

Projet Allociné



Franck Le Fur

Table des matières

1	Présentation	2
1.1	Présentation du projet	2
1.2	Spécifications techniques	2
1.3	Organisation du travail	3
2	Préliminaires	4
2.1	Préparation de l'environnement	4
2.1.1	GitHub	4
3	Webscrapping	5
3.1	Introduction	5
3.2	Scraper les informations des films de 1960 à 2024	6
3.3	Utilisation de Selenium	8
3.4	Scraper les nouveaux films	9
4	Requêtage d'une API publique	11
4.1	API OMDB	11
4.2	11
5	Création de la base de données	12
5.1	SGBD	12
5.2	Modèles MCD et MPD	12
5.3	Création de la base et des tables	14
5.4	Aggrégation et nettoyage des données	15
5.5	Exemples de requêtes SQL	15
6	Automatisation	16
6.1	Présentation de Crontab	16
6.2	Script à exécuter	16
6.3	Création d'un environnement virtuel sous WSL	16
6.4	Création d'une tâche Crontab	16
7	Création d'une API	18
7.1	Fast API	18
7.2	18
7.3	CRUD	19
7.4	Démonstration de l'API	20

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

1.2 Spécifications techniques

Ce projet sera mené en **Python**, ce langage offre toutes les librairies, de façon gratuites, nécessaires à la conduite de ce projet.

Les données seront collectées à partir du site allocine.fr ne utilisant les librairies python **Beautifulsoup** et **Selenium**.

Des données supplémentaires seront récupérées par requêtage de l'api publique **OMDB** en utilisant la librairie python **requests**.

Nettoyage et aggrégation des données seront fait à l'aise de librairies python **Numpy** et **Panda**.

Nous utiliserons deux SGBD : **MySQL** et **MongoDB**, la mise en base des données sera faite à l'aide des librairies python **mysql.connector** et **pymongo**.

Les tâches de scrapping, de requêtage de l'api public, de nettoyage et aggrégation des données seront automatisées à l'aide de l'outil **Crontab** disponible sur la machine Linux **WSL**.

L'API sera créée avec **FastAPI**, la documentation sera faite selon le modèle **OpenAPI**.

Un repo **github** sera créé pour le versioning des fichiers.

Enfin, le présent rapport sera rédigé en **Latex** via l'utilitaire en ligne **overleaf**.

