Projet Allociné

Franck Le Fur

Table des matières

1 Présentation **2**
- 1.1 Présentation du projet . . . 2
- 1.2 Spécifications techniques . . . 2

2 Préliminaires **4**
- 2.1 Création de l'environnement . . . 4
  - 2.1.1 Création d'un nouvel environnement . . . 4
  - 2.1.2 Installation des packages . . . 4
  - 2.1.3 GitHub . . . 4

3 Webscrapping **6**
- 3.1 Webscrapping . . . 6
- 3.2 Scraper les films déjà sortis . . . 6
- 3.3 Utilisation de Selenium . . . 6
- 3.4 Scraper les nouveaux films . . . 7

4 Requêtage d'une API publique **9**
- 4.1 API OMDB . . . 9
- 4.2 9

5 Création de la base de données **10**
- 5.1 SGBD . . . 10
- 5.2 Modèles MCD et MPD . . . 10
- 5.3 Création de la base et des tables . . . 12
- 5.4 Aggrégation et nettoyage des données . . . 13
- 5.5 Exemples de requêtes SQL . . . 13

6 Automatisation **14**
- 6.1 Présentation de Crontab . . . 14
- 6.2 Script à exécuter . . . 14
- 6.3 Création d'un environnement virtuel sous WSL . . . 14
- 6.4 Création d'une tâche Crontab . . . 14

7 Création d'une API **16**
- 7.1 Fast API . . . 16
- 7.2 16
- 7.3 CRUD . . . 16
- 7.4 Démonstration de l'API . . . 17

1 Présentation

1.1 Présentation du projet

Ce projet présente une application permettant de faire des requêtes sur une base de données concernant le cinéma. L'application offre les options classiques de filtres de films selon des mots clés dans le titre, les noms d'acteurs, le réalisateurs, le compositeur, l'année de production, les catégories de films.
Elle offre également la possibilité de combiner ensemble tous ces filtres et ainsi de répondre à des questions telles que :

— Quels films ont réuni les acteurs Pierre Richard et Gérard Depardieu ?

— Dans combien de films de la franchise "Terminator" ont joué ensemble Linda Hamilton et Schwarzenegger ?

— Combien de films avec Pierre Richard ont vu leur musique composée par Vladimir Cosma ?

Cette application peut être proposée à des professionnels du cinéma, par exemple des critiques de films, ayant besoin de filtres avancés pour faire des recherches sur des associations dans le cinéma.

L'**objectif fonctionnel** est de construire une base de données et de l'exposer via une API
Organisation du travail :

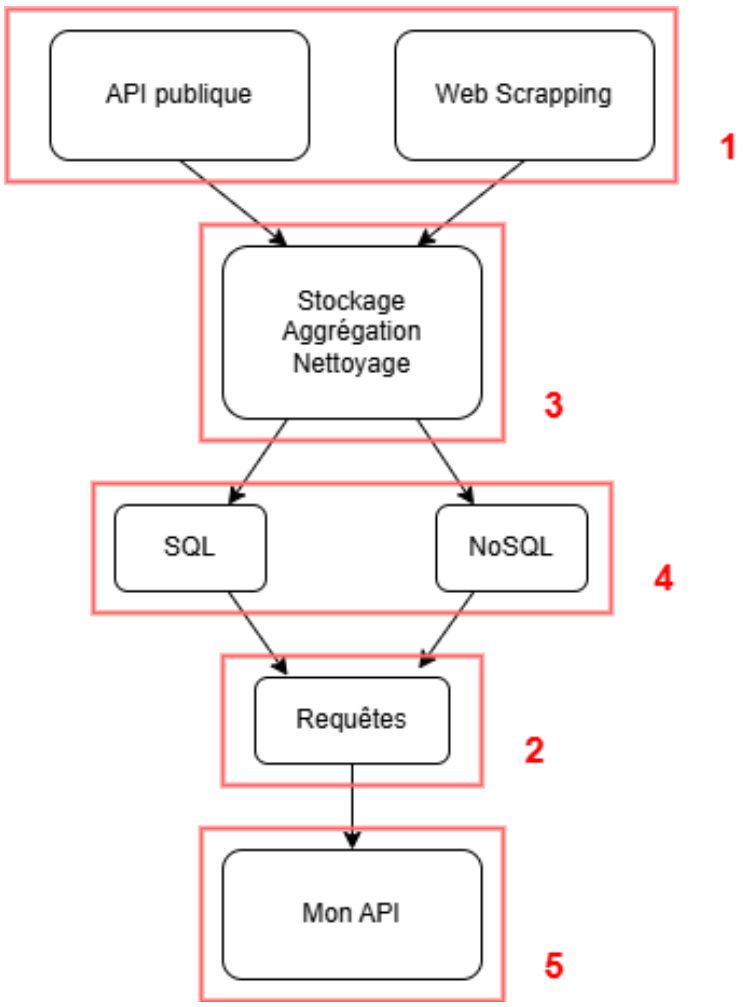
1.2 Spécifications techniques

Ce projet sera mené en langage Python, Nous choisissons le langage Python dans ce projet car il dispose d'un grand nombre de librairies gratuites permettant de mener à bien les différentes étapes de ce projet : Scrapping des données sur internet, requêtage de d'une API publique, sauvegarde de ces données dans des fichiers csv,

les librairies Python : Numpy et Pandas pour la manipulation des données en python,

Les technologies utilisées (python MySQL) et pourquoi Les accès aux API, quels API (OMDB) et comment j'ai fait etc ...

