

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	Preliminaires	4
2.1	Préparation de l'environnement	4
2.1.1	GitHub	4
3	Webscrapping	5
3.1	Introduction	5
3.2	Présentation des outils de web scraping	6
3.3	Scraper les informations des films de 1960 à 2024	6
3.4	Utilisation de Selenium	9
3.5	Scraper les films à l'affiche	9
3.6	Quelques difficultés rencontrées	9
4	Requêtage d'une API publique	11
4.1	Présentation de l'API OMDB	11
4.2	Récupération des informations	12
5	Nettoyage et agrégation des données	13
5.1	Nettoyage des données	13
6	Création de la base de données	14
6.1	Introduction au SGBD	14
6.2	Modèles MCD et MPD	14
6.3	Création de la base et des tables	16
6.4	Aggrégation et nettoyage des données	16
6.5	Questions	16
6.6	Exemples de requêtes SQL	17
7	Automatisation	18
7.1	Présentation de Crontab	18
7.2	Script à exécuter	18
7.3	Création d'un environnement virtuel sous WSL	18
7.4	Création d'une tâche Crontab	18
8	Création d'une API	20
8.1	Fast API	20
8.2	20
8.3	CRUD	21
8.4	Démonstration de l'API	22

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](https://www.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

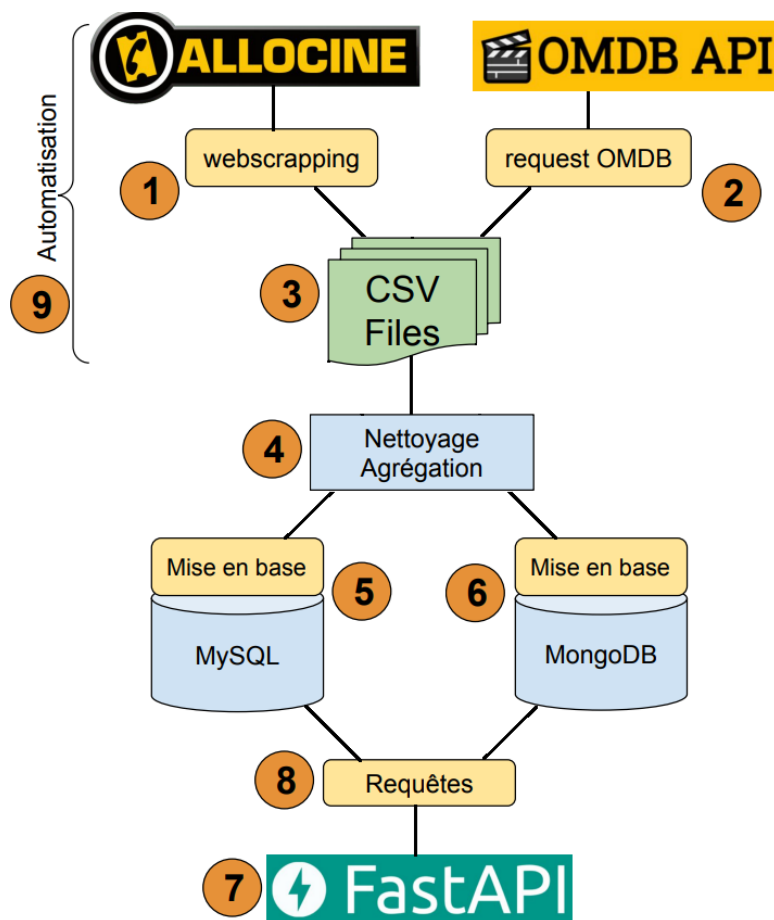


FIGURE 1 – Organigramme

Etapes

- 1 Scrapping du site allociné,
- 2 Requête de l'api publique OMDB,
- 3 Ecriture des données en CSV,
- 4 Nettoyage et agrégation des données,
- 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
```

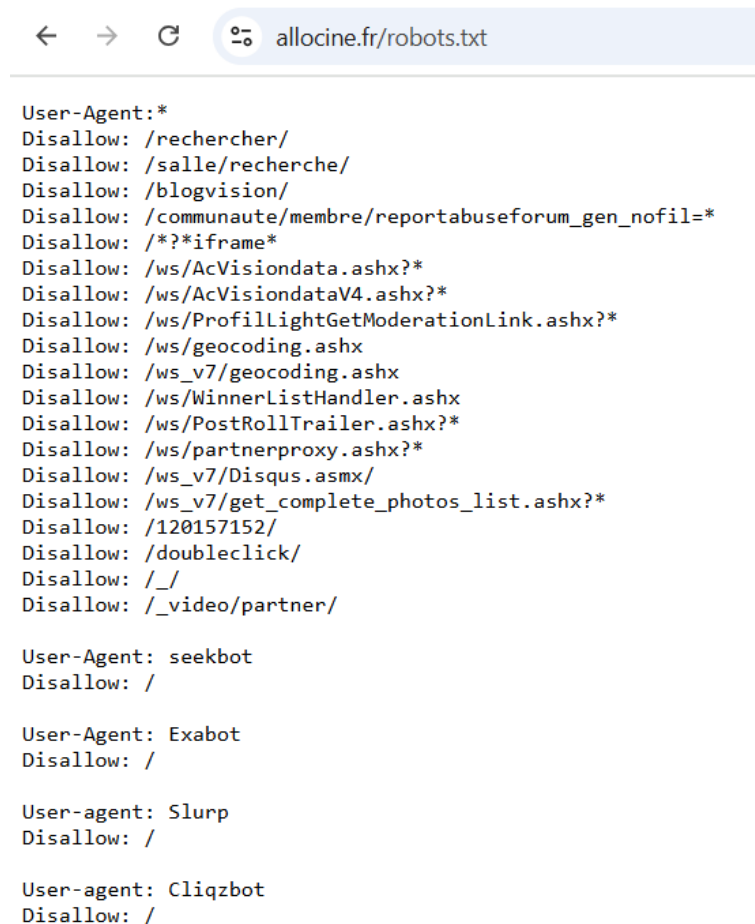
