Database Project - YouTube

Created by Davide Picello and Mahdi Mouad

🇬🇧 *English version, automatically translated by Notion, below*

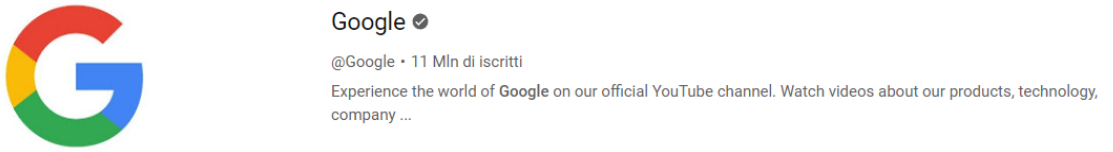
🇮🇹Abstract

YouTube è una piattaforma di condivisione video online che permette agli utenti di interagire con una vasta gamma di contenuti video. Sul sito è possibile vedere videoclip, trailer, cortometraggi, notizie, live streaming e altri contenuti come video blog, brevi video originali, video didattici e altro ancora, classificabili anche per età, con gli utenti che possono anche interagire votando, commentando, aggiungendo ai preferiti e, nei casi ove necessario, segnalando i video.

La maggioranza dei suoi video sono liberi per visualizzazione, ma ci sono eccezioni, inclusi i video caricati in modalità "premium" basati su un abbonamento solitamente mensile noleggio film, nonché YouTube Premium.

Analisi dei requisiti

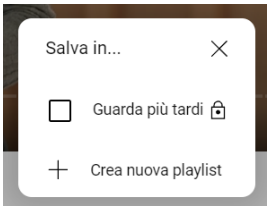
Il progetto prevede una base di dati che raccoglie informazioni sui **video (o live) con** dettagli come titolo, descrizione, data di pubblicazione, numero di visualizzazioni, **likes** e dislikes.
Ogni video è associato a un canale, che rappresenta **l'account** dell'utente che ha caricato il video.



Esempio di un canale youtube

Gli **utenti** possono interagire con i video attraverso azioni come il like, il dislike, i commenti e l'aggiunta ai preferiti. Il progetto prevede anche una tabella per gestire le informazioni sui like e i dislike ricevuti da ciascun video, consentendo di calcolare il rating complessivo di un video basato sulla somma di queste valutazioni.

Bottoni di like e dislike

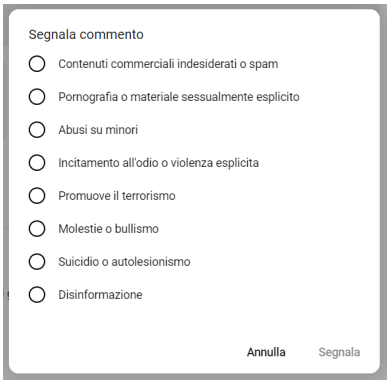


Finestra per salvare un video in una playlist

Inoltre, il progetto include la gestione delle **playlist**, che consentono agli utenti di creare elenchi personalizzati di video. Ogni playlist è associata a un account utente (il creatore) e può contenere una serie di video.

Per tenere traccia delle visualizzazioni dei video, è stata creata una tabella "**Views**" che registra l'account che ha visualizzato il video, l'ID del video e la data della visualizzazione.

Oltre alle interazioni degli utenti, il progetto prevede anche la gestione delle **segnalazioni** dei video e dei commenti.



Finestra per segnalare un commento

Sono presenti due tabelle separate per registrare le segnalazioni dei video e dei commenti, con informazioni come l'account che ha effettuato la segnalazione, l'ID

del video o del commento segnalato, il motivo della segnalazione e la data della segnalazione.

Infine, sono presenti le tabelle per gestire i **like** ai video e ai commenti. Ogni like è associato a un account utente, all'ID del video o del commento e alla data in cui è stato effettuato.
Queste informazioni consentono di calcolare il rating complessivo di un video o di un commento basato sulla somma dei like e dei dislike.

In sintesi, il progetto di database per YouTube mira a gestire le informazioni relative ai video, agli utenti e alle interazioni tra questi fornendo delle statistiche su di esse.

Glossario dei termini

Termine	Descrizione	Collegamenti
Account	Rappresenta un utente registrato su YouTube con informazioni come nome, email e data di registrazione. Un utente produce e/o interagisce con i contenuti della piattaforma	
Video	Rappresenta un video o una live caricato su YouTube, con attributi come il titolo, la descrizione, la durata e la data di pubblicazione.	Account
Views	Registra le visualizzazioni di un video, con informazioni sull'account che ha guardato il video, l'ID del video e la data di visualizzazione.	Account, Video

Termine	Descrizione	Collegamenti
Abbonamenti	Rappresenta la relazione di abbonamento tra due canali su YouTube. Registra gli abbonamenti degli utenti a determinati canali, con informazioni sull'account che effettua l'abbonamento e il canale a cui si è abbonato.	Account(canale), Account(iscritto)
Playlist	Rappresenta una raccolta di video organizzati da un utente, con attributi come il titolo, la descrizione e l'account creatore della playlist.	Account, Video
Commenti	Rappresenta i commenti lasciati dagli utenti su un video, con attributi come il messaggio del commento, l'account che ha commentato, l'ID del video e la data del commento.	Account, Video, Commenti(in caso di risposta ad un altro commento)
SavedPlaylist	Associa gli account agli ID delle playlist salvate, permettendo agli utenti di salvare playlist di loro interesse.	Account, Playlist
SegnalazioniCommenti	Registra le segnalazioni fatte dagli utenti su un commento, con informazioni sull'account che ha effettuato la segnalazione, l'ID del commento segnalato e il motivo della segnalazione.	Account, Commenti
SegnalazioniVideo	Registra le segnalazioni fatte dagli utenti su un	Account, Video

Termine	Descrizione	Collegamenti
	video, con informazioni sull'account che ha effettuato la segnalazione, l'ID del video segnalato e il motivo della segnalazione.	
LikeVideo	Registra i like e i dislike dati dagli utenti a un video, con informazioni sull'account che ha espresso il like/dislike, l'ID del video e la data dell'azione.	Account, Video
LikeCommenti	Registra i like e i dislike dati dagli utenti a un commento, con informazioni sull'account che ha espresso il like/dislike, l'ID del commento e la data dell'azione.	Account, Commenti

