channel_total_views`: relacionado com a quantidade total de visualizações que o canal teve em toda sua história. Como considerável parte dos canais listados dentre os vídeos em alta são de alta relevância, os valores nesse campo são bem grandes e só conseguem ser armazenados se tratados como inteiros longos;

- `channel_keywords`: é uma lista de palavras-chave escolhidas pelo detentor do canal para facilitar o trabalho de algoritmos de recomendação (assim como as tags dos vídeos). No conjunto de dados gerado, corresponde a uma grande *string* com as palavras-chave sendo separadas por espaços. Cada palavra-chave é uma *string* e todas, juntas, podem somar até 500 caracteres[1];

- `channel_url`: trata-se da *URL* padrão do canal que postou o vídeo, aquela que o próprio *YouTube* gera através da concatenação do *ID* do canal ao esquema genérico de um *link* para canal (foi exatamente dessa forma que foi modelado no *script* que fornece o *dataset*). Guardamos cada valor desse campo em uma *string*.

Capítulo 3

Projeto do Banco de Dados

3.1 Modelo Conceitual

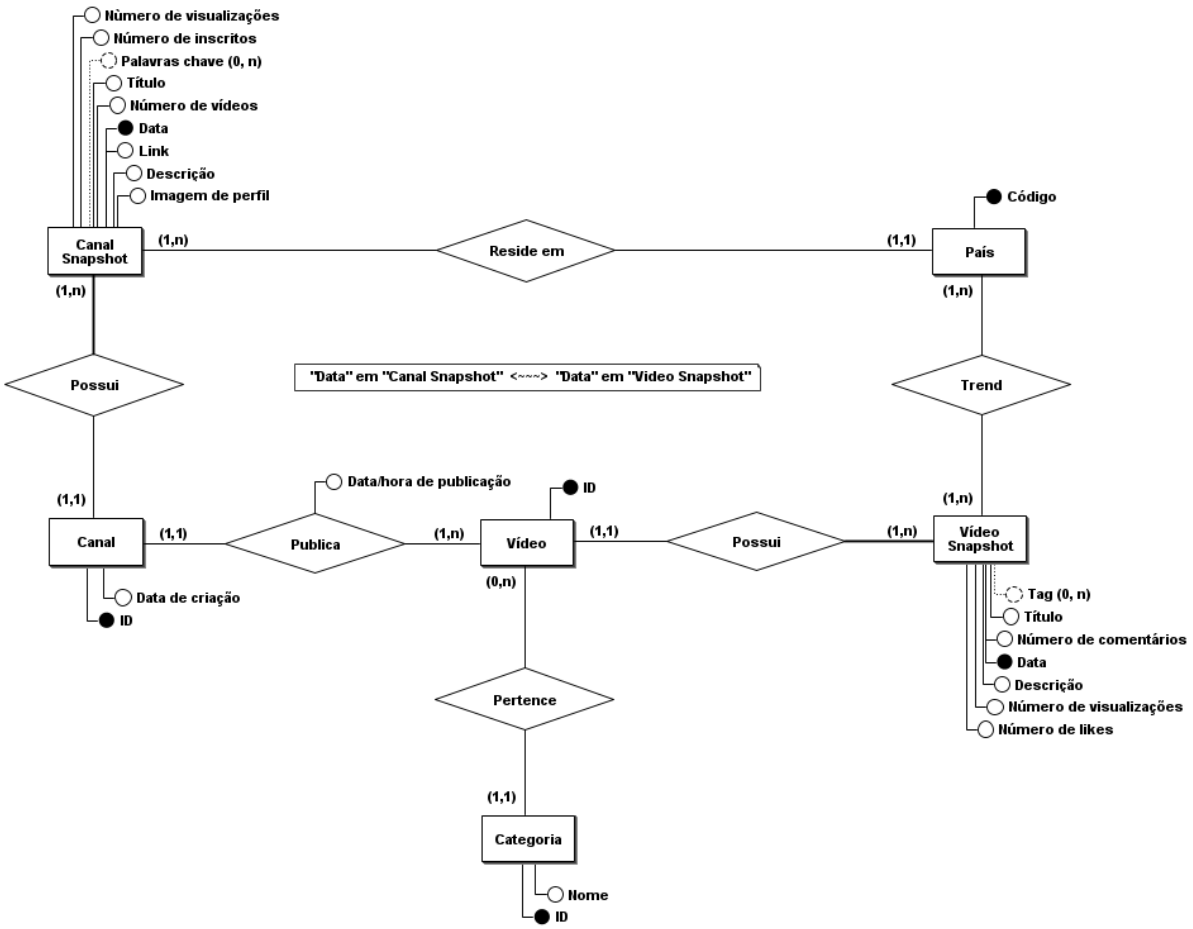


Figura 3.1: Modelagem conceitual do *dataset* utilizado.

O foco da nossa modelagem está em capturar o estado temporal de canais e vídeos em alta. As entidades principais são o `Canal Snapshot` e o `Vídeo Snapshot`, que representam "fotografias" do estado de um canal e de um vídeo em um momento específico.

O `Canal Snapshot` armazena informações temporais sobre um canal, como número de visualizações, inscritos, palavras-chave, título, número de vídeos, data da *snapshot*, *link*, descrição e imagem de perfil. Cada `Canal Snapshot` está associado a um `País`, o que permite registrar as características do canal em um contexto geográfico específico. Essa abordagem nos permite acompanhar a evolução e a presença do canal em diferentes regiões ao longo do tempo.

Por outro lado, a entidade `Vídeo Snapshot` captura o estado de um vídeo em uma data específica, incluindo atributos como título, número de comentários, data, descrição, número de visualizações e número de likes. Essa estrutura é essencial para registrar a popularidade do vídeo exclusivamente na data em que ele se torna viral ou aparece em alta. A entidade `Trend` conecta cada `Vídeo Snapshot` a um `País`, representando os momentos em que o vídeo esteve em destaque em uma região específica, possibilitando uma análise detalhada de sua trajetória de popularidade. Para armazenar os dados atemporais de um vídeo (por exemplo, associá-lo ao canal que o publica, ou definir a sua categoria), cada `Vídeo Snapshot` está associado a uma entidade `Vídeo`.

Além disso, a modelagem inclui a entidade `Canal`, que representa o canal original que publica os vídeos e possui atributos fixos, como *ID* e data de criação. Cada vídeo é publicado por um canal e é classificado em uma `Categoria` (como `Gaming` ou `Comedy`, por exemplo), que permanece inalterada ao longo do tempo.

Essa modelagem com `Canal Snapshot` e `Vídeo Snapshot` como entidades centrais possibilita o monitoramento detalhado dos vídeos em alta, permitindo acompanhar seus atributos em diferentes períodos e regiões.