3 Webscrapping

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 Présentation des outils de web scraping

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](#).)

Exemple classique d'utilisation de BeautifulSoup

On commence par utiliser l'inspecteur du navigateur pour détecter les balises html qui contiennent les informations souhaitées, ensuite on utilise la méthode **find** sur un objet **soup** de BeautifulSoup pour récupérer le contenu de ces balises, par exemple la commande

```
elt_categories = soup.find('div', class_='filter-entity-section')
```

renvoie toutes les balises html **div** ayant un attribut **class** égal à "filter-entity-section".

Ensuite on peut récupérer le texte d'une balise :

```
elt.a.text
```

ou bien la valeur d'un attribut :

```
elt.get_attribute('title')
```

3.3 Scraper les informations des films de 1960 à 2024

Nous allons scraper les films de 1960 à 2024, nous nous limiterons aux films ayant plus de 30 avis utilisateurs et nous nous fixons un maximum de 250 films par année.

Finalement 8800 films seront collectés.

Nous commençons par scraper les liens des années disponibles à partir du menu **années**.

Remarque :

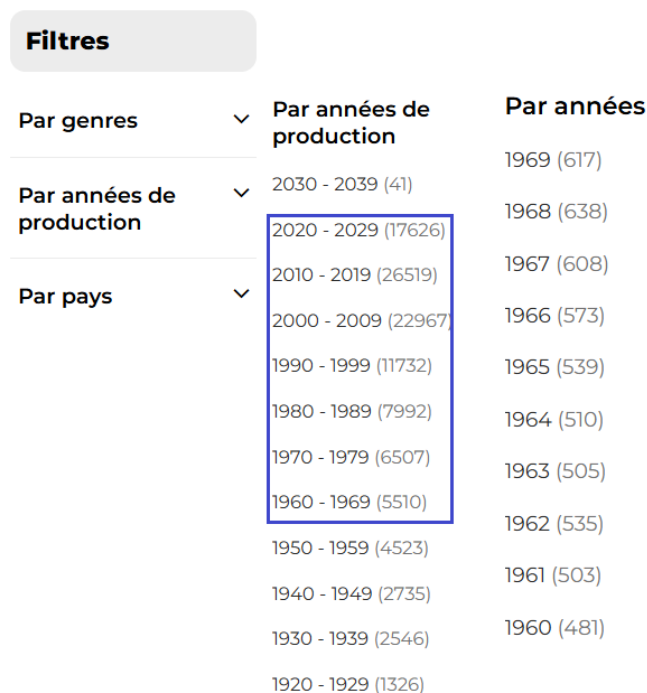


FIGURE 3 – filtre par années

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

Lors du scraping ces données seront stockées dans des **DataFrame Pandas**. Notons que toutes ces informations sont structurées et seront destinés à aller en base SQL.