Operazioni tipiche

- Operazioni tipiche di un server YouTube in una giornata (*stime approssimative: le frequenze effettive possono variare in base a molti fattori*)

Caricamento e processamento di nuovi video	Migliaia o addirittura milioni di volte al giorno, a seconda della quantità di contenuti caricati dagli utenti.
Visualizzazioni e conteggio di like/dislike	Milioni o miliardi di volte al giorno (poiché i video vengono visualizzati da un vasto numero di utenti).
Gestione degli abbonamenti e delle notifiche	Migliaia o milioni di volte al giorno
Elaborazione delle segnalazioni e delle richieste di rimozione dei contenuti	Migliaia o decine di migliaia di volte al giorno
Calcolo delle statistiche di visualizzazione e interazione:	Costantemente in tempo reale, mentre gli utenti interagiscono con i video.

Fornitura di suggerimenti e raccomandazioni personalizzate	Continuamente in tempo reale.
Monitoraggio e mitigazione degli abusi, spam e attività fraudolente	Costantemente in tempo reale, poiché YouTube deve individuare e affrontare rapidamente tali comportamenti indesiderati.
Archiviazione e gestione dei dati	Costantemente in tempo reale
Manutenzione del sistema e risoluzione dei problemi tecnici	Periodicamente o in risposta a problemi o necessità specifiche del sistema.

Progettazione Concettuale

Lista entità

Account

Nome del campo	Tipo di dato + vincoli	Descrizione
id_Account	SERIAL	Codice numerico univoco per ogni account
handle	varchar(256) NOT NULL UNIQUE	Nome unico per ogni utente, modificabile dall'utente
mail	varchar(256) NOT NULL UNIQUE	Indirizzo email dell'account
password	varchar(256) NOT NULL	Password dell'account
dataIscrizione	date NOT NULL	Data di iscrizione dell'account
imgProfilo	varchar(256)	Link all'immagine del profilo
nome_Utente	varchar(256) NOT NULL	Nome dell'utente
cognome_Utente	varchar(256) NOT NULL	Cognome dell'utente
compleanno	timestamp	Data di compleanno dell'utente
genere	Gender NOT NULL	Genere dell'utente
paese	varchar(256) NOT NULL	Paese dell'utente

Nome del campo	Tipo di dato + vincoli	Descrizione
StatoAccount	Stato NOT NULL	Stato dell'account: descrive se l’account è sospeso, attivo o eliminato(quando si elimina un account da interfaccia utente non viene del tutto eliminato dal database)
descrizione	varchar(1000)	Descrizione dell'account, scritta dall’utente
premium	boolean NOT NULL	Indica se attivato l’abbonamento a YouTube premium

Vincoli:

- PRIMARY KEY: id_Account

Video

Nome del campo	Tipo di dato + vincoli	Descrizione
id_Video	SERIAL	ID unico per ogni video
titolo	varchar(256) NOT NULL	Titolo del video
descrizione	varchar(500)	Descrizione del video
dataPubblicazione	timestamp NOT NULL	Data di pubblicazione del video
durata	INT NOT NULL	Durata del video
costo	float	Costo del video (0 se gratuito)
categoria	Categorie	Categoria del video
visibilita	Visibilita DEFAULT 'Pubblico'	Visibilità del video
stato	Stato NOT NULL	Stato del video
id_Account	INT NOT NULL	ID dell'account che ha caricato il video

Nome del campo	Tipo di dato + vincoli	Descrizione
isLive	boolean	Flag che indica se il video è una live in corso (0 → no, 1 → si)
dataFine	timestamp	Nel caso fosse una live, questo è il momento in cui finisce
thumbnail	varchar(256)	Link all'immagine di anteprima del video

Vincoli:

- PRIMARY KEY: id_Video
- FOREIGN KEY: id_Account REFERENCES Account(id_Account) ON UPDATE CASCADE ON DELETE CASCADE
- CHECK: (dataFine IS NULL AND isLive = false) OR (dataFine IS NOT NULL AND isLive = true)

Abbonamenti

Nome del campo	**Tipo di dato + vincoli**	**Descrizione**
canale	INT	ID del canale a cui ci si abbona
iscritto	INT	ID del canale di colui che si abbona
livello	Abbonamento NOT NULL	Livello di abbonamento. Se abbonamento = 'Gratis', solo iscrizione
dataIscrizione	timestamp NOT NULL	Data di iscrizione all'abbonamento

Vincoli:

- PRIMARY KEY: (canale, iscritto)

- FOREIGN KEY: canale REFERENCES Account(id_Account) ON UPDATE CASCADE ON DELETE CASCADE
- FOREIGN KEY: iscritto REFERENCES Account(id_Account) ON UPDATE CASCADE ON DELETE CASCADE
- CHECK: (iscritto <> canale) -- Controlla che uno non si iscriva a se stesso

Views

Nome del campo	Tipo di dato + vincoli	Descrizione
account	INT NOT NULL	ID dell'account che sta guardando il video
id_Video	INT NOT NULL	ID del video che viene visualizzato
dataView	timestamp NOT NULL	Data e ora in cui è avvenuta la visualizzazione

Vincoli:

- PRIMARY KEY: (account, id_Video)
- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE
- FOREIGN KEY: id_Video REFERENCES Video(id_Video) ON DELETE CASCADE ON UPDATE CASCADE

Playlist

Nome del campo	Tipo di dato + vincoli	Descrizione
id_Playlist	SERIAL	ID univoco della playlist
account	INT NOT NULL	ID dell'account che ha creato la playlist
titolo	varchar(256) NOT NULL DEFAULT 'Guarda più tardi’	Titolo della playlist

Nome del campo	Tipo di dato + vincoli	Descrizione
descrizione	varchar(500)	Descrizione della playlist
visibilità	Visibilita DEFAULT 'Privato'	Visibilità della playlist, di default impostata a 'Privato'

Vincoli:

- PRIMARY KEY: id_Playlist
- UNIQUE: (titolo, account)
- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE

VideoPlaylist

Nome del campo	Tipo di dato + vincoli	Descrizione
id_Video	INT NOT NULL	ID del video salvato
id_Playlist	INT NOT NULL	ID della playlist in cui si è salvato il video

- PRIMARY KEY: (id_Video, id_Playlist),
- FOREIGN KEY(id_Video) REFERENCES Video(id_Video) ON DELETE CASCADE ON UPDATE CASCADE,
- FOREIGN KEY(id_Playlist) REFERENCES Playlist(id_Playlist) ON DELETE CASCADE ON UPDATE CASCADE

Savedplaylist

Nome del campo	Tipo di dato + vincoli	Descrizione
account	INT NOT NULL	Account che salva la playlist
id_Playlist	INT NOT NULL	ID della playlist salvata

La definizione della tabella include i seguenti vincoli:

- PRIMARY KEY: (account, id_Playlist)

- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE
- FOREIGN KEY: id_Playlist REFERENCES Playlist(id_Playlist) ON DELETE CASCADE ON UPDATE CASCADE

Commenti

Nome del campo	Tipo di dato + vincoli	Descrizione
id_Commento	SERIAL	Codice univoco per ogni commento
account	INT NOT NULL	Chi commenta
id_Video	INT NOT NULL	Che video commenta
messaggio	varchar(500) NOT NULL	Testo del commento
donazione	float	Campo per eventuali donazioni
dataCommento	timestamp NOT NULL	Data e ora del commento
id_Risposta	INT	Eventuale ID di un altro commento al quale si risponde

La definizione della tabella include i seguenti vincoli:

- PRIMARY KEY: id_Commento
- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE
- FOREIGN KEY: id_Video REFERENCES Video(id_Video) ON DELETE CASCADE ON UPDATE CASCADE
- FOREIGN KEY: id_Risposta REFERENCES Commenti(id_Commento) ON DELETE CASCADE ON UPDATE CASCADE
- CHECK: donazione >= 0

SegnalazioniVideo

Nome del campo	Tipo di dato + vincoli	Descrizione
id_Segnalazione	SERIAL	Codice univoco per ogni segnalazione
account	INT NOT NULL	Chi segnala
id_Video	INT NOT NULL	Video segnalato
motivo	Motivo	Motivo della segnalazione
descrizione	varchar(50)	Descrizione della segnalazione
data	timestamp NOT NULL	Data di segnalazione

Le chiavi primarie e le chiavi esterne sono mantenute come nell'originale:

- PRIMARY KEY: id_Segnalazione
- FOREIGN KEY: account REFERENCES (id_Account) ON DELETE CASCADE ON UPDATE CASCADE
- FOREIGN KEY: id_Video REFERENCES (id_Video) ON DELETE CASCADE ON UPDATE CASCADE

SegnalazioniCommenti

Nome del campo	Tipo di dato + vincoli	Descrizione
id_Segnalazione	SERIAL	Codice univoco per ogni segnalazione
account	INT NOT NULL	Chi segnala
id_Commento	INT NOT NULL	Commento segnalato
motivo	Motivo	Motivo della segnalazione
data	timestamp NOT NULL	Data di segnalazione

Le chiavi primarie e le chiavi esterne sono mantenute come nell'originale:

- PRIMARY KEY: id_Segnalazione
- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE

- FOREIGN KEY: id_Commento REFERENCES Commenti(id_Commento) ON DELETE CASCADE ON UPDATE CASCADE

LikeVideo

Nome del campo	Tipo di dato + vincoli	Descrizione
id_Like	SERIAL	Codice univoco per ogni like ai video
account	INT NOT NULL	Account che ha messo il like/dislike
id_Video	INT NOT NULL	Video a cui è stato messo il like/dislike
data	timestamp NOT NULL	Momento dell'assegnazione del like/dislike
valutation	Likes NOT NULL	Valutazione del like/dislike (-1 = dislike, 1 = like)

- PRIMARY KEY: id_Like
- FOREIGN KEY: account REFERENCES Account(id_Account) ON UPDATE CASCADE ON DELETE CASCADE
- FOREIGN KEY: id_Video REFERENCES Video(id_Video) ON UPDATE CASCADE ON DELETE CASCADE

LikeCommenti

Nome del campo	Tipo di dato + vincoli	Descrizione
id_Like	SERIAL	Codice univoco per ogni like ai commenti
account	INT NOT NULL	Account che ha messo il like/dislike
id_Commento	INT NOT NULL	Commento a cui è stato messo il like/dislike

Nome del campo	Tipo di dato + vincoli	Descrizione
data	timestamp NOT NULL	Momento dell'assegnazione del like/dislike
valutation	Likes NOT NULL	Valutazione del like/dislike (-1 = dislike, 1 = like)

- PRIMARY KEY: id_Like
- FOREIGN KEY: account REFERENCES Account(id_Account) ON UPDATE CASCADE ON DELETE CASCADE
- FOREIGN KEY: id_Commento REFERENCES Commenti(id_Commento) ON UPDATE CASCADE ON DELETE CASCADE

Lista delle Relazioni

Account-Video:

- un video appartiene ad account (1,1)
- un account ha diversi video (1,N)

Account-Playlist

- una playlist è creata da un solo account (1,1)
- un account ha una o più playlists associate(almeno 2 "mi piace" e "guarda più tardi") (2,N)

Views-Video

- un video ha diverse views (0,N)
- una views appartiene ad un ed un solo video (1,1)

Account-Views

- un account ha diverse views associate ad esso (0,N)
- una views appartiene ad un unico canale (1,1)

Account-Commenti

- un account ha diversi commenti associate ad esso (0,N)

- un commento appartiene ad un unico canale (1,1)

Account-SavedPlaylist

- un account almeno 2 playlist salvate(“mi piace” e ”guarda più tardi”) (2:N)
- una playlist è salvata da diverse persone (1,N)

Account-Abbonamento

- una account ha diversi abbonati (0,N)
- un account è abbonato a diversi canali (0,N)

Account-LikeCommento

- un account può mettere like o dislike a commenti diversi(0,N)
- un like ad un commento appartiene ad un solo account (1,1)

Account-LikeVideo

- un account può mettere like o dislike a video diversi (0,N)
- un like ad un commento appartiene ad un solo account (1,1)