Além disso, colocamos uma observação na modelagem que "`Data em Canal Snapshot se e somente se Data em Video Snapshot`". Tivemos que colocar essa restrição visto que nossa extração de dados garante que sempre que haja uma instância de `Canal Snapshot` com determinada `Data`, necessariamente há uma instância de `Vídeo Snapshot` com mesma `Data` e, analogamente, sempre que haja uma instância de `Vídeo Snapshot` com determinada `Data`, necessariamente há uma instância de `Canal Snapshot` com mesma `Data`.

3.2 Modelo Lógico

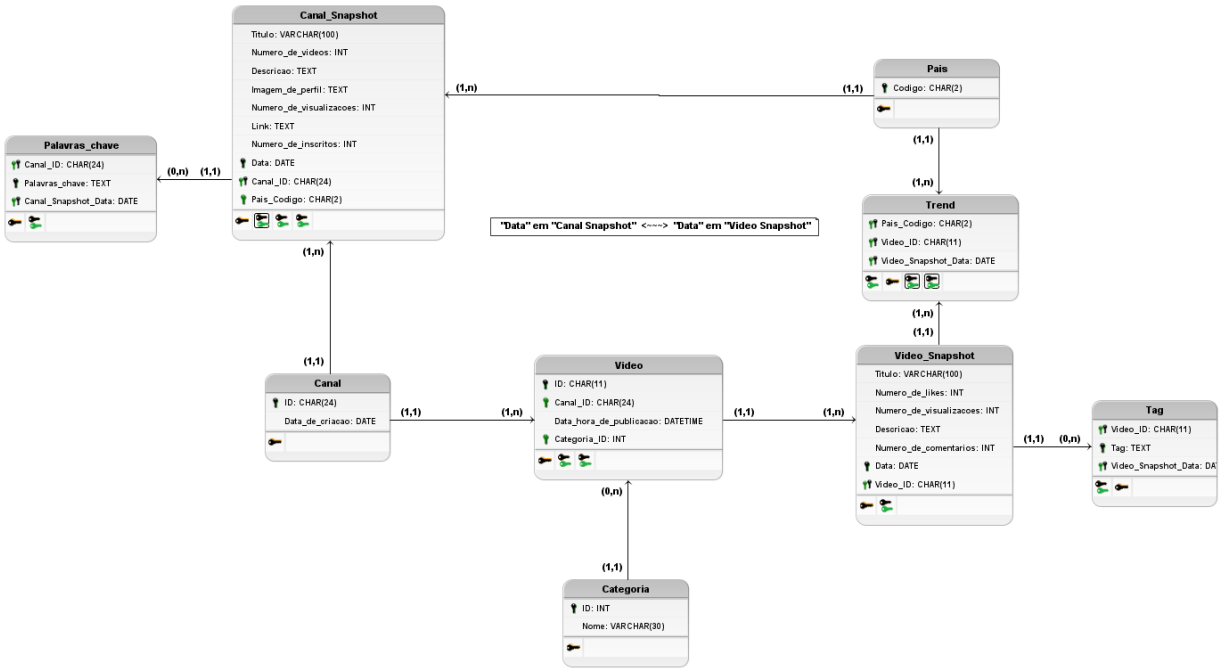


Figura 3.2: Modelagem lógica do *dataset* utilizado.

Utilizamos a própria ferramenta `brModelo` para a tradução da modelagem conceitual para a modelagem lógica. Ainda assim, o resultado da tradução apresentou diversos erros como chaves primárias e estrangeiras em locais errados, nomes de variáveis incoerentes. Sendo assim, tivemos que corrigir esses detalhes da modelagem lógica gerada pela ferramenta.

Acreditamos que, após nossas correções, o modelo lógico representa nosso modelo com exatidão.

3.3 Modelo Físico

3.3.1 Criação da Modelagem

Primeiramente, a modelagem física foi projetada diretamente da lógica, da qual convertemos para a física utilizando a ferramenta presente no próprio `brModelo`, fazendo as alterações necessárias conforme fomos verificando os erros da conversão automática. Após isso, para popular o banco de dados, utilizamos o *CSV* gerado pela *API*, separando-o em outros *CSV*s menores que correspondiam a cada tabela (utilizando a biblioteca `pandas` da linguagem `Python` para tratar o formato dos dados e fazer essa separação), portanto, continham apenas as colunas requeridas por cada uma. Dito isso, foi utilizada uma função de importação, do próprio `SQL`, que importa todas as linhas de um *CSV* para tuplas de uma tabela específica. A seguir, encontra-se o código utilizado para a criação dos esquemas desse banco de dados e como os populamos com os *CSV*s gerados (junto com o código de criação).

1. Tabela `Video`

   ```
   CREATE TABLE Video (
     ID CHAR(11) PRIMARY KEY,
     Canal_ID CHAR(24),
     Data_hora_de_publicacao DATE,
     Categoria_ID INT
   );
   ```

   A tabela `Video` contém informações de cada vídeo, incluindo um identificador único, um campo de identificação para o canal responsável, a data de publicação e a categoria do vídeo.

2. Tabela `Categoria`

   ```
   CREATE TABLE Categoria (
     ID INT PRIMARY KEY,
     Nome VARCHAR(30)
   );
   ```

   A tabela `Categoria` armazena as categorias disponíveis, onde cada categoria possui um identificador e um nome. Essa tabela foi populada utilizando um *CSV* estático que contém todas as categorias possíveis.

3. Tabela `Canal`

   ```
   CREATE TABLE Canal (
     ID CHAR(24) PRIMARY KEY,
     Data_de_criacao DATE
   );
   ```

   A tabela `Canal` define os canais existentes, identificados por um *ID* único e com uma data de criação.

4. Tabela `Pais`

   ```
   CREATE TABLE Pais (
     Codigo VARCHAR(3) PRIMARY KEY
   );
   ```

A tabela `Pais` apenas registra os códigos dos países, onde seu código é a chave primária. Inicialmente, tivemos um problema ao converter os valores do *CSV* para essa tabela, causando comportamentos inesperados no código. Por algum motivo, os campos do *CSV* não eram lidos com o tipo correto que era previsto, gerando diversos erros ao tentar adicionar chaves estrangeiras para essas tabelas, pois não estava havendo correspondência. Resolvemos, temporariamente, colocando os valores de forma manual nessa tabela.

Foi necessário um tratamento adicional nos dados, pois, além de vários países duplicados, alguns canais não eram associados com nenhum país. Sendo assim, deixamos os canais sem países associados ao país `N/A` por padrão, porque estava havendo erros ao tentar utilizar a chave estrangeira com a tabela `Pais`, como já foi dito (e ao adicionar valores `NULL`).