FIGURE 4 – Vignette du film

Acteurs et actrices



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)



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

Lesley Dunlop infirmière Nora
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 messenger
Roy Evans Le chauffeur de taxi
Joan Rhodes Le cuisinier
Nula Conwell L'infirmière Kathleen
Tony London Porter jeune

FIGURE 5 – Casting complet et équipe technique

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

3.4 Utilisation de Selenium

Lors de l'utilisation de BeautifulSoup, certains liens apparaissent **décorés** et sont donc inexploitable, l'utilisation de la librairie **Selenium** permet de résoudre ce problème.

```
<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>
```

FIGURE 7 – Inspecteur Chrome

```
<span class="ACrL2ZACrpbG1zL2RlY2VubmllLTlWmZAv item-content" title="2030 - 2039">
  2030 - 2039
</span>
décoration
```

FIGURE 8 – Résultat BeautifulSoup

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.

3.5 Scraper les films à l'affiche

Ici nous ne scrapons pas les films de toute une année mais uniquement les films de la semaine à partir de la page allociné dédiée. Cette tâche sera automatisée de façon hebdomadaire pour récupérer les informations de tous les nouveaux films à l'affiche (voir section Crontab).

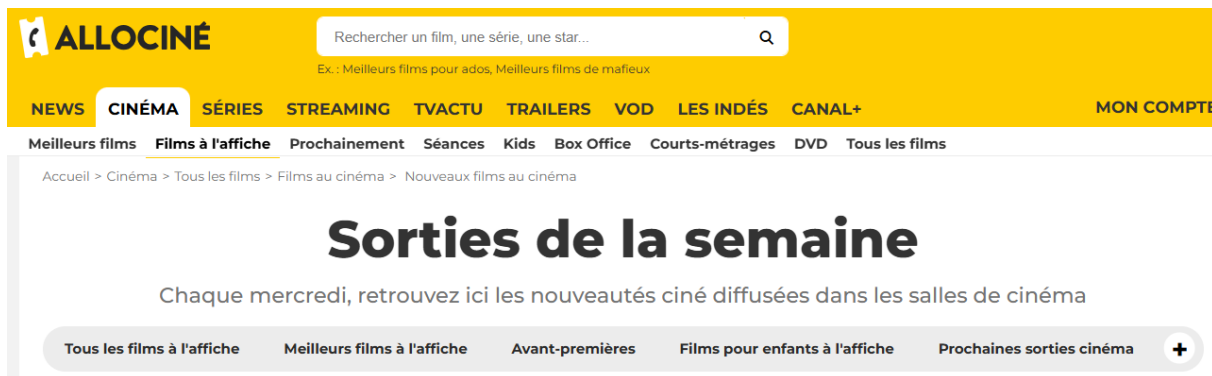


FIGURE 9 – Section "Sorties de la semaine"

3.6 Quelques difficultés rencontrées

- 1 Nous avons commencé par scraper 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 construisons la liste des pays à partir des informations des films et non pas à partir de cette liste, ensuite, pour chaque nouveau scrapin nous ajoutons dans la table 'pays' les pays ne s'y trouvant pas déjà pour éviter les doublons. Cette méthode est plus robuste.