Nous disposons d'une base de données SQL et MongoDB (NoSQL), toutes deux construites à partir de données du site "Allociné".


API publique
Web Scrapping
1
Stockage
Aggrégation
Nettoyage
3
SQL
NoSQL
4
Requêtes
2
Mon API
5

2 Préliminaires

2.1 Création de l'environnement

2.1.1 Création d'un nouvel environnement

```
C:\Users\Utilisateur>conda create --name block1
C:\Users\Utilisateur>conda activate block1
```

2.1.2 Installation des packages

```
C:\Users\Utilisateur>conda install numpy scipy seaborn tqdm pandas
    matplotlib
C:\Users\Utilisateur>conda install requests beautifulsoup4 selenium
C:\Users\Utilisateur>pip install mysql-connector-python
```

2.1.3 GitHub

Commandes sur Git Bash

mix beautiful and selenium

Franck-LF committed on Dec 27, 2024

Commits on Dec 26, 2024

start with selenium to scrap the website

Franck-LF committed on Dec 26, 2024

Commits on Dec 18, 2024

change a few things

Franck-LF committed on Dec 18, 2024

Commits on Dec 16, 2024

a few bits changed

Franck-LF committed on Dec 16, 2024

Commits on Dec 14, 2024

start the webscraping part

Franck-LF committed on Dec 14, 2024

start the project, add 3 new files

Franck-LF committed on Dec 14, 2024

3 Webscrapping

3.1 Webscrapping

3.2 Scraper les films déjà sortis

Nous scrappons les films déjà sortis (entre 1960 et 2024)

On scrape la liste des catégories de films à partir du site allocine.com

Dans un premier temps nous scrapons les catégories et les pays à partir des listes du site (voir ci-dessous).

Ce n'est pas forcément pertinent car certains éléments n'apparaissent pas dans ces listes mais apparaissent dans les informations des films (par exemple 'Bostwana' n'est pas dans la liste des pays mais il y a quelques films dont le pays associé est 'Bostwana'). Au lieu de scraper ces listes nous déduirons directements les listes des catégories et des pays à partir des films dont les informations auront été scrapées.

Nous scrapons la liste des années, puis pour chaque année nous scrapons les films

3.3 Utilisation de Selenium

Lors de l'utilisation de Beautiful Soup, certains éléments sont **décorés**, certains liens sont **invisibles**, on ne peut pas directement les scraper. Le contournement trouvé est d'utiliser la librairie Pyhon Selenium qui permet (entre autre) :

— d'utiliser les XPath,

— de récupérer tous les élements et non-décorés.

Pour montrer les limites de Beautiful Soup
Résultat avec beautiful Soup

```
<li class="filter-entity-item">
 <span class="ACrL2ZACrpbG1zL2RlY2VubmllLTIwMzAv item-content" title="2030 - 2039">
  2030 - 2039
 </span>
 <span class="light">
  (47)
 </span>
</li>
<li class="filter-entity-item">
 <span class="ACrL2ZACrpbG1zL2RlY2VubmllLTIwMjAv item-content" title="2020 - 2029">
  2020 - 2029
 </span>
 <span class="light">
  (17931)
 </span>
</li>
```

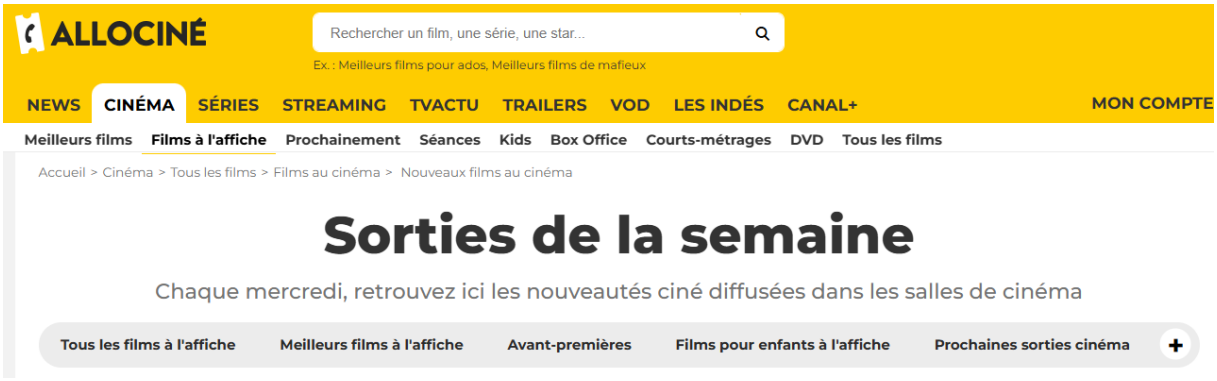
Résultat avec Selenium

```
▼ <li class="filter-entity-item"> == $0
    <a class="xXx item-content" title="2030 - 2039" href="/films/decennie-2030/">2030 - 2039</a>
    <span class="light">(47)</span>
  </li>
▶ <li class="filter-entity-item"> ··· </li>
▶ <li class="filter-entity-item"> ··· </li>
▶ <li class="filter-entity-item"> ··· </li>
▶ <li class="filter-entity-item"> ··· </li>
▶ <li class="filter-entity-item"> ··· </li>
▶ <li class="filter-entity-item"> ··· </li>
▶ <li class="filter-entity-item"> ··· </li>
▶ <li class="filter-entity-item"> ··· </li>
```