5. Tabela `Video_Snapshot`

```
CREATE TABLE Video_Snapshot (
  Titulo VARCHAR(100),
  Numero_de_likes BIGINT,
  Numero_de_visualizacoes BIGINT,
  Descricao TEXT,
  Numero_de_comentarios BIGINT,
  Data DATE,
  Video_ID CHAR(11),
  PRIMARY KEY (Data, Video_ID)
);
```

A tabela `Video_Snapshot` guarda informações em um certo momento do tempo detalhadas de cada vídeo, como o título, número de likes, visualizações, descrição, número de comentários e a data específica desse instante dos dados, vinculados ao vídeo identificado pelo *ID* do vídeo. Esta tabela permite o registro de *snapshots* dos dados de vídeos ao longo do tempo.

6. Tabela `Canal_Snapshot`

```
CREATE TABLE Canal_Snapshot (
  Titulo VARCHAR(100),
  Numero_de_videos INT,
  Descricao TEXT,
  Imagem_de_perfil TEXT,
  Numero_de_visualizacoes BIGINT,
  Link TEXT,
  Numero_de_inscritos BIGINT,
  Data DATE,
  Canal_ID CHAR(24),
  Pais_Codigo VARCHAR(3),
  PRIMARY KEY (Data, Canal_ID)
);
```

De maneira similar à anterior, a tabela `Canal_Snapshot` armazena *snapshots* de canais de um certo momento no tempo, com dados sobre título, número de vídeos, descrição, imagem de perfil, visualizações totais, *link*, número de inscritos, a data da *snapshot* e o país do canal.

Também foi necessário um tratamento adicional aqui, pois alguns canais aparecem mais de uma vez de maneira errônea.

7. Tabela `Tag`

```
CREATE TABLE Tag (
```

```
  Video_ID CHAR(11) NOT NULL,
  Tag VARCHAR(500),
  Video_Snapshot_Data DATE,
  PRIMARY KEY (Video_ID, Tag, Video_Snapshot_Data)
);
```

A tabela `Tag` registra as *tags* associadas a cada vídeo, e cada linha possui o identificador do vídeo, uma lista das *tags* e a data do *snapshot* de vídeo relacionado, permitindo manter as *tags* de um vídeo em diferentes *snapshots*. (No código, foi utilizado uma função para lidar com esses campos multivalorados, separando em diversas linhas distintas).

Nesse caso, tivemos que utilizar a função `explode` do `pandas` para separar campos multivalorados em diversas linhas

8. Tabela `Palavra_chave`

```
CREATE TABLE Palavra_chave (
  Canal_ID CHAR(24) NOT NULL,
  Palavra_chave VARCHAR(500),
  Canal_Snapshot_Data DATE,
  PRIMARY KEY (Canal_ID, Palavra_chave, Canal_Snapshot_Data)
);
```

De modo equivalente, a tabela `Palavra_chave` armazena palavras-chave de cada canal, relacionadas às datas das *snapshots* específicas do canal.

Aqui, tivemos que usar novamente a função para separar um campo multivalorado em diversas outras linhas.

9. Tabela `Trend`

```
CREATE TABLE Trend (
  Pais_Codigo VARCHAR(3),
  Video_ID CHAR(11),
  Video_Snapshot_Data DATE,
  PRIMARY KEY (Pais_Codigo, Video_ID, Video_Snapshot_Data)
);
```

Por fim, a tabela `Trend` contém informações sobre as tendências dos vídeos, associadas ao código do país, ao vídeo e ao snapshot de vídeo de uma data específica.

Depois da declaração dos esquemas, foi feito a declaração das chaves estrangeiras de acordo com o planejado no modelo lógico, ficando:

```
ALTER TABLE Video ADD CONSTRAINT FK_Video_Canal_ID
  FOREIGN KEY (Canal_ID)
  REFERENCES Canal (ID)
  ON DELETE RESTRICT;

ALTER TABLE Video ADD CONSTRAINT FK_Video_Categoria_ID
  FOREIGN KEY (Categoria_ID)
  REFERENCES Categoria (ID)
  ON DELETE RESTRICT;

ALTER TABLE Video_Snapshot ADD CONSTRAINT FK_Video_Snapshot_Video_ID
  FOREIGN KEY (Video_ID)
```

```
  REFERENCES Video (ID)
  ON DELETE CASCADE;

ALTER TABLE Canal_Snapshot ADD CONSTRAINT FK_Canal_Snapshot_Canal_ID
  FOREIGN KEY (Canal_ID)
  REFERENCES Canal (ID)
  ON DELETE CASCADE;

ALTER TABLE Canal_Snapshot ADD CONSTRAINT
  FK_Canal_Snapshot_Pais_Codigo
  FOREIGN KEY (Pais_Codigo)
  REFERENCES Pais (Codigo)
  ON DELETE RESTRICT;

ALTER TABLE Tag ADD CONSTRAINT FK_Tag_Video_ID
  FOREIGN KEY (Video_ID)
  REFERENCES Video_Snapshot (Video_ID)
  ON DELETE CASCADE;

ALTER TABLE Tag ADD CONSTRAINT FK_Tag_Video_Snapshot_Data
  FOREIGN KEY (Video_Snapshot_Data)
  REFERENCES Video_Snapshot (Data)
  ON DELETE CASCADE;

ALTER TABLE Palavra_chave ADD CONSTRAINT
  FK_Palavra_chave_Canal_Snapshot_Data
  FOREIGN KEY (Canal_Snapshot_Data)
  REFERENCES Canal_Snapshot (Data)
  ON DELETE CASCADE;

ALTER TABLE Palavra_chave ADD CONSTRAINT FK_Palavra_chave_Canal_ID
  FOREIGN KEY (Canal_ID)
  REFERENCES Canal_Snapshot (Canal_ID)
  ON DELETE CASCADE;

ALTER TABLE Trend ADD CONSTRAINT FK_Trend_Pais_Codigo
  FOREIGN KEY (Pais_Codigo)
  REFERENCES Pais (Codigo)
  ON DELETE RESTRICT;

ALTER TABLE Trend ADD CONSTRAINT FK_Trend_Video_ID
  FOREIGN KEY (Video_ID)
  REFERENCES Video_Snapshot (Video_ID)
  ON DELETE RESTRICT;

ALTER TABLE Trend ADD CONSTRAINT FK_Trend_Video_Snapshot_Data
  FOREIGN KEY (Video_Snapshot_Data)
  REFERENCES Video_Snapshot (Data)
  ON DELETE RESTRICT;
```

3.3.2 Evidência de População do Banco de Dados

Como evidência de criação e população do banco, segue capturas de telas das seguintes consultas:

Canal

COUNT(*)
2516

Figura 3.3: Contador de `Canal`.