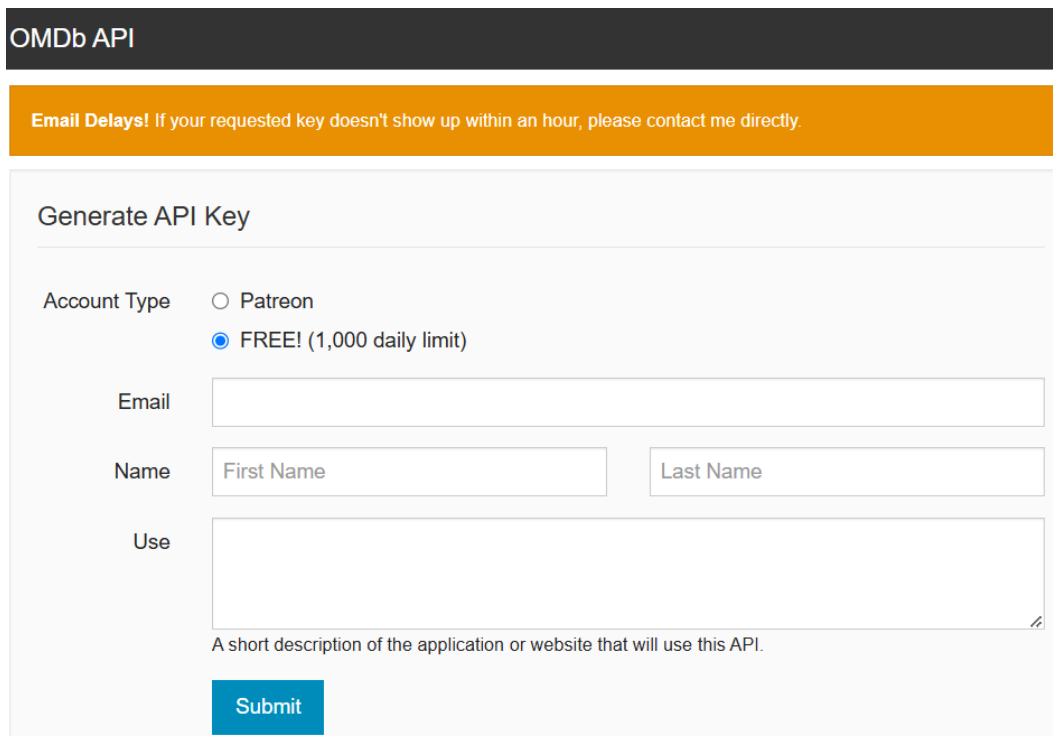
- 2 Les films récents n'ont pas de section "ratings" ou bien ont des sections "ratings" vides, des tests supplémentaires ont été ajoutés dans la phase de web scraping.
- 3 Les films récents ont parfois une section "rating" mais avec zéro avis utilisateur alors que jusqu'à présent nous ne prenions que les films avec au moins 30 avis, des options supplémentaires ont été créées et sont transmises pour
- 4 Lors du web scraping des films à l'affiche début 2025, certains films avaient en fait une année de production "2024" et avaient déjà été scrapés. Un nettoyage des doublons a été fait à partir de la base MySQL, ce phénomène ne peut plus arriver car désormais nous scrapons uniquement les films à l'affiche (et non pas des années entières).

4 Requête d'une API publique

4.1 Présentation de l'API OMDB

Nous avons choisi l'api publique [OMDB](#) pour collecter des informations de résumé de films et d'url d'affiche de films, à noter que ces informations de texte et d'url sont non-structurés et seront destinés à aller en base NoSQL.

Pour accéder à l'API d'OMDB nous demandons une clé dans la section dédiée, cela nous offre un accès gratuit à 1000 requêtes par jour.