Playlist-SavedPlaylist

- una playlist può essere salvata da diversi utenti (1,N)
- un utente può avere diverse playlist salvate(almeno “mi piace” e “guarda più tardi” che gli appartengono) (2:N) -

Video-LikeVideo

- un video ha diversi associati ad esso (0,N)
- un like è associato ad un solo video (1,1)

Commento-LikeCommento

- un commento può avere diversi like (0,N)
- un like a commento è associato ad un solo commento (1,1)

Video-SegnalazioniVideo

- un video può ricevere diverse segnalazioni (0,N)
- una segnalazione si riferisce ad un solo video (1,1)

Commenti-SegnalazioniCommenti

- un commento può avere diverse segnalazioni (0,N)
- una segnalazione appartiene ad un unico commento (1,1)

Playlist-Video

- un video può appartenere a diverse playlists (0,N)
- una playlist ha diversi video (0,N)

Account-SegnalazioniCommenti

- un account può fare diverse segnalazioni a commenti (0,N)
- una segnalazione a commento è fatta da un solo account (1,1)

Account-SegnalazioniVideo

- un account può fare diverse segnalazioni a video (0,N)
- una segnalazione a video è fatta da un solo account (1,1)

VideoPlaylist-Video

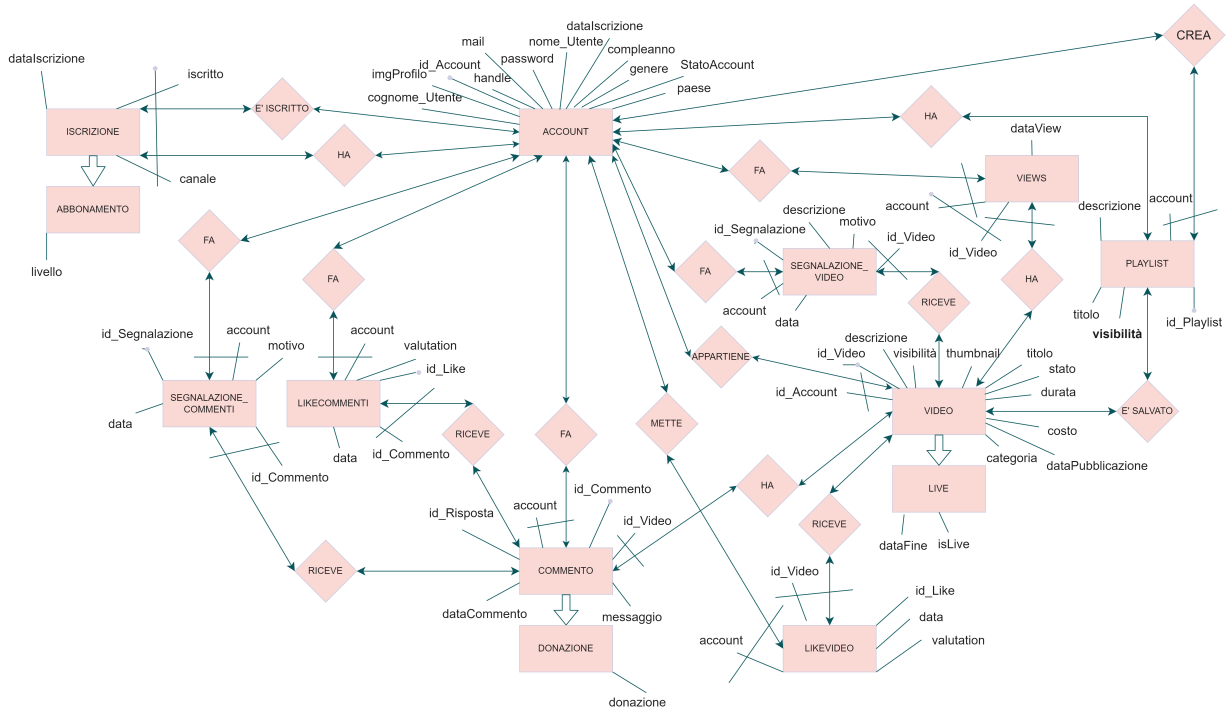
- un video può essere salvato in diverse playlist (0:N)
- una playlist può avere diversi video (1:N)

VideoPlaylist-Playlist

- Una playlist può essere composta da 1 o più video (1:N)
- Un video può essere salvato da 0 o più playlist (0:N)

Progettazione della base di dati

Schema concettuale



Analisi logica: gestione di ridondanze ed ereditarietà

Ridondanze

Nelle fasi iniziali della progettazione, questo progetto, aveva un problema di ridondanza.
Non esistevano infatti le tabelle
VideoPlaylist e **SavedPlaylist**.
Il che comportava che
Playlist avesse come attributi: id_Playlist, account, id_Video, titolo, descrizione, visibilita e account di chi la salvava.

Questo comportava gravi problemi di ridondanza in quanto, per ogni video salvato, avremmo salvato ogni volta, in modo ridondante, anche gli attributi: account, titolo, descrizione, visibilità → tutti attributi inerenti alla playlist e non ai video salvati.

Per risolvere questo problema, parte della tabella **Playlist** (la parte relativa ai video) è stata salvata in una nuova tabella, **VideoPlaylist** e **SavedPlaylist** ,che hanno il compito di mantenere le informazioni dei video salvati e di chi le salva, eliminando così la ridondanza.

VideoPlaylist e **SavedPlaylist** hanno il minimo indispensabile per poter sapere quale video è salvato in quale playlist e chi ha salvato quella playlist.