ID	Data_de_criacao
UC__AsSnEuyVgO9TWvZE_ziA	2012-02-27
UC_-gT7OYiRCK9Sp8SIv9WgQ	2006-08-02
UC_1O5w8-Gv812YiZwxd1qSw	2021-05-24
UC_1QUZJSTYqda_dVYlNV03w	2014-02-01
UC_446tDNo7UckPX78hM0Nlg	2012-10-11
UC_5niPa-d35gg88HaS7RrIw	2012-02-11
UC_7520oUmZ2piOVjxc3D_nw	2014-01-03
UC_auhJENZIGOfoqWheFe3Ow	2021-10-13
UC_Av98lDjf5KvFib5elhpYg	2014-12-11
UC_bPnE6C8ucaq7JHPKlaf-Q	2022-03-13
NULL	NULL

Figura 3.4: Exemplo de dados relativos a `canal`.

Canal Snapshot

COUNT(*)
16330

Figura 3.5: Contador de `Canal_Snapshot`.

Numero_de_inscritos	Data	Canal_ID	Pais_Codigo
1260000	2024-10-19	UC__AsSnEuyVgO9TWvZE_ziA	JP
1980000	2024-10-19	UC_446tDNo7UckPX78hM0Nlg	FR
502000	2024-10-19	UC_auhJENZIGOfoqWheFe3Ow	US
1030000	2024-10-19	UC_CXtrkW1a343sChq8DnBGA	FR
2860000	2024-10-19	UC_F3i4sHKon8CwRxvU-SNhA	US
461000	2024-10-19	UC_HN4UwLGkimyAzgfDhap9w	FR
427000	2024-10-19	UC_K39BFds6dGtK6aVtqZKbw	FR
3810000	2024-10-19	UC_Np5NHrORbF6wbRVZjMHtg	HK
3760000	2024-10-19	UC_oToDrJ6uca7d1dFVBmLtg	BR
1840000	2024-10-19	UC_pdYA_tB0YXnbzYqg3lh_A	NULL
NULL	NULL	NULL	NULL

Figura 3.6: Exemplo de dados relativos a `Canal_Snapshot`.

Titulo	Numero_de_videos	Descricao	Imagem_de_perfil	Numero_de_visualizacoes	Link
SekineRisa	1609	美容・旅行などを中心に動画をアップしてます!動...	https://yt3.ggpht.com/ytc/AIdro_n2r9Uw-3XIh...	691253494	https://www.youtube.com/channel/UC__AsSnE...
Neoxi	365	Moi c'est Valentin, je reviens sur certaines affair...	https://yt3.ggpht.com/kYpTG2OWsqgHNI8KOF...	235490113	https://www.youtube.com/channel/UC_446tDN...
Alan Roblox	218	Hi. I'm just love the game RobloxROBLOX Brook...	https://yt3.ggpht.com/BeRcW9OEckKYGQdxJu...	91371210	https://www.youtube.com/channel/UC_auhJEN...
Karaté Bushido Officiel	1686	La chaîne KARATE BUSHIDO OFFICIEL est la ch...	https://yt3.ggpht.com/wc-TFqXsu3eU38nN_2I...	410995377	https://www.youtube.com/channel/UC_CXtrkW...
KING VADER	123	THIS IS KING VADER'S OFFICIAL YOUTUBE CHA...	https://yt3.ggpht.com/ytc/AIdro_nwcaom8ZEdl...	350728112	https://www.youtube.com/channel/UC_F3i4sH...
Julien Doré	79		https://yt3.ggpht.com/Wxunv5LxhbA-3cxUk0g...	413762095	https://www.youtube.com/channel/UC_HN4Uw...
Lulu ronce de noyer	80	Bienvenue sur ma deuxième chaine youtube ! C...	https://yt3.ggpht.com/uJ6CXpx-n3vY1g7HLgqe...	27636069	https://www.youtube.com/channel/UC_K39BFd...
火影忍者一家	911	欢迎来到【火影忍者】频道！🍥👁️🐍这里是火...	https://yt3.ggpht.com/qAJ9zUVxa9FXlxBSDiP0t...	3174037562	https://www.youtube.com/channel/UC_Np5NHr...
Canal GOAT	2640	Faça parte da comunidade GOAT na Casa de A...	https://yt3.ggpht.com/lQbsmXG4-b751D81rRQ...	378656875	https://www.youtube.com/channel/UC_oToDrJ...
Poli Landim	1369		https://yt3.ggpht.com/ytc/AIdro_k4hwWZJ6t7...	254966019	https://www.youtube.com/channel/UC_pdYA_t...
NULL	NULL	NULL	NULL	NULL	NULL

Figura 3.7: Exemplo de dados relativos a `Canal_Snapshot`.

Categoria

COUNT(*)
32

Figura 3.8: Contador de `Categoria`.

ID	Nome
1	Film & Animation
2	Autos & Vehicles
10	Music
15	Pets & Animals
17	Sports
18	Short Movies
19	Travel & Events
20	Gaming
21	Videoblogging
22	People & Blogs
NULL	NULL

Figura 3.9: Exemplo de dados relativos a `Categoria`.

Pais

COUNT(*)
63

Figura 3.10: Contador de `Pais`.

Codigo
AE
AG
AR
AT
AU
AZ
BE
BG
BR
BY
NULL

Figura 3.11: Exemplo de dados relativos a `Pais`.

Palavra Chave

COUNT(*)
207110

Figura 3.12: Contador de `Palavra_Chave`.

Canal_ID	Palavra_chave	Canal_Snapshot_Data
UC__AsSnEuyVgO9TWvZE_ziA	hairmake	2024-10-19
UC__AsSnEuyVgO9TWvZE_ziA	kawaii	2024-10-19
UC__AsSnEuyVgO9TWvZE_ziA	make	2024-10-19
UC__AsSnEuyVgO9TWvZE_ziA	youtuber	2024-10-19
UC__AsSnEuyVgO9TWvZE_ziA	デカ目	2024-10-19
UC__AsSnEuyVgO9TWvZE_ziA	ヘアメイク	2024-10-19
UC__AsSnEuyVgO9TWvZE_ziA	メイク	2024-10-19
UC__AsSnEuyVgO9TWvZE_ziA	ユーチューバー	2024-10-19
UC_446tDNo7UckPX78hM0Nlg	criminelles	2024-10-19
UC_446tDNo7UckPX78hM0Nlg	documentaire	2024-10-19
NULL	NULL	NULL

Figura 3.13: Exemplo de dados relativos a `Palavra_Chave`.