The screenshot shows the 'OMDb API' website. At the top, there's a dark header with 'OMDb API' in white. Below it is an orange banner with the text 'Email Delays! If your requested key doesn't show up within an hour, please contact me directly.' The main content area is titled 'Generate API Key'. It features a form with the following elements: 'Account Type' with radio buttons for 'Patreon' and 'FREE! (1,000 daily limit)' (which is selected); an 'Email' input field; a 'Name' section with 'First Name' and 'Last Name' input fields; a 'Use' input field for a description; and a blue 'Submit' button at the bottom. A small note below the 'Use' field says 'A short description of the application or website that will use this API.'

La documentation d'OMDB nous indique la forme des requêtes.

Par exemple pour le film **In The Lost Lands**, la requête sera :

`https://www.omdbapi.com/?apikey=xxxx&t=in+the+lost+lands`

A l'aide de l'utilitaire **requests** nous obtenons le résultat sous forme d'un json.

```
1 { 'Title': 'In the Lost Lands',
2   'Year': '2025',
3   'Rated': 'R',
4   'Released': '07 Mar 2025',
5   'Runtime': '101 min',
6   'Genre': 'Action, Adventure, Fantasy',
7   'Director': 'Paul W.S. Anderson',
8   'Writer': 'Constantin Werner, Paul W.S. Anderson, George R.R. Martin',
9   'Actors': 'Milla Jovovich, Dave Bautista, Arly Jover',
10  'Plot': 'A queen sends the powerful and feared sorceress Gray Alys to the
        ghostly wilderness of the Lost Lands in search of a magical power, where
        the sorceress and her guide, the drifter Boyce, must outwit and outfight
        man and demon.',
11  'Language': 'English',
12  'Country': 'Germany, Canada, United States',
13  'Awards': 'N/A',
```

```
14 | 'Poster': 'https://m.media-amazon.com/images/M/  
    MV5BOWYxYjEyYTUtY2FkZC00Y2QwLTk1ZjMtMTAyMTAyMzQ3MDZiXkEyXkFqcGc@.  
    _V1_SX300.jpg',  
15 | 'Ratings': [{'Source': 'Internet Movie Database', 'Value': '5.4/10'}],
```

4.2 Récupération des informations

Nous disposons d'un Dataframe Pandas avec la liste des films ainsi que les informations déjà scrapées, pour requêter l'api OMDB nous créons une fonction qui crée une requête OMDB au bon format à partir du titre du film puis une fonction qui requêtes l'api OMDB et insère les données collectées dans de nouvelles colonnes du DataFrame.

A l'issue de cette étape les données sont enregistrées dans des fichiers CSV.

5 Nettoyage et agrégation des données

5.1 Nettoyage des données

Avant d'insérer les données en bases, nous devons nous assurer qu'elle sont au bon format.

Exemples de nettoyage

- 1 Les informations scrapées concernant les catégories de films, les acteurs, réalisateurs et compositeurs ont été mis sous la forme d'une chaîne de caractères où les entités sont séparées par une virgule.

`"Alain Delon,Olga Georges-Picot,Charles Bronson,Brigitte Fossey,Bernard Fresson,`

`Jean-Paul Tribout,Ellen Bahl,Stéphane Bouy"` Nous transformons cette chaîne de caractères en liste python de chaînes de caractères

`["Alain Delon", "Olga Georges-Picot" ...]` en nous assurant qu'il n'y a pas de doublon.

- 2 Certaines informations sont manquantes sur le site allociné (durée du film, nom du réalisateur, date de sortie et même la liste des acteurs pour les films d'animation etc), les champs dans nos Dataframe Pandas sont alors représentés par des `NaN` en python, nous les remplaçons par des chaînes de caractères vides ou bien par des valeurs nulles selon le type de champs souhaités.
- 3 Les durées sont stockées sous forme de chaînes de caractères de la forme `1h 35min`, nous les transformons en durée en minutes (95) de type **entier**.
- 4 Les notes sont stockées sous la forme d'une chaîne de caractères `"3,5"` où parties entière et décimale sont séparées par une virgule, nous remplaçons les virgules par des points.
