

# Projet Allociné



Franck Le Fur

# Table des matières

<b>1</b>	<b>Présentation</b>	<b>2</b>
1.1	Présentation du projet . . . . .	2
<b>2</b>	<b>Preliminaires</b>	<b>3</b>
2.1	Création de l'environnement . . . . .	3
2.1.1	Création d'un nouvel environnement . . . . .	3
2.1.2	Installation des packages . . . . .	3
<b>3</b>	<b>Webscrapping</b>	<b>4</b>
3.1	Webscrapping . . . . .	4
3.2	Utilisation de Selenium . . . . .	4
<b>4</b>	<b>Requêtage d'une API publique</b>	<b>6</b>
4.1	API OMDB . . . . .	6
4.2	... . . . .	6
<b>5</b>	<b>Création de la base de données</b>	<b>7</b>
5.1	Modèles MCD et MPD . . . . .	7
5.2	Création de la base et des tables . . . . .	8
<b>6</b>	<b>Automatisation</b>	<b>9</b>
6.1	Création d'une tâche Crontab . . . . .	9
<b>7</b>	<b>Création d'une API</b>	<b>10</b>
7.1	Fast API . . . . .	10
7.2	... . . . .	10

# 1 Présentation

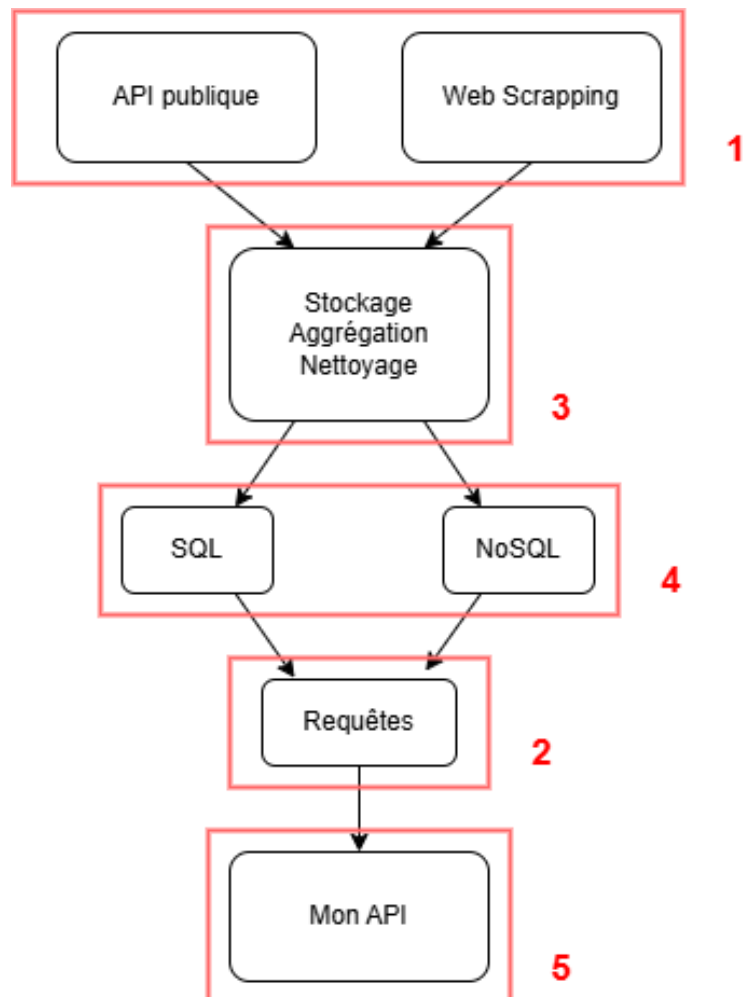
## 1.1 Présentation du projet

Nous présentons une application proposant des filtres avancés sur le cinéma, par exemple filtrer les films :

- dans lesquels ont joué deux acteurs (ou plus),
- dans lesquels ont joué un acteur et le film a été produit par un producteur,
- dans lesquels ont joué un acteur et dont la musique a été composé par telle personne,
- ...

Cette application peut être proposée à des professionnels du cinéma, comme des critiques de films, ayant besoin de filtres avancés.

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



## 2 Préliminaires

### 2.1 Création de l'environnement

#### 2.1.1 Création d'un nouvel environnement

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

#### 2.1.2 Installation des packages

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

## 3 Webscrapping

### 3.1 Webscrapping

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).