Tag

COUNT(*)
244661

Figura 3.14: Contador de `Tag`.

Video_ID	Tag	Video_Snapshot_Data
_9FkcwZmDEs	P丸様。	2024-10-19
_9FkcwZmDEs	tiktok	2024-10-19
_9FkcwZmDEs	アニメ	2024-10-19
_9FkcwZmDEs	お笑い	2024-10-19
_9FkcwZmDEs	コント	2024-10-19
_9FkcwZmDEs	そら谷のアニメっち	2024-10-19
_9FkcwZmDEs	たすくこま	2024-10-19
_9FkcwZmDEs	ちくわ	2024-10-19
_9FkcwZmDEs	はじめまして松尾です	2024-10-19
_9FkcwZmDEs	ひょうどうチャンネル	2024-10-19
NULL	NULL	NULL

Figura 3.15: Exemplo de dados relativos a `Tag`.

Trend

COUNT(*)
23792

Figura 3.16: Contador de `Trend`.

Pais_Codigo	Video_ID	Video_Snapshot_Data
AU	_4qLcHbQbbQ	2024-10-19
AU	_AZTn6_Kv-s	2024-10-19
AU	_jTFLg3arYU	2024-10-19
AU	_QgjyAm5uxo	2024-10-19
AU	_rAdNU-NIx8	2024-10-19
AU	_VwVGwNA3Xs	2024-10-19
AU	-1-ldW4kpLM	2024-10-19
AU	-1Gw3h1X_vk	2024-10-19
AU	0VuIgNqt6Cg	2024-10-19
AU	1DIDRFO6snQ	2024-10-19
NULL	NULL	NULL

Figura 3.17: Exemplo de dados relativos a `Trend`.

Video

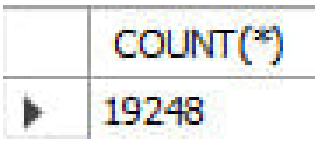
COUNT(*)
4025

Figura 3.18: Contador de `Video`.

ID	Canal_ID	Data_de_publicacao	Categoria_ID
___LS7bppps	UCn0VbHe_Chjws3qjUF4JJMw	2024-11-02	20
_0qzLGtpv2M	UCy0I5Hcl2k7dN4UdZTedfeQ	2024-11-07	17
_4qLcHbQbbQ	UCH569oN0jV7lr0QZFLD4AZA	2024-10-15	22
_5CTg5asoC0	UC8-Th83bH_thdKZDJCrn88g	2024-10-23	23
_6FKzrWSW0Y	UCWeGmNiIrrqcDYdLh9sjVDg	2024-11-09	1
_6HA4PD2zZw	UCky8Z5AYEHA55-r2TFNC_KA	2024-10-06	24
_6yuXxFEUT8	UCIXguhHCl8eDTkXpEuiGPUA	2024-11-04	20
_7EKz_rKxAw	UCwpyuvmJ_mOrebYbUPXtaBQ	2024-11-05	17
_7ZuqNdDc5Q	UCEU5ZK7DwN9ppqPFJiGah3A	2024-10-31	10
_9FkcwZmDEs	UCOnA15zQ7OafLsnN8J-CMvg	2024-10-16	1
NULL	NULL	NULL	NULL

Figura 3.19: Exemplo de dados relativos a `Video`.

Video Snapshot



	COUNT(*)
▶	19248

Figura 3.20: Contador de `Video_Snapshot`.

Titulo	Numero_de_likes	Numero_de_visualizacoes	Descricao	Numero_de_comentarios	Data	Video_ID
WE'VE GOT NEWS!	22466	424433	Order Jinger's new book People Pleaser: https:/...	1213	2024-10-19	_4qLcHbQbbQ
爆弾のタイマー長すぎて緊張感なくなった【アニメ】...	14569	770006	【マリマリマリーカオスカフェ】コントに出てきたカフェが...	355	2024-10-19	_9FkcwZmDEs
«Легкий способ бросить курить»	205678	2453108		856	2024-10-19	_9HkgCaghH4
что мне подарили?! шок 🐼 продолжение в т...	15194	298571		268	2024-10-19	_9prkd0pzaA
Hosts take 3 points in top match | Korea Republi...	8953	654156	Enjoy the highlights of the match between Kore...	1354	2024-10-19	_AZTn6_Kv-s
FUI DOMAR UM TIRANOSSAURO REX E ENCON...	101439	1596806	Hoje chegou o grande de dia de domar o rei dos...	7429	2024-10-19	_CfF7C9BNN4
would you eat this? #shorts	117737	1916049	#shorts #mukbang #asmr #eating #asmreatin...	490	2024-10-19	_efahxEBeW8
【報告】7年付き合ってきた彼氏と別れました。	26219	644666	いつも見てくれてありがとうございます(○__○)/○...	2951	2024-10-19	_fL-_NutP9k
全面抗争ドッキリでガチで〇〇発生！？10人ニ...	2727	793833	https://abe.ma/3TXq5Sa続きはこちらから♪※この...	383	2024-10-19	_ItQuhtRQ0w
The Legend of Ochi | Official Trailer HD | A24	14332	462096	SUBSCRIBE: http://bit.ly/A24subscribeFrom wri...	1182	2024-10-19	_JTFLg3arYU
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figura 3.21: Exemplo de dados relativos a `Video_Snapshot`.

3.4 Consultas

3.4.1 Views Auxiliares

Antes de realizarmos algumas consultas, decidimos por criar algumas *views* que nos auxiliassem, sendo elas:

```
-- VISOES AUXILIARES:

-- IDs dos videos e suas ultimas datas de aparicao em alta de cada pais
CREATE VIEW Ultima_aparicao_video_pais AS
SELECT V.ID AS Video_ID, T.Pais_Codigo, MAX(VS.Data) AS Ultima_data
FROM Video AS V JOIN
    Video_Snapshot AS VS ON V.ID = VS.Video_ID JOIN
      Trend AS T ON V.ID = T.Video_ID AND
              VS.Data = T.Video_Snapshot_Data
GROUP BY V.ID, T.Pais_Codigo;

-- IDs dos videos e suas ultimas datas de aparicao em alta
CREATE VIEW Ultima_aparicao_video AS
SELECT Video_ID, MAX(Ultima_Data) AS Ultima_data
FROM Ultima_aparicao_video_pais
GROUP BY Video_ID;

-- IDs dos canais e suas ultimas datas de aparicao em alta de cada pais
CREATE VIEW Ultima_aparicao_canal_pais AS
SELECT C.ID AS Canal_ID, UAVP.Pais_Codigo, MAX(UAVP.Ultima_Data) AS Ultima_data
FROM Ultima_aparicao_video_pais AS UAVP JOIN
    Video AS V ON UAVP.Video_ID = V.ID JOIN
      Canal AS C ON V.Canal_ID = C.ID
GROUP BY C.ID, UAVP.Pais_Codigo;
```