1.3 Organisation du travail

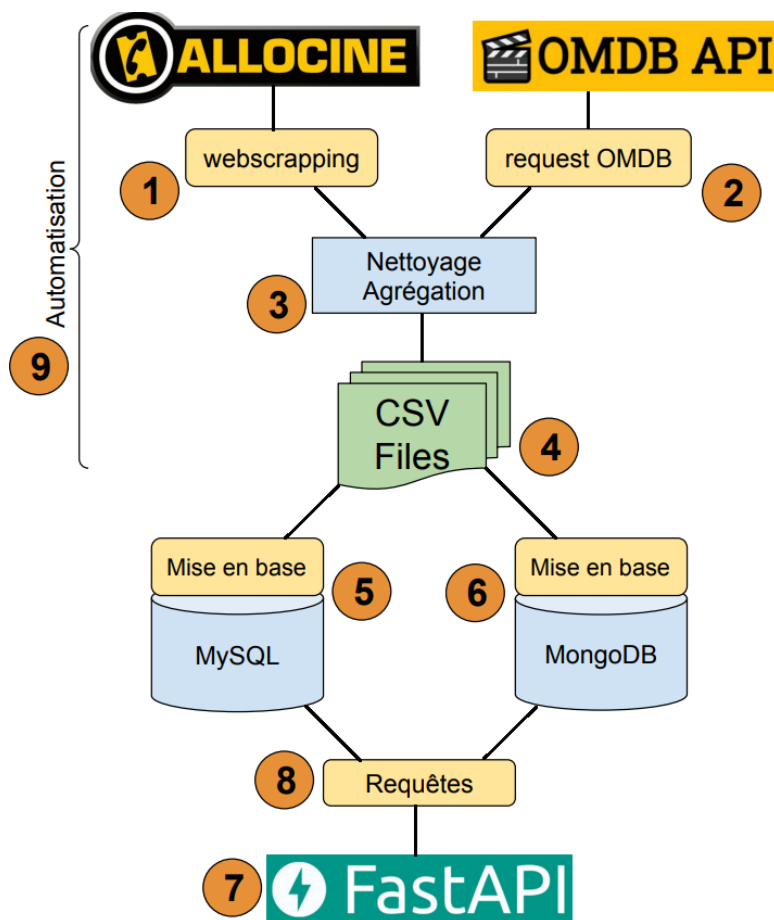


FIGURE 1 – Organigramme

Etapes

- 1 Scrapping du site allociné,
- 2 Requête de l'api publique OMDB,
- 3 Nettoyage et agrégation des données,
- 4 Ecriture des données en CSV,
- 5 Insertion des données dans MySQL,
- 6 Insertion des données dans MongoDB,
- 7 Création de l'API avec Fast,
- 8 Requêtes sur les bases pour notre API,
- 9 Automatisation de l'extraction des données.

2 Préliminaires

2.1 Préparation de l'environnement

Dans un premier temps nous créons un nouvel environnement virtuel sous conda avec toutes les librairies python nécessaires.

Création d'un nouvel environnement

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

Installation des packages python

```
1 C:\Users\Utilisateur>conda install numpy tqdm pandas matplotlib
2 C:\Users\Utilisateur>conda install requests beautifulsoup4 selenium
3 C:\Users\Utilisateur>conda install mysql-connector-python
4 C:\Users\Utilisateur>conda install pymongo</code><br>
5 C:\Users\Utilisateur>conda install unicode</code><br>
6 C:\Users\Utilisateur>conda install fastapi</code><br>
7 C:\Users\Utilisateur>conda install pyjwt</code><br>
8 C:\Users\Utilisateur>conda install uvicorn
```

2.1.1 GitHub

Création d'un repository **Github** via l'interface github puis connexion de notre répertoire de travail au repository **Github**.

```
git init
git add .gitignore
git branch -m master main (pour renommer la branche)
git commit -m "first commit"
git remote add origin https://github.com/Franck-LF/projectBlock1
git push -u origin main
```

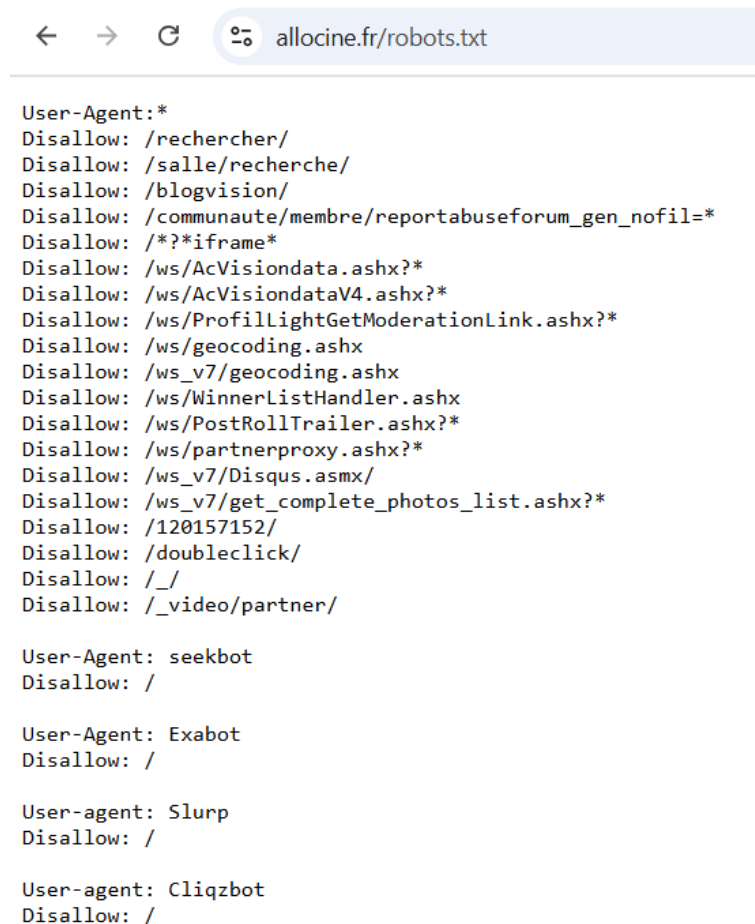
3 Webscrapping

3.1 Introduction

Le **webscrapping** est un processus d'extraction automatique de données à partir d'un site web. Il s'agit de parcourir des pages web et d'y récupérer le contenu souhaité. Ce processus doit être fait en respectant les conditions d'utilisation des sites et les lois sur la protection des données.