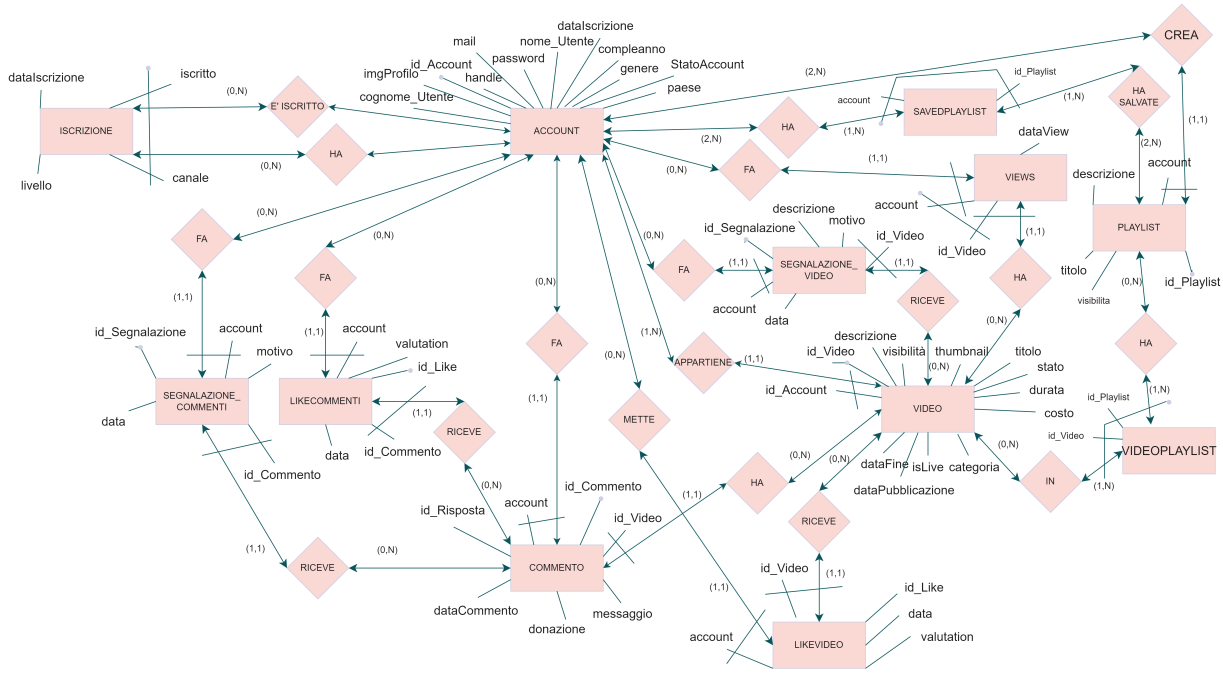
Ereditarietà (eliminazione delle generalizzazioni)

Le generalizzazioni sono state gestite incorporandole nelle classi principali.

Nello specifico:

- **Iscrizione** incorpora **abbonamento** includendo “livello” e impostandolo a 0 se non si è abbonati
- **Commento** incorpora **donazione** includendo “donazione” equivalente a 0 se non avviene.
- **Video** incorpora **live** includendo gli attributi “isLive” e “dataFine” rispettivamente impostati a 0 se la live è finita, o si tratta di un semplice video, ed a NULL se si tratta di un video: se invece si fosse trattato di una live terminata avremmo avuto la sua data di fine.

Diagramma E-R ristruttutato



Queries

Queries SQL

1) Creazione view per video e stampa delle visualizzazioni per i canali che hanno almeno 10 views:

```
CREATE VIEW ViewsPerVideo AS(
SELECT id_Video, COUNT(account) AS total_views
FROM Views
GROUP BY id_Video
);

SELECT Video.id_account, SUM(total_views) AS ViewsCanale
FROM ViewsPerVideo, Video
GROUP BY Video.id_account
HAVING SUM(total_views) > 10;
```

2) Query che crea le playlist guarda più tardi e video piaciuti per ogni utente:

```
INSERT INTO Playlist (account, titolo, descrizione, visibilita)
SELECT id_Account, 'Guarda più tardi', ' ', 'Privato'
FROM account;

INSERT INTO Playlist (account, titolo, descrizione, visibilita)
SELECT id_Account, 'Video piaciuti', 'descrizione', 'Privato'
FROM account;
```

3) Query che aggiunge video a cui ogni utente ha messo mi piace alla playlist video piaciuti:

```
INSERT INTO VideoPlaylist(id_Video,id_Playlist)
SELECT L.id_Video, P.id_Playlist
```

```
FROM Playlist AS P, LikeVideo AS L
WHERE L.valutation='1' AND L.account=P.account AND P.titolo='Vi
```

4) Video ordinati in base al rating ricevuto(like+dislike/views):

```
DROP VIEW IF EXISTS ViewsPerVideo;
DROP VIEW IF EXISTS SommaLike;
DROP VIEW IF EXISTS voto;

CREATE VIEW ViewsPerVideo AS(
SELECT id_Video, COUNT(account) AS total_views
FROM Views
GROUP BY id_Video);

CREATE VIEW voto AS(
SELECT id_Video,
    CASE
        WHEN valutation = '1' THEN 1.0
        ELSE -1.0
    END AS stato
FROM LikeVideo);

CREATE VIEW SommaLike AS(
SELECT id_Video, SUM(stato) AS SommaLike
FROM voto
GROUP BY id_Video);

SELECT S.id_Video, (S.SommaLike/V.total_views) AS RATING
FROM SommaLike AS S JOIN ViewsPerVideo AS V ON S.id_Video=V.id_V
ORDER BY RATING DESC;
```

5) Video in tendenza per views(top 10):

```
SELECT V.*
FROM Video AS V
JOIN (
  SELECT id_Video, COUNT(*) AS num_views
  FROM Views
  GROUP BY id_Video
) AS VIEWS ON V.id_Video = VIEWS.id_Video
ORDER BY VIEWS.num_views DESC
LIMIT 10;
```

6) Utenti a cui far vedere pubblicità:

```
SELECT *
FROM account
WHERE premium = FALSE;
```

7) Lista delle live in corso:

```
SELECT *
FROM Video
WHERE isLive = true;
```

8) Proiezione dei commenti: in questo caso una conversazione di commenti che rispondono al commento 1682

```
SELECT  *
FROM Commenti
WHERE id_commento = 1682 OR id_risposta = 1682
ORDER BY datacommento;
```

Queries parametriche in C++

Per l’implementazione delle query parametriche è stato sviluppato un software C++ che, tramite un piccolo menù, permette di scegliere una tra le seguenti queries.

Ognuna chiederà in input un parametro, tranne l’ultima che ne chiede due.

1. **Ricerca video**

   Questa query simula la ricerca di un determinato video .