- 5 Les dates de sortie de film sont stockées au format **5 mars 2025** sont converties au type Pandas **datetime**, pour se faire nous devons au préalable convertir les mois du français vers l'anglais.

A noter qu'un nettoyage a déjà été fait lors du scrapping.
Quel est le type des données dans mongodb ?

6 Création de la base de données

6.1 Introduction au SGBD

Un SGBD (Système de Gestion de Base de Données) est un logiciel prévu pour stocker, manipuler et extraire des données.

Pour ce projet, nous avons choisi les SGBD :

- **MySQL** pour toutes **les données structurées** c'est-à-dire les données numériques ou catégorielles, les entités concernées (films, acteurs, réalisateurs, compositeurs, catégories) ont des relations étroites et des requêtes de jointures ou de filtres complexes seront nécessaires, **MySQL** semble adaptée à ce besoin,
- **MongoDB** pour toutes **les données non-structurées** comme du texte ou des urls, dans notre projet il s'agira des synopsis des films et des urls vers les affiches de films, données qui sont uniquement liées avec l'entité film et ne nécessite donc pas de requête complexe.

Avantages et inconvénients SQL, NoSQL à développer

6.2 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"), il est indépendant du SGBD choisi.

MPD : Modèle physique de données ("Le MPD, c'est là où la théorie se transforme en lignes de code fonctionnelles"), il dépend du SGBD choisi, il représente les tables, les colonnes, les clés primaires et étrangères, on y précise les types de données, les contraintes etc...

Pour concevoir un **MCD** nous devons :

- comprendre "le métier", c'est-à-dire connaître les entités mises en jeu (ici les films, acteurs, catégories ...),
- identifier leurs attributs (titre, nom etc ...),
- comprendre si ces attributs sont de type catégoriels ou quantitatifs,