Dans ce projet les données seront scrappées à partir du site **allocine.fr**, ces données sont entièrement publiques, en effet il s'agit exclusivement de titres de films, noms d'acteurs, réalisateurs et compositeurs.

Il nous faut vérifier que le site **allocine.fr** autorise ce genre de pratiques, pour cela nous consultons le fichier "robots.txt".



```
User-Agent:*
Disallow: /rechercher/
Disallow: /salle/recherche/
Disallow: /blogvision/
Disallow: /communaute/membre/reportabuseforum_gen_nofil=*
Disallow: /*?*iframe*
Disallow: /ws/AcVissiondata.ashx?*
Disallow: /ws/AcVissiondataV4.ashx?*
Disallow: /ws/ProfillLightGetModerationLink.ashx?*
Disallow: /ws/geocoding.ashx
Disallow: /ws_v7/geocoding.ashx
Disallow: /ws/WinnerListHandler.ashx
Disallow: /ws/PostRollTrailer.ashx?*
Disallow: /ws/partnerproxy.ashx?*
Disallow: /ws_v7/Disqus.asmx/
Disallow: /ws_v7/get_complete_photos_list.ashx?*
Disallow: /120157152/
Disallow: /doubleclick/
Disallow: /_/
Disallow: /_video/partner/

User-Agent: seekbot
Disallow: /

User-Agent: Exabot
Disallow: /

User-agent: Slurp
Disallow: /

User-agent: Cliqzbot
Disallow: /
```

FIGURE 2 – fichier robots.txt

Le fichier robots.txt n'indique aucune restriction sur le chemin `/films/`, point d'entrée exclusif de notre web scraping, nous pouvons donc collecter en toute légalité les données souhaitées.

3.2 Scraper les informations des films de 1960 à 2024

Nous utilisons la librairie python **requests** pour récupérer le contenu de pages html, ensuite le web scraping se fera à l'aide de la librairie **Beautifulsoup**, cette librairie permet d'analyser et de parser le contenu html d'une page web.

Un usage réduit et très spécifique de la librairie **Selenium** sera détaillée [ici](#).

Nous allons scraper les films par année, pour cela nous commençons par scraper les liens des années disponibles à partir du menu **années**.

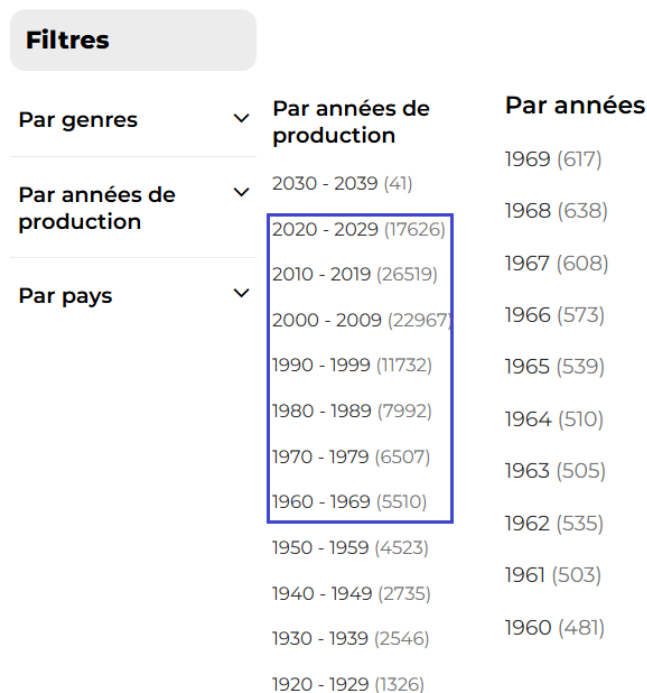


FIGURE 3 – filtre par années

Ensuite pour chaque année nous parcourons la liste de films puis à partir de la page du film, de la section **infos techniques** ainsi que de la section **Casting complet et équipe technique** nous collectons les informations suivantes :

- Le titre du film
- Le titre original du film
- La date de sortie
- La durée
- La liste des catégories associées
- Les pays de production
- La liste des acteurs
- La liste des réalisateurs
- La liste des compositeurs
- La note associée au film
- Le nombre de notes spectateurs
- Le nombre d'avis spectateurs