   Selezionando questa opzione, verrà chiesto di inserire in input il nome di un video.
   Se esiste, ed è pubblico, verranno stampate alcune informazioni sul video e sul proprietario.

2. **Ricerca account**

   Questa query simula la ricerca di un determinato account.

   Selezionando questa opzione, verrà richiesto di inserire in input l'handle di un account.
   Se l'account esiste e il suo stato è "Attivo", saranno restituite diverse informazioni sull'account.

3. **Ricerca video per categoria**

   Selezionando questa opzione, verrà richiesto di inserire in input una categoria. Saranno restituiti i video pubblici appartenenti a quella categoria.

4. **Ricerca account per categoria**

   Selezionando questa opzione, verrà richiesto di inserire in input una categoria. Saranno restituite varie informazioni sugli account attivi correlati ai video appartenenti a quella categoria (basta avere un video per quella determinata categoria per apparire tra i risultati).

5. **Ricerca video per account**

   Questa query simula la ricerca di tutti i video appartenenti ad un determinato account

Selezionando questa opzione, verrà richiesto di inserire in input l'handle di un account.
Saranno restituite informazioni sui video pubblici dell'account corrispondente.

6. **Ricerca video per account e categoria**

   Questa query simula la ricerca di tutti i video appartenenti ad una determinata categoria, pubblicati da un determinato account.

   Selezionando questa opzione, verrà richiesto di inserire in input un handle di account e una categoria. Se esistono, verranno restituiti i video di quel account, appartenenti a quella determinata categoria.

Per lo sviluppo di tutte le queries è stato utilizzato un unico file: questo per rendere più agevole lo sviluppo, il testing, la compilazione e l'esecuzione del programma.

Per quanto riguarda l'esecuzione è necessario configurare correttamente i parametri PG_HOST, PG_USER, PG_DB, PG_PASS e PG_PORT alle righe 8-11 del codice.

Successivamente, è necessario compilare il programma con il comando: `g++ query.cpp -L dependencies\lib -lpq -o query` (avendo a disposizione le librerie adatte nella cartella dependencies)

L'esecuzione richiede il comando: `./query`

L'interazione con il programma, poi, è abbastanza intuitiva.

Indici

```
CREATE INDEX idx_likevideo_voto ON LikeVideo (valutation);

CREATE INDEX idx_video_titolo ON Video (titolo);

CREATE INDEX idx_views_account ON Views (account);
```

`idx_likevideo_voto` sulla tabella `LikeVideo`: Questo indice migliora il calcolo del rating dei video. Indicizza la colonna `valutation` nella tabella `LikeVideo`, consentendo un accesso più rapido ai dati relativi alle valutazioni (like/dislike) dei video.

`idx_video_titolo` sulla tabella `Video`: Questo indice è utile per le query parametriche, come la barra di ricerca. Indicizza la colonna `titolo` nella tabella `Video`, consentendo una ricerca più veloce e ottimizzata dei video in base al titolo.

`idx_views_account` sulla tabella `Views`: Questo indice facilita la visualizzazione della cronologia e il calcolo delle visualizzazioni per video e canali. Indicizza la colonna `account` nella tabella `Views`, consentendo un accesso rapido alle visualizzazioni associate a un account specifico.

Popolamento database

Per il popolamento del database si è usato, per le tabelle più piccole, ChatGpt: molto utile per generare dati di fantasia e sempre diversi ma con grandi limiti riguardo a quantità di risultati prodotti e mantenimento dei vari vincoli tra le varie tabelle (per esempio è capitato che generasse commenti con una data antecedente alla data di pubblicazione del video). Tutti gli errori di questo tipo sono dovuti essere verificati e sistemati manualmente, causando molte perdite di tempo.

Per le tabelle più grandi invece sono stati realizzati dei software C++ che le popolassero.
Questo metodo richiede sicuramente più sforzo e tempo per essere realizzato, ma garantisce dei risultati di altissima qualità ed affidabilità rispetto ai risultati forniti da ChatGpt.

Nella cartella “Popolamento”, sono allegati questi file.

🇬🇧Abstract

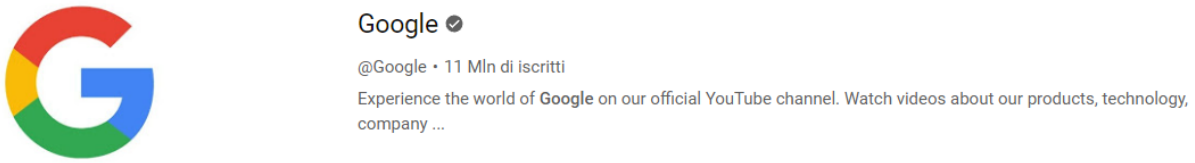
YouTube is an online video sharing platform that allows users to interact with a wide range of video content. On the site, it is possible to watch video clips,

trailers, short films, news, live streaming, and other content such as video blogs, short original videos, and educational videos. It is also possible to classify content by age, with users also able to interact by voting, commenting, adding to favourites, and, where necessary, reporting videos.

The majority of its videos are free to view, but there are exceptions, including videos uploaded in "premium" mode based on a usually monthly subscription, film rentals, as well as YouTube Premium.

Requirements Analysis

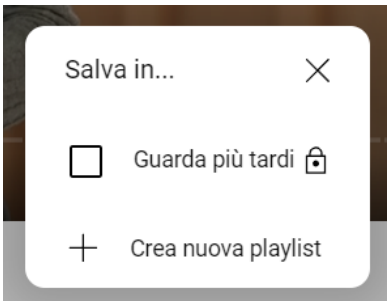
The project involves a database that collects information on **videos (or live broadcasts)** with details such as title, description, publication date, number of views, **likes** and dislikes.
Each video is associated with a channel, which represents
the account of the user who uploaded the video.



Example of a YouTube channel

Users can interact with videos through actions such as liking, disliking, commenting and adding to favourites. The project also includes a table to manage information on the likes and dislikes received by each video, allowing the overall rating of a video based on the sum of these ratings to be calculated.