Nous n'enregistrons pas les affiches de films dans la base de données, ce n'est pas très pertinent d'un point de vue gestion de la mémoire, nous préférons garder en base les urls des affiches, les affiches étant déjà stockées sur internet.

3.4 Scraper les nouveaux films

La tâche est un peu différente puisque nous scrappons les films à partir de la page Allocine "Les sorties de la semaine"



Ensuite nous récupérons les pages des films à l'aise de Beautiful Soup, la suite du scrapping est identique à l'exception de quelques points : Les films récents n'ont pas de section "ratings" ou bien ont des sections "ratings" vides, il faut séparer les différents cas dans le scrapping. Un autre problème plus compliqué est que lors du scrapping des films par années certains films étaient recensés en 2024 (année de la production) mais sont à l'affiche en 2025, nous risquons de les scrapper 2 fois si nous ne faisons pas attention, pour cela nous ajoutons une requêt SQL et vérifions à la fois le titre et la date de sortie des films en base (vérifier uniquement le titre n'est pas suffisant), ainsi nous nous assurons de ne pas insérer 2 fois le même film dans la base.

4 Requêtage d'une API publique

4.1 API OMDB

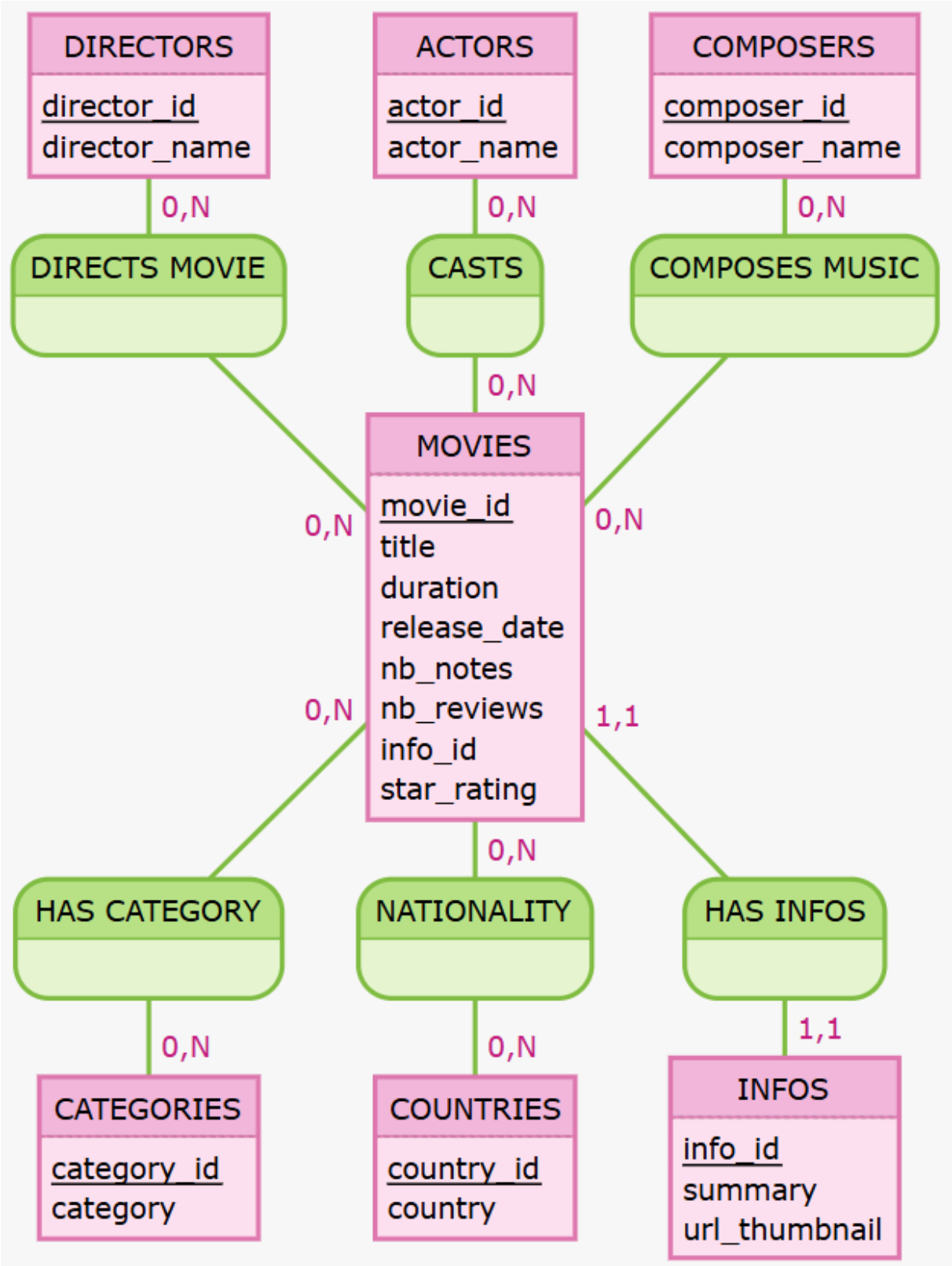
4.2 ...