FIGURE 4 – Vignette du film

Acteurs et actrices









				Lesley Dunlop infirmière Nora
Anthony Hopkins Rôle : Dr. Frederick Treves (chirurgien)	John Hurt Rôle : John Merrick, Elephant Man	Anne Bancroft Rôle : Mrs. Madge Kendal (actrice)	John Gielgud Rôle : Carr Gomm (directeur de l'hôpital)	Helen Ryan La princesse Alex
				Kenny Baker Nain à plumes
				John Standing Fox
				Dexter Fletcher Le garçon de Bytes
				Phoebe Nicholls La mère de Merrick
				Pat Gorman Bobby au champ de foire
				Claire Davenport Femme obèse
				Orla Pederson Homme squelettique
				Patsy Smart Femme désespérée
				Frederick Treves Alderman
				Richard Hunter Hodges
				James Cormack Pierce
				Alfie Curtis Le livreur de lait
				Robert Lewis Bush Le messager
				Roy Evans Le chauffeur de taxi
				Joan Rhodes Le cuisinier
				Nula Conwell L'infirmière Kathleen
				Tony London Porter jeune
				
Wendy Hiller Rôle : L'infirmière en chef	Freddie Jones Rôle : Bytes (propriétaire d'Elephant Man)	Hannah Gordon Rôle : Mrs. Treves	Michael Elphick Rôle : Le gardien de nuit	

FIGURE 5 – Casting complet et équipe technique

Remarque : Nous ne collectons pas tous les films mais seulement les films ayant plus de 30 d'avis et nous fixons un maximum de 250 films par an, pour un total de 8800 films entre 1960 et 2024.

Remarque : Nous avons également scrapé les catégories et les pays à partir des listes du site mais il apparaît que la liste des pays à cet endroit du site n'est pas complète, en effet cette liste ne contient pas le **Bostwana** mais il existe bien des films dont le pays de production est le **Bostwana**. Finalement, nous construirons la liste des pays à partir des informations des films et non pas à partir de cette liste..

Infos techniques

Nationalité	U.S.A.
Distributeur	Carlotta Films
Récompenses	4 prix et 16 nominations
Année de production	1980
Date de sortie DVD	11/12/2001
Date de sortie Blu-ray	03/11/2009
Date de sortie VOD	08/02/2007
Type de film	Long métrage
Secrets de tournage	23 anecdotes
Budget	5 000 000 USD
Date de reprise	22/06/2020
Langues	Anglais
Format production	-
Couleur	N&B
Format audio	-
Format de projection	-
N° de Visa	54114

FIGURE 6 – Infos techniques

Filtres			
Par genres	▼	Par genres	Par années de production
		Action (9465)	2030 - 2039 (41)
		Animation (3624)	2020 - 2029 (17626)
		Arts Martiaux (460)	2010 - 2019 (26519)
		Aventure (6072)	2000 - 2009 (22967)
		Biopic (2767)	1990 - 1999 (11732)
		Bollywood (208)	1980 - 1989 (7992)
		Classique (18)	1970 - 1979 (6507)
		Comédie (21271)	1960 - 1969 (5510)
		Comédie dramatique (8102)	1950 - 1959 (4523)
		Comédie musicale (1171)	1940 - 1949 (2735)
		Concert (75)	1930 - 1939 (2546)
		Dessin Animé (6)	1920 - 1929 (1326)
		Divers (25636)	1910 - 1919 (469)
		Drama (34)	1900 - 1909 (66)
		Drame (41525)	1890 - 1899 (36)
Par années de production	▼		Par pays
			France (20341)
			U.S.A. (39321)
			Afrique du Sud (323)
			Albanie (55)
			Algérie (151)
			Allemagne (5074)
			Allemagne de l'Est (68)
			Allemagne de l'Ouest (744)
			Arabie Saoudite (57)
			Argentine (878)
			Arménie (33)
			Australie (1111)
			Autriche (639)
			Belgique (1886)
Par pays	▼		

3.3 Utilisation de Selenium

Lors de l'utilisation de BeautifulSoup, 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 éléments et non-décorés.