Like and dislike buttons

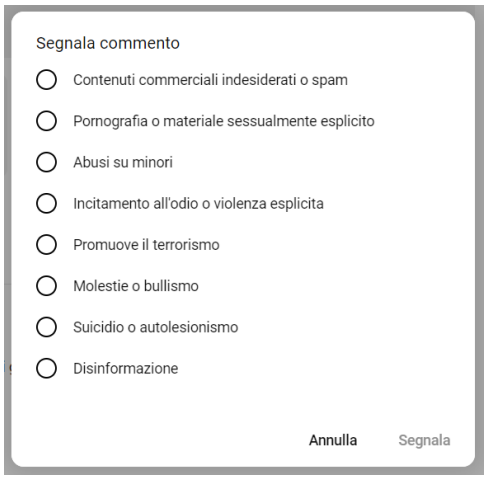


Window to save a video to a playlist

Moreover, the project includes the management of **playlists**, which allow users to create custom lists of videos. Each playlist is associated with a user account (the creator) and can contain a series of videos.

To keep track of video views, a "**Views**" table has been created that records the account that viewed the video, the ID of the video, and the date of the view.

In addition to user interactions, the project also includes the management of **video and comment reports**.



Window to report a comment

There are two separate tables to record the reports of videos and comments, with information such as the account that made the report, the ID of the reported video or comment, the reason for the report, and the date of the report.

Finally, there are tables to manage the **likes** to videos and comments. Each like is associated with a user account, the ID of the video or comment, and the date on which it was made.

This information allows the overall rating of a video or comment to be calculated based on the sum of likes and dislikes.

In summary, the YouTube database project aims to manage information related to videos, users and interactions between them by providing statistics on them.

Glossary of Terms

Term	Description	Links
Account	Represents a registered user on YouTube with information such as name, email, and registration date.	
A user produces and/or interacts with the platform's content		
Video	Represents a video or a live broadcast uploaded to YouTube, with attributes such as the title, description, duration, and publication date.	Account
Views	Records the views of a video, with information on the account that watched the video, the video ID, and the viewing date.	Account, Video
Subscriptions	Represents the subscription relationship between two channels on YouTube. Records user subscriptions to certain channels, with information on the account making the subscription and the channel to which it is subscribed.	Account(channel), Account(subscriber)

Term	Description	Links
Playlist	Represents a collection of videos organised by a user, with attributes such as the title, description, and the account creator of the playlist.	Account, Video
Comments	Represents the comments left by users on a video, with attributes such as the comment message, the account that commented, the video ID and the comment date.	Account, Video, Comments(in case of reply to another comment)
SavedPlaylist	Associates accounts with the IDs of saved playlists, allowing users to save playlists of their interest.	Account, Playlist
CommentReports	Records the reports made by users on a comment, with information on the account that made the report, the ID of the reported comment, and the reason for the report.	Account, Comments
VideoReports	Records the reports made by users on a video, with information on the account that made the report, the ID of the reported video, and the reason for the report.	Account, Video
VideoLikes	Records the likes and dislikes given by users to a video, with information on the account that expressed the like/dislike, the video ID, and the date of the action.	Account, Video

Term	Description	Links
CommentLikes	Records the likes and dislikes given by users to a comment, with information on the account that expressed the like/dislike, the comment ID, and the date of the action.	Account, Comments

Typical Operations

- Typical operations of a YouTube server in a day (*approximate estimates: actual frequencies may vary based on many factors*)

Uploading and processing of new videos	Thousands or even millions of times a day, depending on the amount of content uploaded by users.
Views and counting of likes/dislikes	Millions or billions of times a day (as videos are viewed by a large number of users).
Managing subscriptions and notifications	Thousands or millions of times a day
Processing of reports and content removal requests	Thousands or tens of thousands of times a day
Calculation of viewing and interaction statistics:	Constantly in real time, as users interact with the videos.
Providing personalised suggestions and recommendations	Continuously in real time.
Monitoring and mitigation of abuse, spam and fraudulent activity	Constantly in real time, as YouTube needs to quickly identify and address such undesirable behaviour.
Storage and data management	Constantly in real time
System maintenance and troubleshooting	Periodically or in response to specific system problems or needs.

Conceptual Design

Entity List

Account

Field Name	Data Type + Constraints	Description
id_Account	SERIAL	Unique numeric code for each account
handle	varchar(256) NOT NULL UNIQUE	Unique name for each user, modifiable by the user
mail	varchar(256) NOT NULL UNIQUE	Email address of the account
password	varchar(256) NOT NULL	Password of the account
registrationDate	date NOT NULL	Account registration date
profileImage	varchar(256)	Link to the profile image
firstName	varchar(256) NOT NULL	User's first name
lastName	varchar(256) NOT NULL	User's last name
birthday	timestamp	User's birthday date
gender	Gender NOT NULL	User's gender
country	varchar(256) NOT NULL	User's country
accountStatus	Status NOT NULL	Account status: describes whether the account is suspended, active or deleted (when an account is deleted from the user interface it is not completely deleted from the database)
description	varchar(1000)	Account description, written by the user
premium	boolean NOT NULL	Indicates whether the YouTube premium subscription is active

Constraints:

- PRIMARY KEY: id_Account

Video

Field Name	Data Type + Constraints	Description
id_Video	SERIAL	Unique ID for each video
title	varchar(256) NOT NULL	Video title
description	varchar(500)	Video description
publicationDate	timestamp NOT NULL	Video publication date
duration	INT NOT NULL	Video duration
cost	float	Video cost (0 if free)
category	Categories	Video category
visibility	Visibility DEFAULT 'Public'	Video visibility
status	Status NOT NULL	Video status
id_Account	INT NOT NULL	ID of the account that uploaded the video
isLive	boolean	Flag indicating whether the video is a live broadcast in progress (0 → no, 1 → yes)
endDate	timestamp	If it is a live broadcast, this is the moment when it ends
thumbnail	varchar(256)	Link to the video preview image

Constraints:

- PRIMARY KEY: id_Video
- FOREIGN KEY: id_Account REFERENCES Account(id_Account) ON UPDATE CASCADE ON DELETE CASCADE
- CHECK: (endDate IS NULL AND isLive = false) OR (endDate IS NOT NULL AND isLive = true)