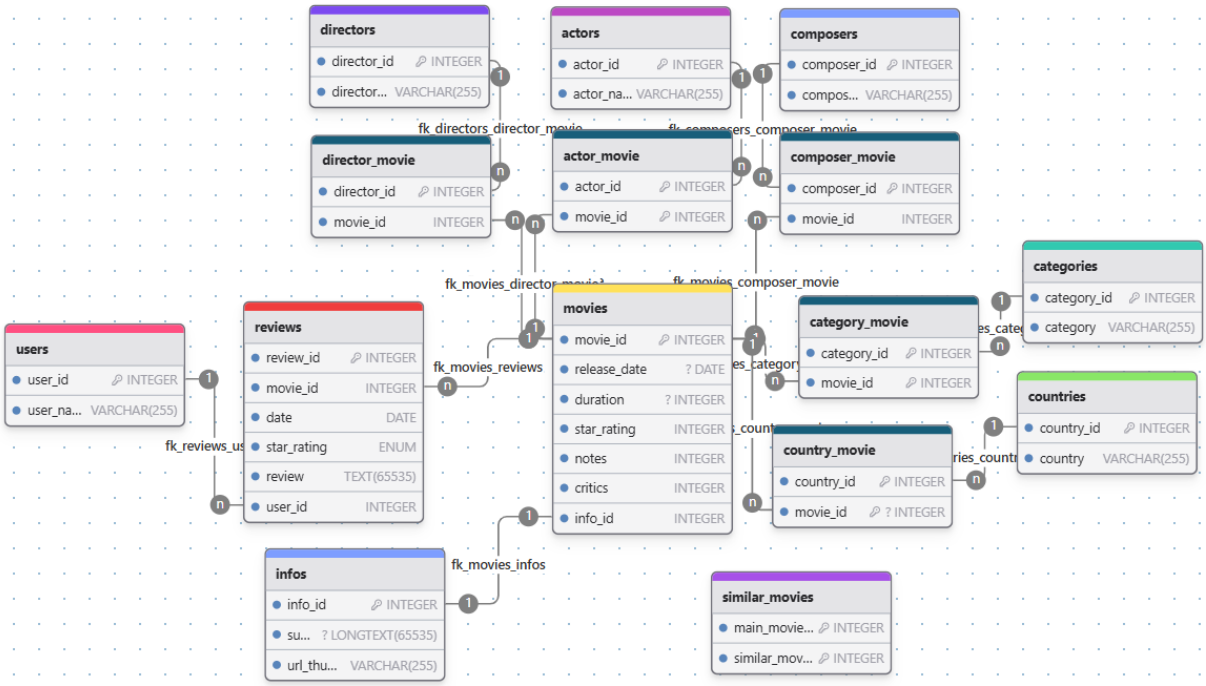
5 Création de la base de données

5.1 SGBD

5.2 Modèles MCD et MPD

MCD : Modèle conceptuel de données (modèle abstrait, "Le MCD est une carte mentale des données, offrant une compréhension partagée entre développeurs et décideurs")
MPD : Modèle physique de données ("Le MPD, c'est là où la théorie se transforme en lignes de code fonctionnelles")


DIRECTORS
director_id
director_name
ACTORS
actor_id
actor_name
COMPOSERS
composer_id
composer_name
0,N
0,N
0,N
DIRECTS MOVIE
CASTS
COMPOSES MUSIC
0,N
MOVIES
movie_id
title
duration
release_date
nb_notes
nb_reviews
info_id
star_rating
0,N
0,N
0,N
1,1
0,N
HAS CATEGORY
NATIONALITY
HAS INFOS
0,N
0,N
1,1
CATEGORIES
category_id
category
COUNTRIES
country_id
country
INFOS
info_id
summary
url_thumbnail