Pour montrer les limites de BeautifulSoup

Résultat avec BeautifulSoup

```

<li class="filter-entity-item">
  <span class="ACrL2ZACrpbG1zL2RlY2Vubml1LTlWmZAv item-content" title="2030 - 2039">
    2030 - 2039
  </span>
  <span class="light">
    (47)
  </span>
</li>
<li class="filter-entity-item">
  <span class="ACrL2ZACrpbG1zL2RlY2Vubml1LTlWmZAv item-content" title="2020 - 2029">
    2020 - 2029
  </span>
  <span class="light">
    (17931)
  </span>
</li>

```

Résultat avec Selenium

```

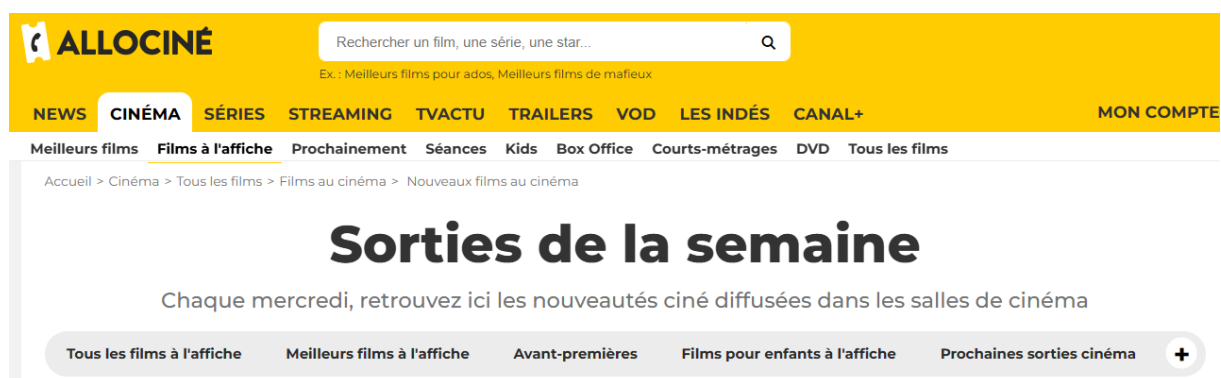
▼ <li class="filter-entity-item"> == $0
  <a class="xXx item-content" title="2030 - 2039" href="/films/decennie-2030/">2030 - 2039</a>
  <span class="light">(47)</span>
</li>