- identifier les relations entre les entités,
- définir les cardinalités dans ces relations (c'est-à-dire le nombres d'entités qui peuvent reliés à une autre entité).

Ensuite nous pouvons passer à la conception de notre **MCD**

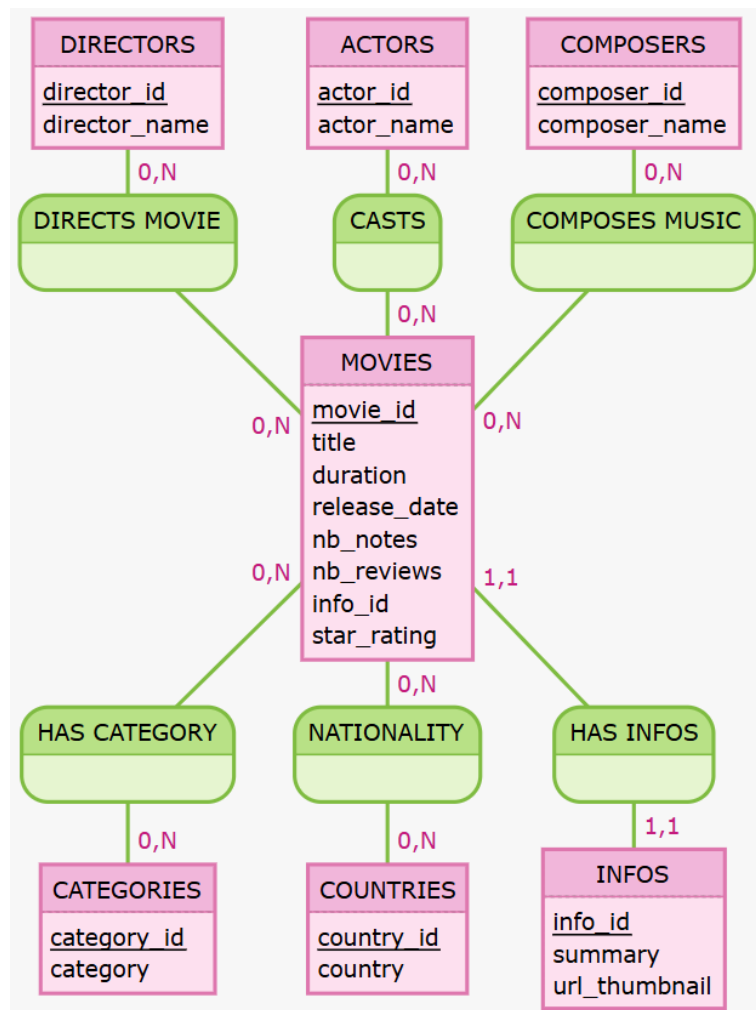


FIGURE 10 – MCD



FIGURE 11 – MPD

6.3 Création de la base et des tables

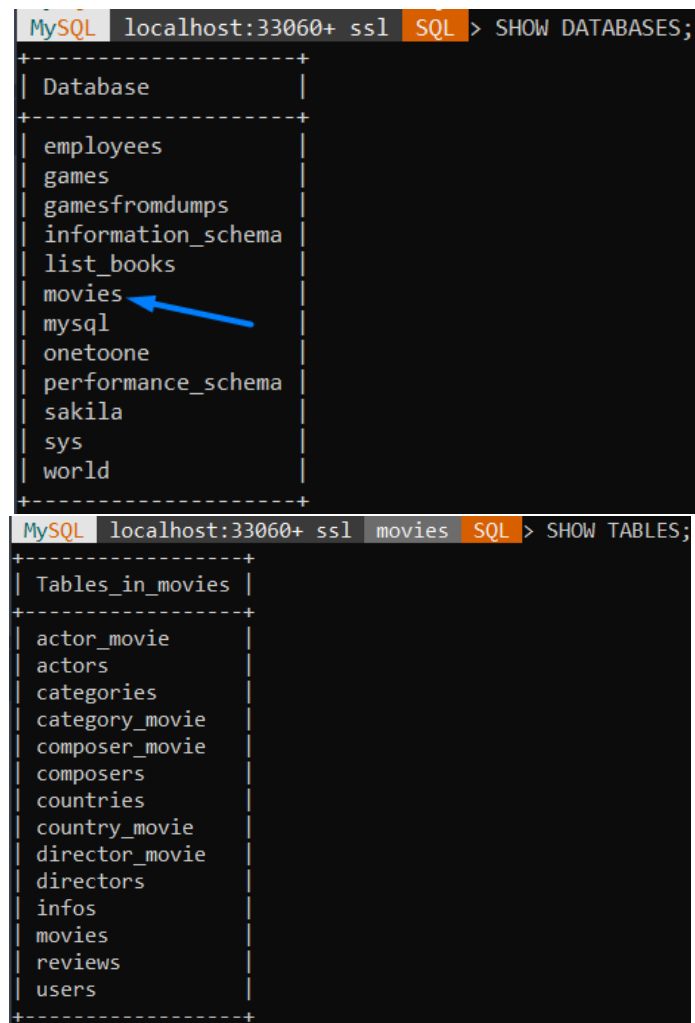
`sql` pour passer en mode SQL

`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.



The screenshot shows two MySQL command-line sessions. The first session shows the output of the `SHOW DATABASES;` command, listing various databases including `employees`, `games`, `gamesfromdumps`, `information_schema`, `list_books`, `movies` (highlighted with a blue arrow), `mysql`, `onetoone`, `performance_schema`, `sakila`, `sys`, and `world`. The second session shows the output of the `SHOW TABLES;` command within the `movies` database, listing tables such as `actor_movie`, `actors`, `categories`, `category_movie`, `composer_movie`, `composers`, `countries`, `country_movie`, `director_movie`, `directors`, `infos`, `movies`, `reviews`, and `users`.

```
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 |
+-----+
```

6.4 Aggrégation et nettoyage des données

On s'assure que les catégories ne sont pas déjà existantes

6.5 Questions

****Générer automatiquement des IDs chaînes de caractères ?****

****Les requêtes faites à partir de notre API doivent avoir accès uniquement à la base en lecture seule****

****Comment faire le lien entre table SQL Movie et NoSQL infos ? (info_id?) * ***

****Comment créer les tables infos et movies ? Faut il vraiment un ID commun ?****

6.6 Exemples de requêtes SQL

7 Automatisation

7.1 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).