5.3 Création de la base et des tables

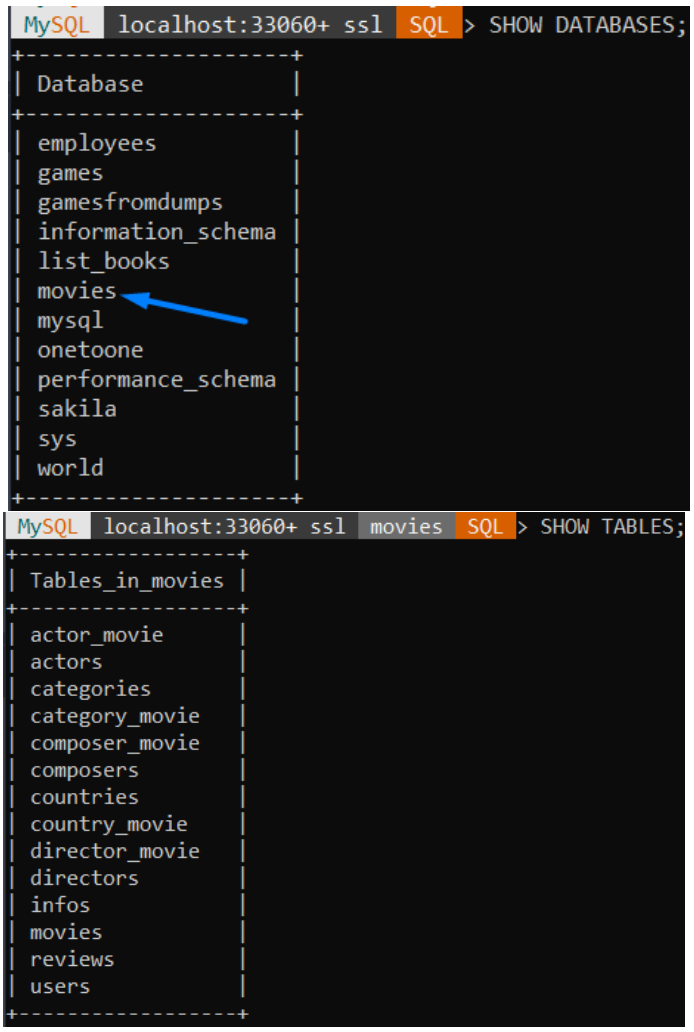
`sql` pour passer en mode SQL

`connect root@localhost` pour ouvrir une session sur MySQL

Une fois le fichier movies.sql corrigé nous pouvons l'exécuter avec MySQL

```
C:\Users\Utilisateur\Documents\Block1>"C:\Program Files\MySQL\MySQL Server
    8.0\bin\mysql.exe" < movies.sql -u root -p
Enter password: *****
INFO
CREATING DATABASE STRUCTURE
INFO
storage engine: InnoDB
INFO
EVERYTHING IS OK
```

La base et les tables ont bien été créées.



5.4 Aggrégation et nettoyage des données

La base de données MySQL a des contraintes sur les types de champs (integer, varchar etc...), avant d'insérer les données en base de données nous devons nous assurer que les données sont au bon format, dans le cas contraire MySQL nous renverra un message d'erreur.

A noter qu'un nettoyage a déjà été fait lors du scrapping.

5.5 Exemples de requêtes SQL

6 Automatisation

6.1 Présentation de Crontab

Crontab est une application d'automatisation de tâches sur système Unix et Linux, plus précisémment elle permet l'exécution de scripts en arrière-plan à des moments précis. Nous utiliserons Crontab pour mettre à jour notre base de données en lui ajoutant chaque semaine les nouveaux films, pour cela nous créerons un environnement sur une machine virtuelle Linux WSL puis une tâche qui exécutera notre script tous les mercredis matins à 8h (date de sortie des films).

6.2 Script à exécuter

Notre script comportera les étapes suivantes :

- Scrapping des films de la semaine à partir de l'url dédiée sur allocine.fr,
- Requête de OMDB pour récupérer des informations supplémentaires,
- Nettoyage des données
- Enregistrement des données dans les bases MySQL et MongoDb.

6.3 Création d'un environnement virtuel sous WSL

`python -m venv env_allocine`
Activation de l'environnement
`source env_allocine/bin/activate`
Installation des packages nécessaires à l'exécution du script
`pip install pandas requests beautifulsoup4 mysql-connector-python`

6.4 Création d'une tâche Crontab

Préparation de l'environnement et création de la tâche Crontab

1 Création sous WSL d'un environnement virtuel venv : `python -m venv env`

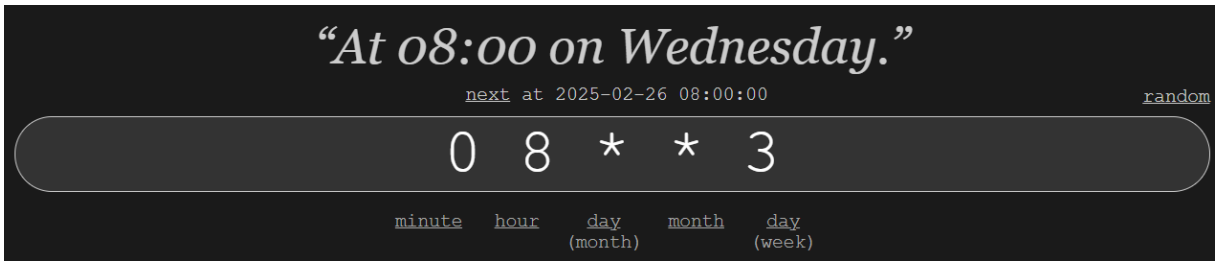
2 Activer l'environnement : `source /env/bin/activate`