3.4.2 Primeira Consulta

Essa consulta exibe, em ordem decrescente, os canais que mais apareceram em alta e a quantidade de vezes que isso aconteceu. Consideramos como aparição cada ocorrência de um vídeo do canal na seção de vídeos em alta de um país em uma data.

```
SELECT CS.Titulo AS Canal, COUNT(CS.Canal_ID) AS Numero_de_aparicoes
FROM Trend AS T JOIN
    Video_Snapshot AS VS ON T.Video_ID = VS.Video_ID AND
                T.Video_Snapshot_Data = VS.Data JOIN
      Video AS V ON VS.Video_ID = V.ID JOIN
        Canal AS C ON V.Canal_ID = C.ID JOIN
          Canal_Snapshot AS CS ON C.ID = CS.Canal_ID AND
                      VS.Data = CS.Data
GROUP BY CS.Titulo
ORDER BY Numero_de_aparicoes DESC;
```

	Canal	Numero_de_vezes_no_Trend
▶	Brawl Stars	173
	HYBE LABELS	149
	FORMULA 1	143
	JYP Entertainment	116
	BABYMONSTER	108
	NFL	100
	SMTOWN	86
	MrBeast	83
	Mnet K-POP	70
	Tyler, The Creator - Topic	70

Figura 3.22: Primeira consulta.

3.4.3 Segunda Consulta

A seguinte consulta exibe, em ordem decrescente, o número de vezes que uma tag apareceu em algum vídeo em alta. Consideramos como aparição cada ocorrência de uma tag em um vídeo na seção de vídeos em alta de um país em uma data.

```
SELECT Tag, COUNT(Tag) AS Numero_de_usos
FROM Ultima_aparicao_video_pais AS UAVP JOIN
    Tag AS T ON UAVP.Video_ID = T.Video_ID AND
          UAVP.Ultima_data = T.Video_Snapshot_Data
GROUP BY Tag
ORDER BY Numero_de_usos DESC;
```

	Tag	Numero_de_usos
▶	football	107
	Highlights	90
	futebol	75
	minecraft	72
	Sports	69
	Funny	63
	soccer	58
	vlog	58
	comedy	57
	melhores momentos	54

Figura 3.23: Segunda consulta.

3.4.4 Terceira Consulta

Essa consulta exibe, em ordem decrescente, o número de vezes que uma palavra-chave apareceu em algum canal em alta. Consideramos como aparição cada ocorrência de uma palavra-chave em um canal para cada vídeo seu em alta em um país em uma data.

```
SELECT Palavra_chave, COUNT(Palavra_chave) AS Numero_de_usos
FROM Ultima_aparicao_canal_pais AS UACP JOIN
    Palavra_chave AS PC ON UACP.Canal_ID = PC.Canal_ID AND
                  UACP.Ultima_data = PC.Canal_Snapshot_Data
GROUP BY Palavra_chave
ORDER BY Numero_de_usos DESC;
```

	Palavra_chave	Numero_de_usos
▶	de	130
	Vidéo	122
	the	119
	News	108
	Funny	85
	TV	83
	Game	83
	football	80
	music	76
	gaming	76

Figura 3.24: Terceira consulta.

3.4.5 Quarta Consulta

Essa consulta apresenta cada país e o número de vezes que um canal sediado apareceu em alta. Consideramos como aparição cada ocorrência de canal sediado no país para cada vídeo seu em alta em um país em uma data.

```
WITH
```

```
  Paises_em_alta AS
    (SELECT P.Codigo, CS.Canal_ID
         FROM Canal_Snapshot AS CS LEFT JOIN
        Pais AS P ON CS.Pais_Codigo = P.Codigo JOIN
          Video AS V USING (Canal_ID) JOIN
            Video_Snapshot AS VS ON V.ID = VS.Video_ID AND
                        CS.Data = VS.Data JOIN
              Trend AS T ON V.ID = T.Video_ID AND
                      VS.Data = T.Video_Snapshot_Data)
(SELECT Codigo AS Pais_sede, COUNT(Codigo) AS Numero_de_aparicoes
 FROM Paises_em_alta
 WHERE Codigo IS NOT NULL
 GROUP BY Codigo
 UNION
 SELECT NULL AS Pais_sede, COUNT(*) AS Numero_de_aparicoes
 FROM Paises_em_alta
 WHERE Codigo IS NULL)
ORDER BY Numero_de_aparicoes DESC;
```

Pais_sede	Numero_de_aparicoes
US	3813
NULL	3411
JP	3078
BR	2854
FR	2628
RU	642
GB	592
AU	373
KR	249
CA	205

Figura 3.25: Quarta consulta.

3.4.6 Quinta Consulta

Essa consulta exibe, em ordem decrescente de total de visualizações, os canais que mais tiveram visualizações em seus vídeos em alta durante o período de coleta.

```
WITH
  Video_mais_recente AS
    (SELECT V.ID, V.Canal_ID, VS.Numero_de_visualizacoes
         FROM Ultima_aparicao_video AS UAV JOIN
          Video AS V ON UAV.Video_ID = V.ID JOIN
            Video_Snapshot AS VS ON V.ID = VS.Video_ID AND
                        UAV.Ultima_data = VS.Data)
SELECT DISTINCT CS.Titulo AS Canal, VR.Total_de_visualizacoes
FROM (SELECT Canal_ID, SUM(Numero_de_visualizacoes) AS Total_de_visualizacoes
    FROM Video_mais_recente
      GROUP BY Canal_ID) AS VR JOIN
```

```
    Canal_Snapshot AS CS USING (Canal_ID)
ORDER BY VR.Total_de_visualizacoes DESC;
```

	Canal	Total_de_visualizacoes
▶	MrBeast	778178743
	Nam Phương	392319202
	La La Learn	308866691
	火影忍者一家	298864820
	ROSÉ	279308430
	Leisi Crazy	245697676
	Stokes Twins	219977597
	路飞与唐舞桐	210789754
	Justin Flom	205547295
	Daniel LaBelle	198846730

Figura 3.26: Quinta consulta.