▶ <li class="filter-entity-item"> ... </li>
▶ <li class="filter-entity-item"> ... </li>
▶ <li class="filter-entity-item"> ... </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'aide de BeautifulSoup, 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 scraper 2 fois si nous ne faisons pas attention, pour cela nous ajoutons une requête 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 Requête d'une API publique

4.1 API OMDB

Accès à l'API etc ... Comment j'ai fait etc

4.2 ...

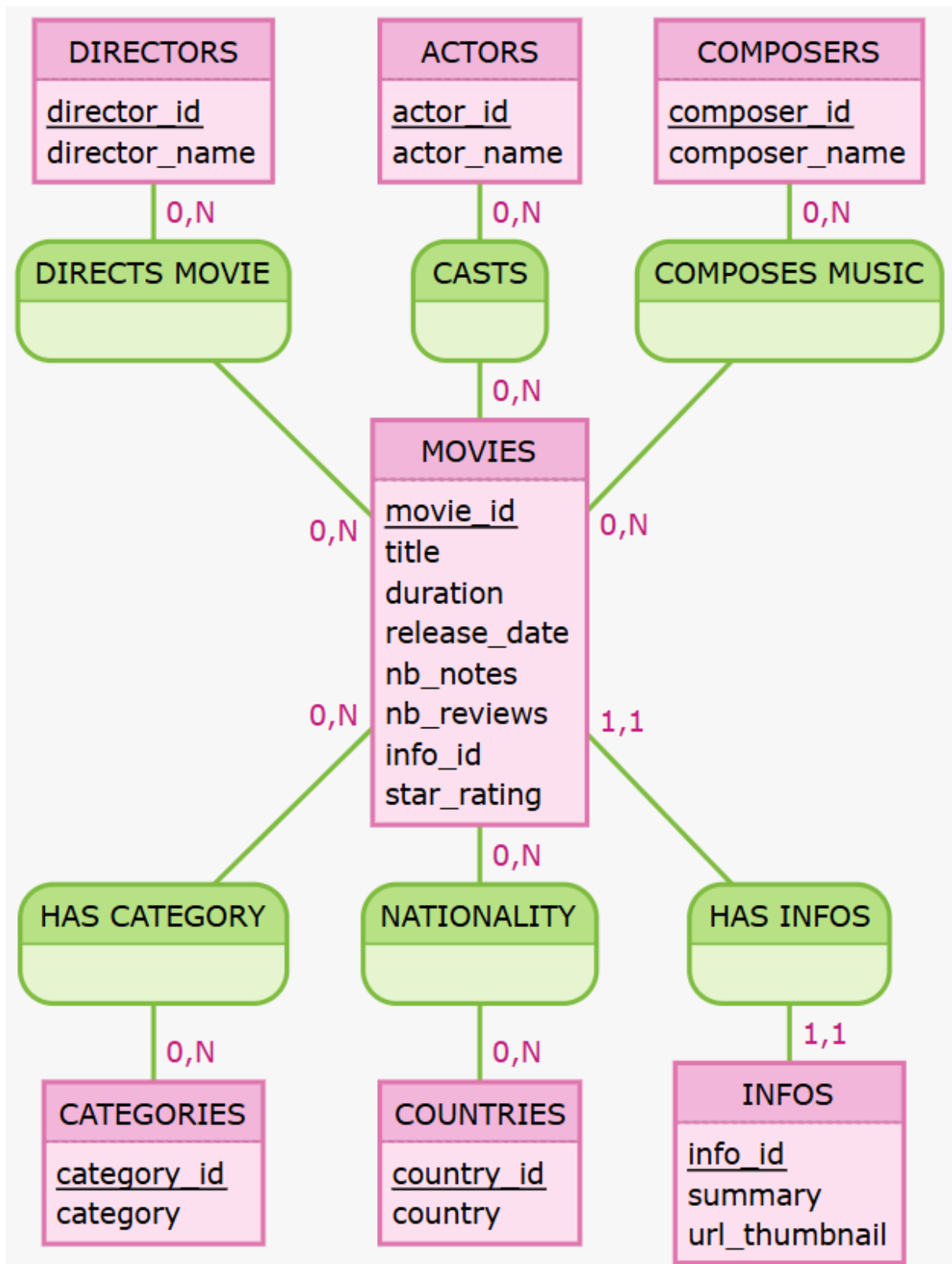
5 Création de la base de données

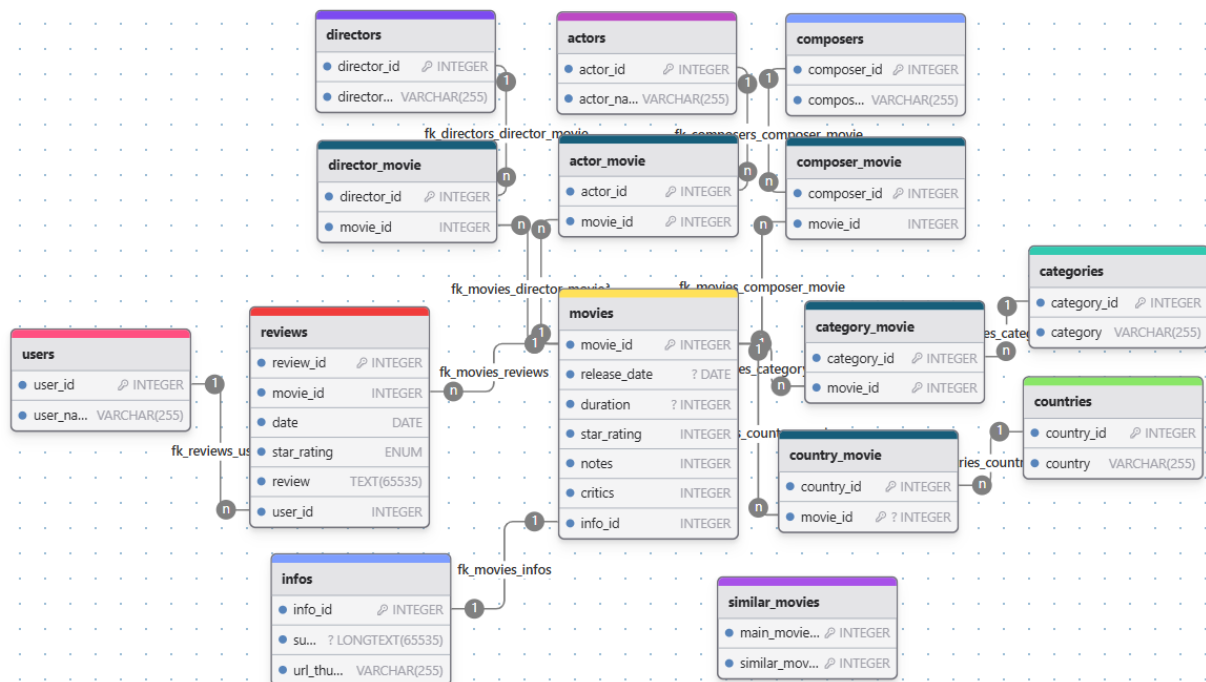
5.1 SGBD

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")





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