3 Installer les librairies nécessaires à l'utilisation du script "numpy", "pandas", "requests", "beautifulsoup4", "httpx", "selenium", "mysql-connector-python"

4 Copie du script.py dans WSL

5 Création de la tâche crontab qui doit se lancer chaque mercredi à 8 heures du matin, ajout d'information dans un fichier log.

```
0 8 3 * * cd /home/franck/testCron && . ~/testCron/env_allocine/bin/activate
&& cd /home/franck/testCron && python script.py »
~/testCron/allocine.log
```



Minute (0-59), Heure (0-23), Jour du mois (1-31), Mois (1-12), Jour de la semaine (0-7, où 0 et 7 représentent dimanche)

Problème lors de l'exécution de la tâche : il ne trouve pas mes modules python

7 Création d'une API

7.1 Fast API

7.2 ...

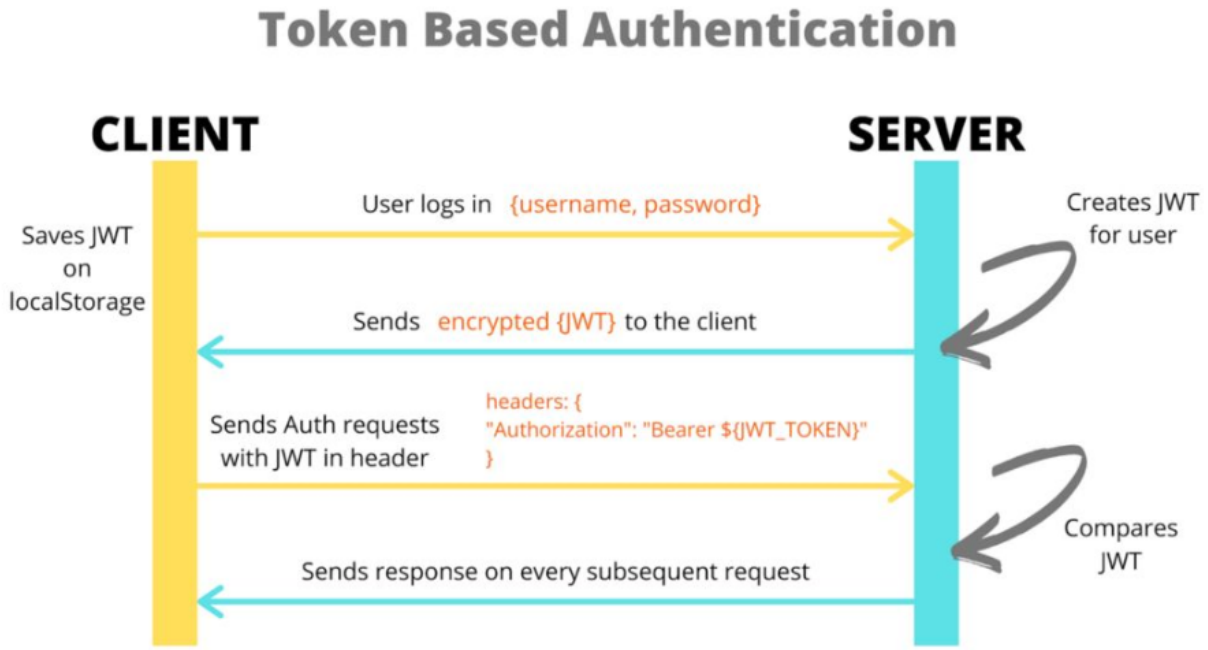
On lance le serveur avec la commande `uvicorn testapi:app -reload`

JWT (JSON Web Token) est un standard pour communiquer des données entre différentes parties à travers des objets JSON. L'information est signée digitalement.

https ://www.geeksforgeeks.org/json-web-token-jwt/

Fonctionnement de JWT :

- Le client (navigateur) envoie au serveur les informations d'authentification (login et password),
- Si l'authentification est acceptée, le serveur génère un jeton JWT (signé avec une clé secrète) et le transmet au client,
- le client peut envoyer des requêtes pour accéder à des ressources, il y joint son token,
- Si le token est bon, le serveur envoie au client les ressources demandées.



On install pyjwt

Nous utilisons asyncio https ://docs.python.org/3/library/asyncio.html pour la génération de tokens.