Subscriptions

Field Name	Data Type + Constraints	Description
channel	INT	ID of the channel to which one subscribes
subscriber	INT	ID of the channel of the subscriber
level	Subscription NOT NULL	Level of subscription. If subscription = 'Free', only subscription
subscriptionDate	timestamp NOT NULL	Subscription date

Constraints:

- PRIMARY KEY: (channel, subscriber)
- FOREIGN KEY: channel REFERENCES Account(id_Account) ON UPDATE CASCADE ON DELETE CASCADE
- FOREIGN KEY: subscriber REFERENCES Account(id_Account) ON UPDATE CASCADE ON DELETE CASCADE
- CHECK: (subscriber <> channel) -- Checks that one does not subscribe to himself

Views

Field Name	Data Type + Constraints	Description
account	INT NOT NULL	ID of the account that is watching the video
id_Video	INT NOT NULL	ID of the video being viewed
viewDate	timestamp NOT NULL	Date and time when the view occurred

Constraints:

- PRIMARY KEY: (account, id_Video)
- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE

- FOREIGN KEY: id_Video REFERENCES Video(id_Video) ON DELETE CASCADE ON UPDATE CASCADE

Playlist

Field Name	Data Type + Constraints	Description
id_Playlist	SERIAL	Unique ID of the playlist
account	INT NOT NULL	ID of the account that created the playlist
title	varchar(256) NOT NULL DEFAULT 'Watch later’	Playlist title
description	varchar(500)	Playlist description
visibility	Visibility DEFAULT 'Private'	Playlist visibility, by default set to 'Private'

Constraints:

- PRIMARY KEY: id_Playlist
- UNIQUE: (title, account)
- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE

VideoPlaylist

Field Name	Data Type + Constraints	Description
id_Video	INT NOT NULL	ID of the saved video
id_Playlist	INT NOT NULL	ID of the playlist where the video was saved

- PRIMARY KEY: (id_Video, id_Playlist),
- FOREIGN KEY(id_Video) REFERENCES Video(id_Video) ON DELETE CASCADE ON UPDATE CASCADE,
- FOREIGN KEY(id_Playlist) REFERENCES Playlist(id_Playlist) ON DELETE CASCADE ON UPDATE CASCADE

Savedplaylist

Field Name	Data Type + Constraints	Description
account	INT NOT NULL	Account that saves the playlist
id_Playlist	INT NOT NULL	ID of the saved playlist

The definition of the table includes the following constraints:

- PRIMARY KEY: (account, id_Playlist)
- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE
- FOREIGN KEY: id_Playlist REFERENCES Playlist(id_Playlist) ON DELETE CASCADE ON UPDATE CASCADE

Comments

Field Name	Data Type + Constraints	Description
id_Comment	SERIAL	Unique code for each comment
account	INT NOT NULL	Who comments
id_Video	INT NOT NULL	Video that is commented
message	varchar(500) NOT NULL	Text of the comment
donation	float	Field for possible donations
commentDate	timestamp NOT NULL	Date and time of the comment
id_Reply	INT	Possible ID of another comment to which one replies

The definition of the table includes the following constraints:

- PRIMARY KEY: id_Comment
- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE
- FOREIGN KEY: id_Video REFERENCES Video(id_Video) ON DELETE CASCADE ON UPDATE CASCADE

- FOREIGN KEY: id_Reply REFERENCES Comments(id_Comment) ON DELETE CASCADE ON UPDATE CASCADE
- CHECK: donation >= 0

VideoReports

Field Name	Data Type + Constraints	Description
id_Report	SERIAL	Unique code for each report
account	INT NOT NULL	Who reports
id_Video	INT NOT NULL	Reported video
reason	Reason	Reason for the report
description	varchar(50)	Description of the report
date	timestamp NOT NULL	Report date

The primary keys and foreign keys are kept as in the original:

- PRIMARY KEY: id_Report
- FOREIGN KEY: account REFERENCES (id_Account) ON DELETE CASCADE ON UPDATE CASCADE
- FOREIGN KEY: id_Video REFERENCES (id_Video) ON DELETE CASCADE ON UPDATE CASCADE

CommentReports

Field Name	Data Type + Constraints	Description
id_Report	SERIAL	Unique code for each report
account	INT NOT NULL	Who reports
id_Comment	INT NOT NULL	Reported comment
reason	Reason	Reason for the report
date	timestamp NOT NULL	Report date

The primary keys and foreign keys are kept as
in the original:

- PRIMARY KEY: id_Report

- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE
- FOREIGN KEY: id_Comment REFERENCES Comments(id_Comment) ON DELETE CASCADE ON UPDATE CASCADE

VideoLikes

Field Name	Data Type + Constraints	Description
account	INT NOT NULL	Who votes
id_Video	INT NOT NULL	Voted video
likeDate	timestamp NOT NULL	Vote date
isLike	boolean NOT NULL	If 'True', it's a like, otherwise it's a dislike

Constraints:

- PRIMARY KEY: (account, id_Video)
- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE
- FOREIGN KEY: id_Video REFERENCES Video(id_Video) ON DELETE CASCADE ON UPDATE CASCADE

CommentLikes

Field Name	Data Type + Constraints	Description
account	INT NOT NULL	Who votes
id_Comment	INT NOT NULL	Voted comment
likeDate	timestamp NOT NULL	Vote date
isLike	boolean NOT NULL	If 'True', it's a like, otherwise it's a dislike

Constraints:

- PRIMARY KEY: (account, id_Comment)
- FOREIGN KEY: account REFERENCES Account(id_Account) ON DELETE CASCADE ON UPDATE CASCADE

- FOREIGN KEY: id_Comment REFERENCES Comment(id_Comment) ON DELETE CASCADE ON UPDATE CASCADE

Conclusion

For populating the database, for smaller tables, ChatGPT was used: it is very useful for generating imaginative and always different data, but with significant limitations regarding the quantity of produced results and the maintenance of various constraints between the tables (for example, it sometimes generated comments with a date preceding the publication date of the video). All errors of this kind had to be manually verified and fixed, causing a lot of time loss.

For larger tables, C++ software was instead developed to populate them.
This method certainly requires more effort and time to be implemented, but it guarantees results of the highest quality and reliability compared to the results provided by ChatGPT.

In the "Popolamento" folder, these files are attached.