3.4.7 Consultas Adicionais

Ainda que as consultas acima atendam, em conjunto, aos requisitos especificados pela proposta do trabalho, para expandir nossas possibilidades de análise sobre os dados obtidos, foram implementadas algumas outras consultas na aplicação. Em particular, foram incluídas consultas que, de acordo com a entrada do usuário, retornam dados específicos ao canal, país, ou categoria selecionada. Apesar de não encontradas detalhadas neste relatório, a aplicação *Web* também implementa as seguintes consultas:

- Vídeos em alta de um canal;
- Vídeos em alta de um país;
- Tags com mais visualizações;
- Views por categoria;
- Likes por categoria;
- Aparições em alta por categoria;
- Categorias dos vídeos em alta por canal;
- Aparições de um canal por país;
- Canais que já apareceram em alta.

Dessa maneira, foi implementado um total de 14 consultas distintas sobre os dados.

Capítulo 4

Aplicação

A aplicação desenvolvida consiste de três componentes principais: o *front-end* construído com `HTML` e `JavaScript`, o back-end escrito em `Python`, e o servidor de banco de dados `MySQL`. O *back-end* em `Python` utiliza a biblioteca `Flask` para estabelecer o servidor *Web*, que serve inicialmente alguns arquivos estáticos e aguarda a escolha de uma consulta, cujos dados serão exibidos. A página inicial exibe um menu *dropdown* em que se pode escolher entre cada uma das consultas possíveis à base de dados. Dependendo da consulta, o *site* então pergunta por dados adicionais. Por exemplo, qual canal consultar, ou sobre qual data se deseja obter informações. Para essas consultas, a *query* padrão será completada através da definição de algumas *user-defined variables*, para que seja específica aos parâmetros selecionados. Há então a consulta ao banco de dados através da biblioteca `mysql` para `Python`, e os dados serão retornados ao *front-end* em arquivos `HTML`, populados por meio dos mecanismos de *templates* do `Flask`. Para gerar gráficos, o *back-end* utiliza a biblioteca `matplotlib`.

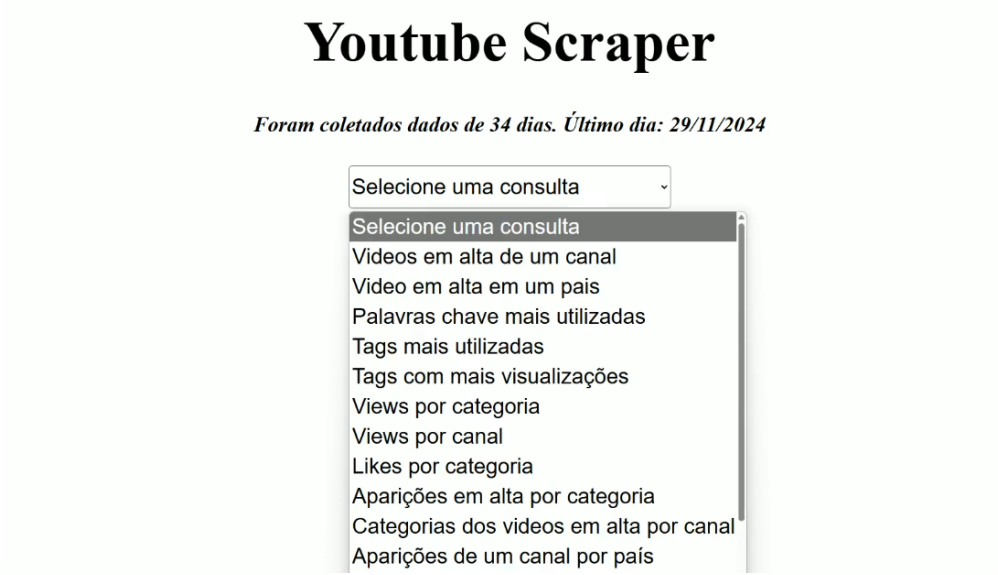


Figura 4.1: Seleção de consulta.

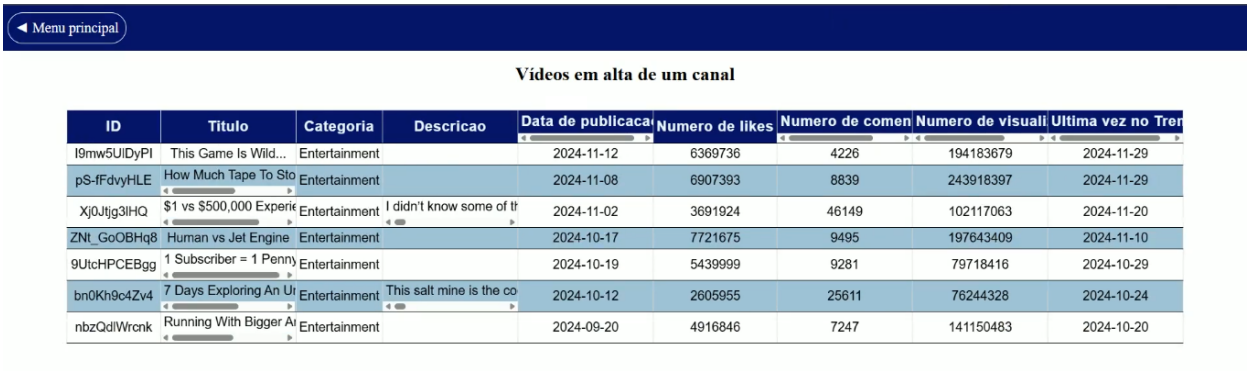


Figura 4.2: Dados em tabela.

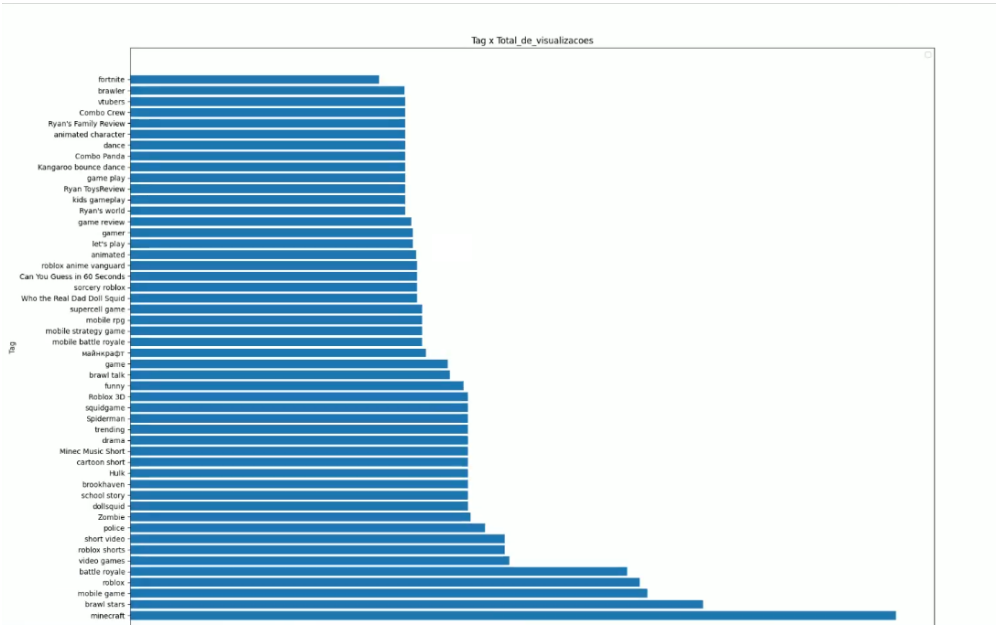


Figura 4.3: Dados em gráfico.