```

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

```

La base et les tables ont bien été créées.


```
MySQL localhost:33060+ ssl SQL > SHOW DATABASES;
+-----+
| Database |
+-----+
| employees |
| games |
| gamesfromdumps |
| information_schema |
| list_books |
| movies |
| mysql |
| onetoone |
| performance_schema |
| sakila |
| sys |
| world |
+-----+

MySQL localhost:33060+ ssl movies SQL > SHOW TABLES;
+-----+
| Tables_in_movies |
+-----+
| actor_movie |
| actors |
| categories |
| category_movie |
| composer_movie |
| composers |
| countries |
| country_movie |
| director_movie |
| directors |
| infos |
| movies |
| reviews |
| users |
+-----+
```

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 précisément 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`

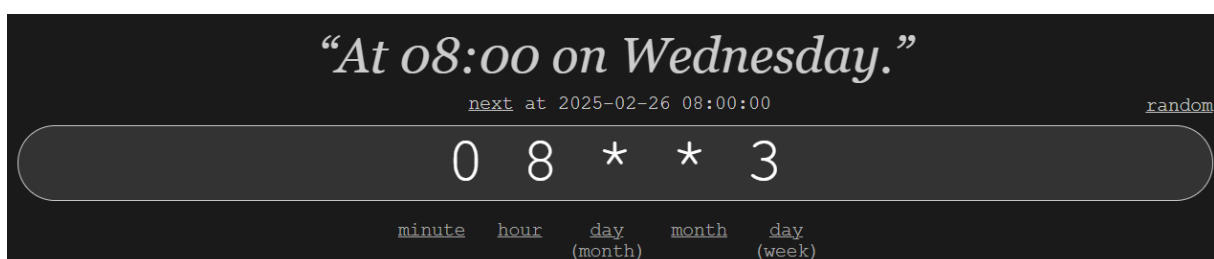
2 Activer l'environnement : `source /env/bin/activate`

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

Pourquoi utiliser une API ?

Le but de cette API est de rendre disponible notre base de données sans que l'utilisateur n'ait à connaître notre base, le nom des tables etc ... il n'aura qu'à consulter la documentation de l'API pour la requêter.

API (Application Programming Interface) partie d'un programme pouvant être utilisée par un autre programme, contrairement aux interfaces qui sont conçues pour être utilisées ou manipulées par des humains

Quand utiliser une API : - Lorsque les données à partager sont volumiques (plusieurs Giga), dans le cas de petits volumes on peut simplement communiquer un fichier Json, XML ..., - Lorsque les utilisateurs ont besoin d'accéder en temps réel aux données, - Lorsque les données sont modifiées ou mises à jour fréquemment, - Lorsque les utilisateurs n'ont besoin d'accéder qu'à une partie des données à la fois, - Lorsque les utilisateurs ont à effectuer d'autres opérations que la simple lecture, (mise à jour, suppression ...).

Terminologie des API : HTTP : principal moyen de communiquer sur internet, http implémente des méthodes comme GET et POST qui récupère des données sur un serveur et envoie de nouvelles données sur un serveur. URL : Pour exécuter une requête ou suivre un lien il suffit d'avoir un navigateur. JSON : Format de fichier texte pour stocker les données et être lisible à la fois par les humains et les machines. XML est un autre format de données utilisé pour les API. REST (REpresentational State Transfer) : Méthodologie de conception d'API, on peut aussi utiliser les termes API web, API http.