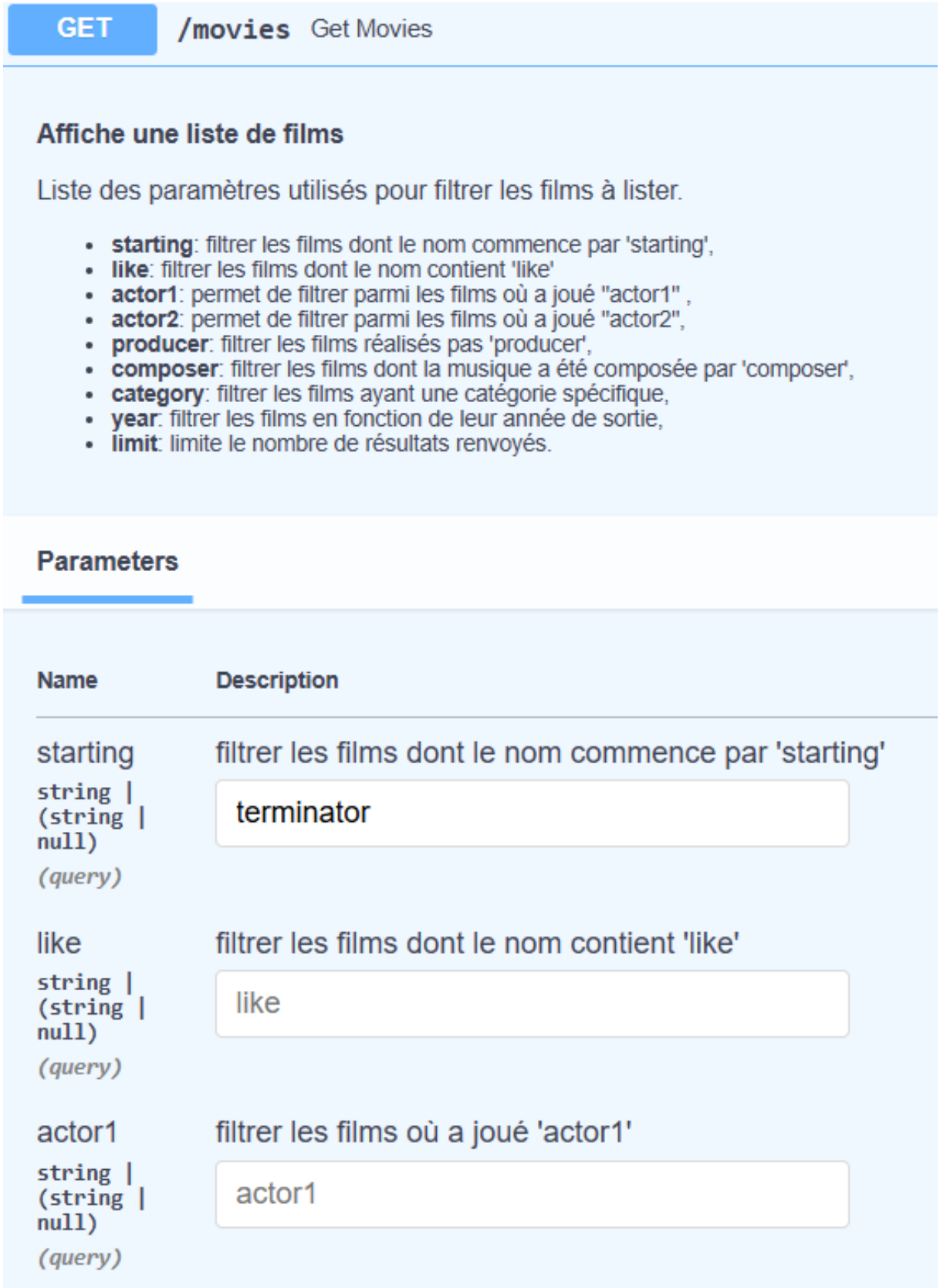
7.3 CRUD

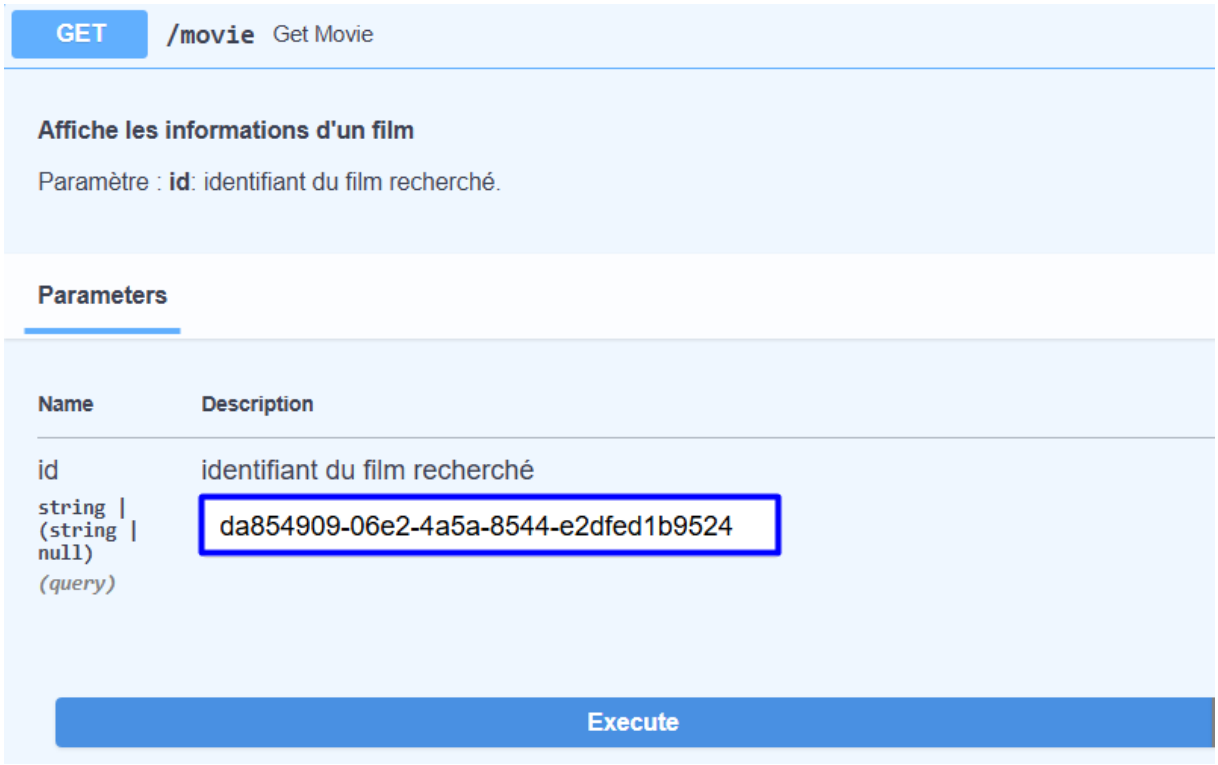
Nous exposons notre base de données à travers notre API, nous montrons des requêtes SQL / mongoDB permettant de sélectionner des films selon certains filtres (nom commençant pas, liste d'acteurs, réalisateurs, catégories, compositeurs, année de production ...)