Capítulo 5

Distribuição do Trabalho

Inicialmente, os membros do grupo se reuniram para debater a modelagem conceitual para o tema proposto e, após diversos protótipos, foi encontrada uma solução plausível para a modelagem. Em seguida, João Victor tratou da implementação do modelo conceitual no `brModelo`.

No entanto, após a rodada inicial de apresentações, se tornou evidente por meio do *feedback* obtido que a modelagem conceitual apresentada não considerava adequadamente certas particularidades sobre a temporalidade dos dados. João Victor, William Victor e Gustavo se reuniram novamente em busca de uma modelagem alternativa, e, após algumas propostas analisadas, Gustavo sugeriu a divisão das entidades utilizando o conceito de *Snapshot* para separar dados temporais de dados atemporais (fixos para uma certa entidade ao longo do tempo), o que permitiu uma modelagem coerente com a complexidade dos dados.

Então, João Victor, junto de William Victor, realizaram a conversão para o modelo lógico e, em seguida, para o modelo físico. De forma quase assíncrona, Yuri e William Victor implementaram o programa que extraísse e tratasse os dados do *YouTube*.

A idealização das consultas que seriam realizadas, assim como a sua implementação utilizando *queries* `MySQL`, foi realizada por William Victor. Também foi sua contribuição o estabelecimento de *views* básicas, a partir das quais *queries* mais complexas poderiam ser construídas, minimizando a complexidade individual de cada consulta.

A respeito da implementação *Web*, em primeira instância, o grupo se reuniu para discutir os detalhes da aplicação e decidir pequenos detalhes. Inicialmente, havia a ideia de pré-computar as consultas necessárias, de modo a simplificar grandemente a implementação da aplicação, que poderia ser estática, requisitando arquivos *CSV* com os dados de um repositório no *GitHub*. No entanto, ao compreender que isso inviabilizaria o desenvolvimento de consultas dinâmicas, e possivelmente não se adequaria aos requisitos do trabalho, o grupo decidiu rever a arquitetura da aplicação.

O uso da ferramenta `Flask` foi indicado por Gustavo, de modo inspirado por arquivos de instruções auxiliares disponibilizados no mural da disciplina no *Google Classroom*. Gustavo auxiliou então Yuri na especificação da arquitetura da aplicação *Web*, determinando as maneiras que o *front-end* se comunicaria com o *back-end* e, por sua vez, como o *back-end* se comunicaria com o banco de dados. Os códigos `HTML`, `CSS` e `JavaScript` foram escritos por Yuri.

Sobre esse documento: os capítulos 1 e 2 foram escritos por William Victor e Yuri. O capítulo 3 foi escrito por João Victor. O capítulo 4 foi escrito por Gustavo, e os capítulos 5 e 6 escritos

inicialmente por João Victor, sendo depois revistos e estendidos por Gustavo.

Todos os capítulos desse documento sofreram atualizações conforme o desenvolvimento do trabalho. A confecção, padronização e formatação do documento foi feita por João Victor e Gustavo.

Capítulo 6

Considerações Finais

Este trabalho, apresentado como projeto final de disciplina de Banco de Dados, criou um contexto interessante no qual aplicar aquilo que estudamos ao longo de um semestre e desenvolver habilidades complementares. Descobrimos como interagir com a *API* do *YouTube*, escrevemos *scripts* para consulta, executados diariamente por um período de aproximadamente um mês para a coleta dos dados, selecionamos e tratamos as informações, populamos um banco de dados `MySQL` e escrevemos um servidor para uma aplicação *Web* que acessa e exibe as estatísticas coletadas.

Cada uma das etapas acima trouxe impedimentos de natureza conceitual e de implementação, que precisaram ser superados pelo grupo. Levando em conta que nenhum dos seus integrantes havia experiência prévia com bancos de dados ou desenvolvimento *Web*, e, visto que conseguimos de forma satisfatória realizar a coleta de dados e, afinal, efetuar consultas dinâmicas ao banco por meio da aplicação, concluímos que o resultado alcançado atendeu às expectativas.

Em resumo, dado o objetivo inicial do projeto, concordamos que o resultado alcançado foi um sucesso.

Bibliografia

[1] Google Developers. *YouTube Data API v3 - Channels*. Acessado em 25 de outubro de 2024. 2024. URL: https://developers.google.com/youtube/v3/docs/channels?hl=pt-br.

[2] Google Developers. *YouTube Data API v3 - Channels - Snippet Country*. Acessado em 25 de outubro de 2024. 2024. URL: https://developers.google.com/youtube/v3/docs/channels.

[3] Alphabet Inc. *YouTube*. Acessado em 11 de outubro de 2024. 2008. URL: https://www.youtube.com.

[4] Google LLC. *Kaggle*. Acessado em 12 de outubro de 2024. 2010. URL: https://www.kaggle.com/datasets/datasnaek/youtube-new.

[5] Gustavo de Mendonça Freire, João Victor Lopez Pereira, William Victor Quintela Paixão e Yuri Rocha de Albuquerque. *Youtube-Scraper-DataBank*. Repository dedicated to the final project of the "Databases 1" course at the Federal University of Rio de Janeiro (UFRJ). 2024. URL: https://github.com/joaovictorlopezpereira/Youtube-Scraper-DataBank.

[6] International Organization for Standardization. *ISO 3166 Country Codes*. Acessado em 24 de outubro de 2024. 2024. URL: https://www.iso.org/iso-3166-country-codes.html.

[7] Jason Urgo. *Social Blade*. Acessado em 11 de outubro de 2024. 2008. URL: https://socialblade.com/.

[8] Mitchell J. Y. *Trending YouTube Scraper*. Acessado em 28 de setembro de 2024. 2018. URL: https://github.com/mitchelljy/Trending-YouTube-Scraper/tree/master.