7.2 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.

7.3 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

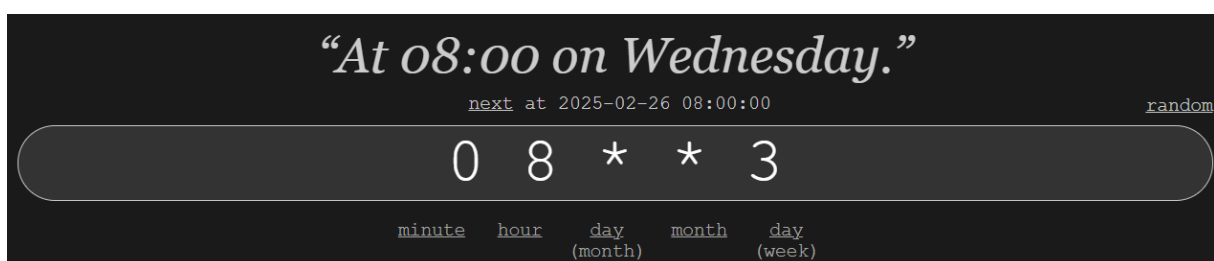
```
pip install pandas requests beautifulsoup4 mysql-connector-python
```

7.4 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

8 Création d'une API

8.1 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

8.2 ...

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

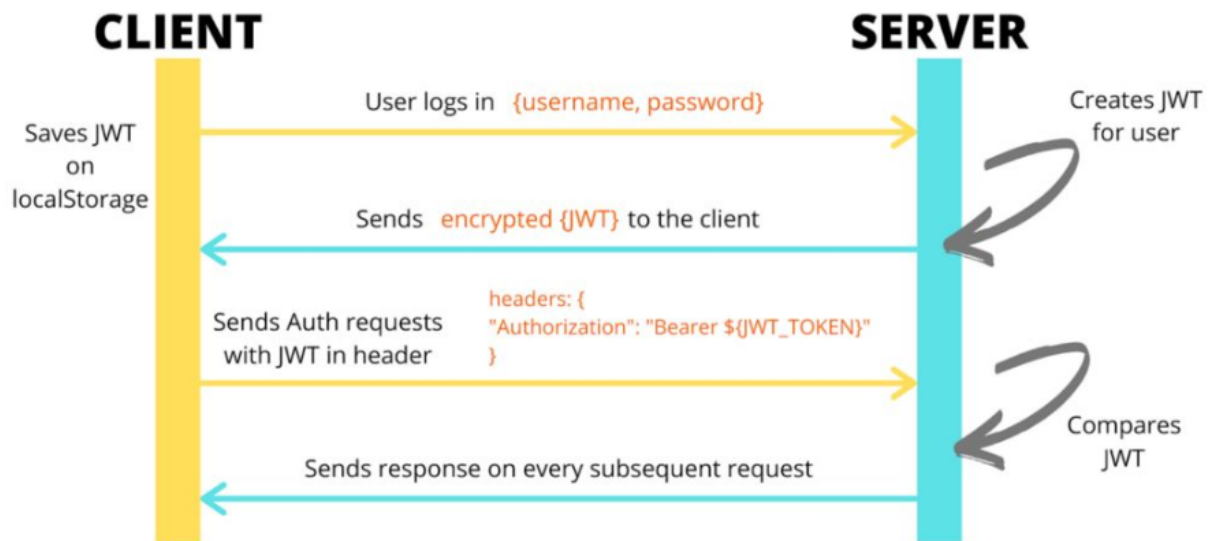
JWT (JSON Web Token) est un standard pour communiquer des données entre différentes parties à travers des objets JSON. L'information est signée numériquement.

<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.

8.3 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 MongoDB.

8.4 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/MV5BZmE0YzI5M2Q1MGNI100WjBmLWE3MmVtMQzNGVjMkY0YTFaXkkyXkFqcGc@_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