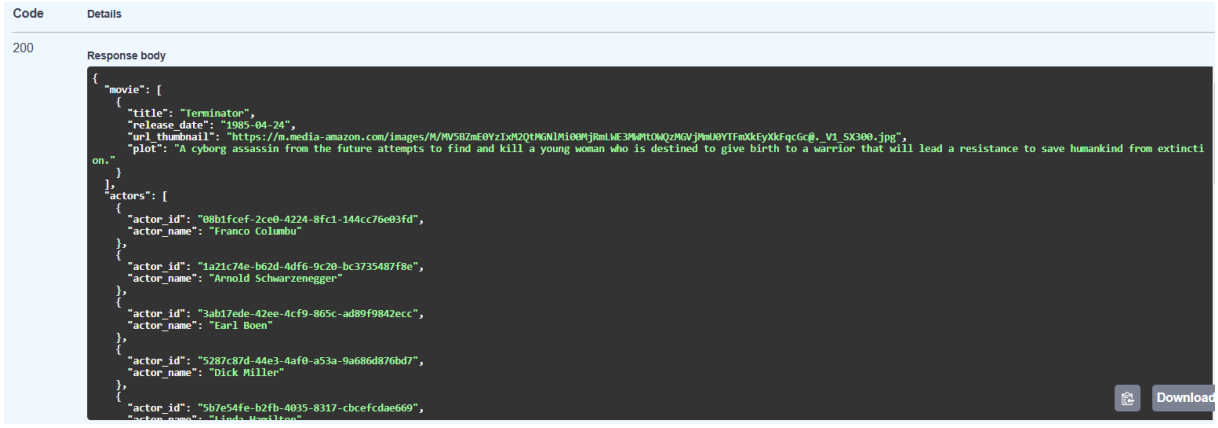
Uniquement le Read, on expose l'API

Un système de filtres a été mis en places pour pouvoir filtrer les films à partir de plusieurs noms d'acteurs et/ou du réalisateur et/ou du compositeur. Des filtres SQL pour faire des jointures entre les tables de films, acteurs, réalisateurs et compositeurs et un filtres pymongo pour extraire des informations de la base MongDB.

7.4 Démonstration de l'API




GET
/movie Get Movie
Affiche les informations d'un film
Paramètre : id: identifiant du film recherché.
Parameters
Name
Description
id
string | (string | null)
(query)
identifiant du film recherché
da854909-06e2-4a5a-8544-e2dfed1b9524
Execute


Code
Details
200
Response body
{
"movie": [
{
"title": "Terminator",
"release_date": "1985-04-24",
"url_thumbnail": "https://m.media-amazon.com/images/M/MV5BZmE0YzIxM2QtMGNlMi00MjRmLWE3MWMtOWQzMGVjMmU0YTFmXkEyXkFqcGc@._V1_SX300.jpg",
"plot": "A cyborg assassin from the future attempts to find and kill a young woman who is destined to give birth to a warrior that will lead a resistance to save humankind from extinction."
}
],
"actors": [
{
"actor_id": "08b1fcef-2ce0-4224-8fc1-144cc76e03fd",
"actor_name": "Franco Columbu"
},
{
"actor_id": "1a21c74e-b62d-4df6-9c20-bc3735487f8e",
"actor_name": "Arnold Schwarzenegger"
},
{
"actor_id": "3ab17ede-42ee-4cf9-865c-ad89f9842ecc",
"actor_name": "Earl Boen"
},
{
"actor_id": "5287c87d-44e3-4af0-a53a-9a686d876bd7",
"actor_name": "Dick Miller"
},
{
"actor_id": "5b7e54fe-b2fb-4035-8317-cbcefcdae669",
Download