Documentation Swagger UI, Doxygen, Sphinx

7.2 ...

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

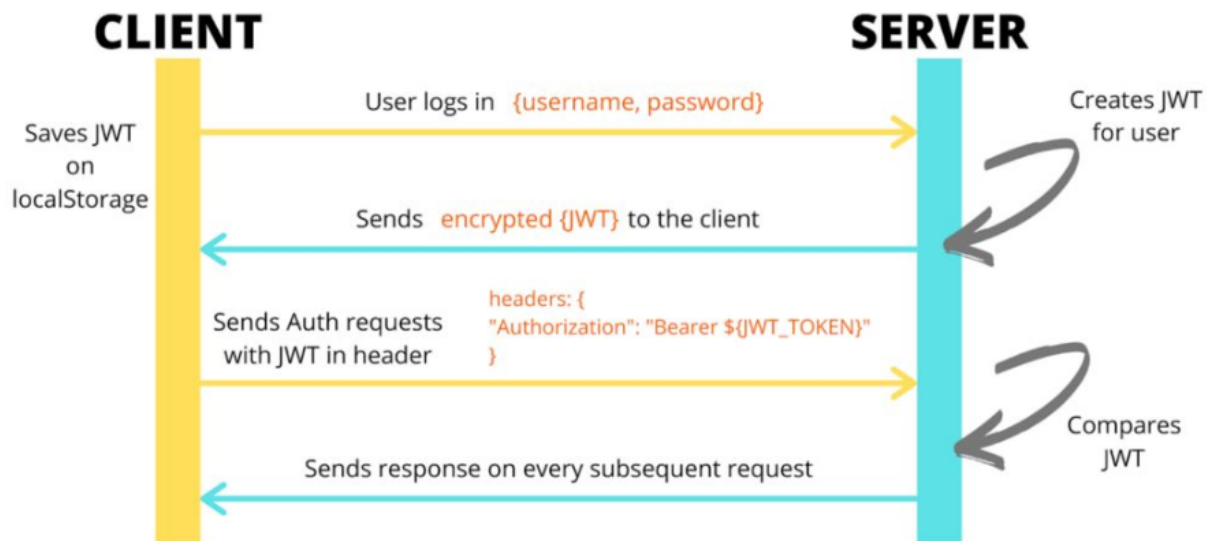
JWT (JSON Web Token) est un standard pour communiquer des données entre différentes parties à travers des objets JSON. L'information est signée numériquement.

<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.

Token Based Authentication



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 par, liste d'acteurs, réalisateurs, catégories, compositeurs, année de production ...)

Uniquement le Read, on expose l'API

Un système de filtres a été mis en place 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 filtre pymongo pour extraire des informations de la base MongoDB.

7.4 Démonstration de l'API

GET

/movies Get Movies

Affiche une liste de films

Liste des paramètres utilisés pour filtrer les films à lister.

- **starting**: filtrer les films dont le nom commence par 'starting',
- **like**: filtrer les films dont le nom contient 'like'
- **actor1**: permet de filtrer parmi les films où a joué "actor1",
- **actor2**: permet de filtrer parmi les films où a joué "actor2",
- **producer**: filtrer les films réalisés par 'producer',
- **composer**: filtrer les films dont la musique a été composée par 'composer',
- **category**: filtrer les films ayant une catégorie spécifique,
- **year**: filtrer les films en fonction de leur année de sortie,
- **limit**: limite le nombre de résultats renvoyés.

Parameters

Name	Description
starting string (string null) (query)	filtrer les films dont le nom commence par 'starting' <div>terminator</div>
like string (string null) (query)	filtrer les films dont le nom contient 'like' <div>like</div>
actor1 string (string null) (query)	filtrer les films où a joué 'actor1' <div>actor1</div>

GET

/movie Get Movie

Affiche les informations d'un film

Paramètre : **id**: identifiant du film recherché.

Parameters

Name	Description
------	-------------

id	identifiant du film recherché
----	-------------------------------

string |
(string |
null)
(query)

da854909-06e2-4a5a-8544-e2dfed1b9524

Execute

Code

Details

200

Response body

```
{
  "movie": [
    {
      "title": "Terminator",
      "release_date": "1985-04-24",
      "url_thumbnail": "https://ia.media-amazon.com/images/M/MV5BZmE0YzIyM2Q1MGNI100WjBmLWE3MmMtMQzNGVjMkY1fXkxYkFqcGc@_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",
      "actor_name": "Linda Hamilton"
    }
  ]
}
```



Download