#### Filtres

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

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 directement 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

### 3.2 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 Beautiful Soup  
Résultat avec beautiful Soup

```
<li class="filter-entity-item">
  <span class="ACrL2ZACrpbG1zL2RlY2VubmllLTlwMzAv item-content" title="2030 - 2039">
    2030 - 2039
  </span>
  <span class="light">
    (47)
  </span>
</li>
<li class="filter-entity-item">
  <span class="ACrL2ZACrpbG1zL2RlY2VubmllLTlwMjAv 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.

## 4 Requête d'une API publique

### 4.1 API OMDB

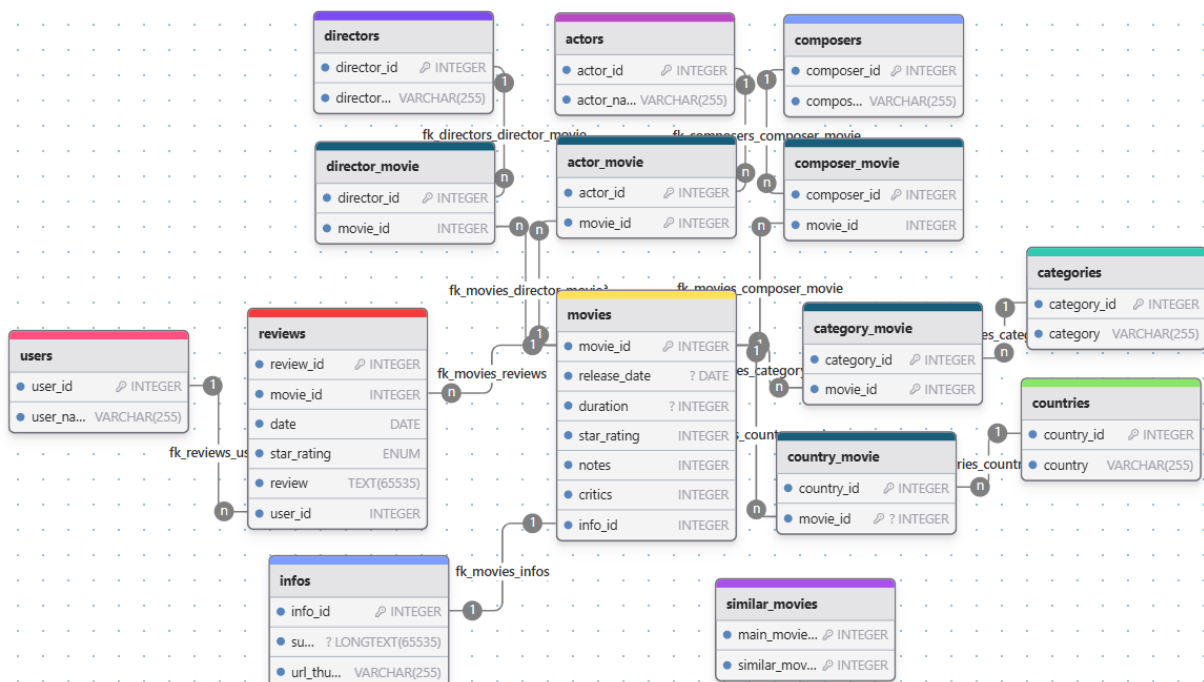
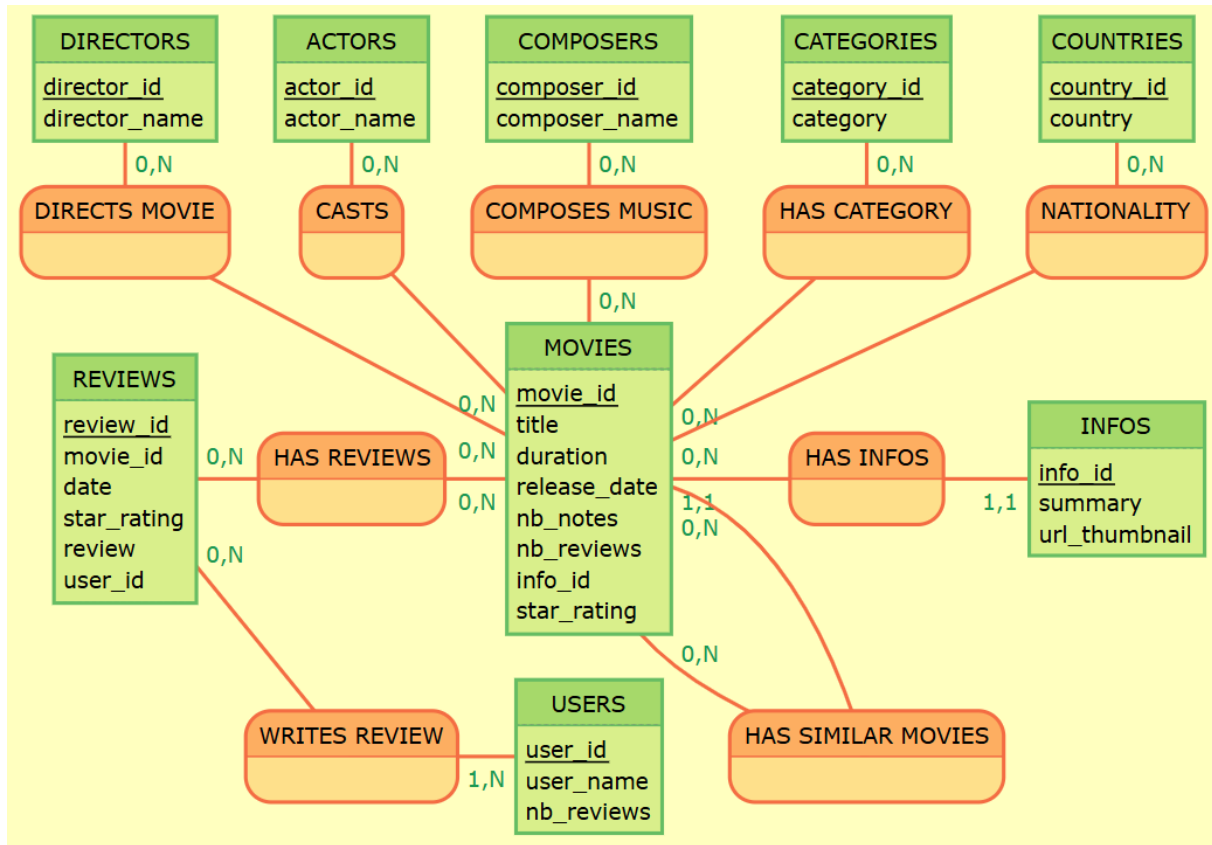
### 4.2 ...

## 5 Création de la base de données

### 5.1 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")





## 5.2 Création de la base et des tables

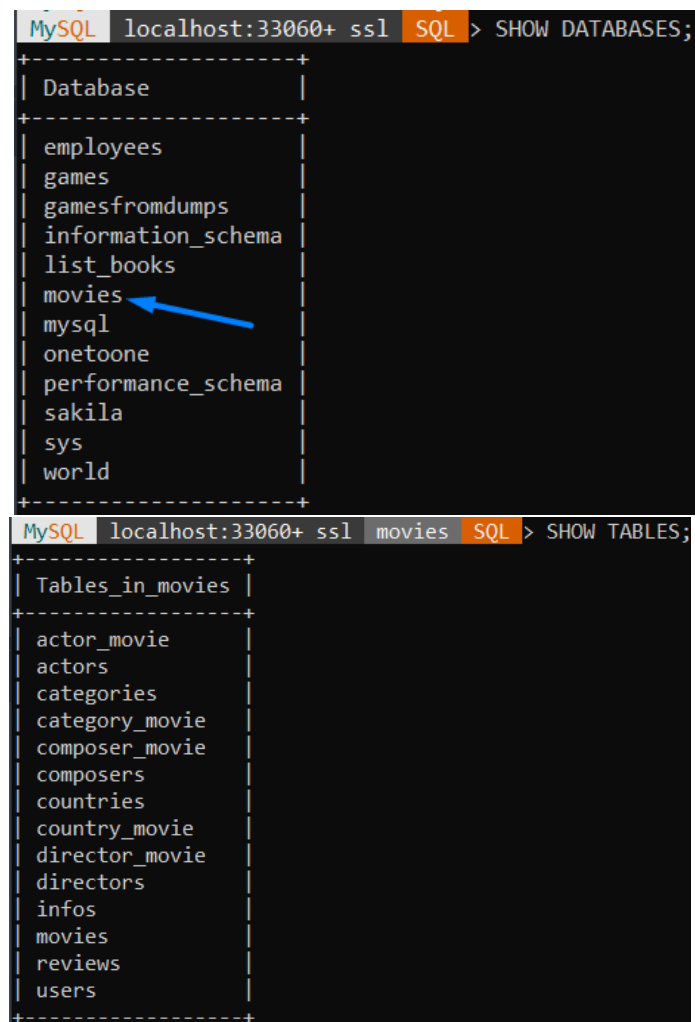
`sql` pour passer en mode SQL

`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 image shows two screenshots of a MySQL command prompt. The top screenshot 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 bottom screenshot 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`.

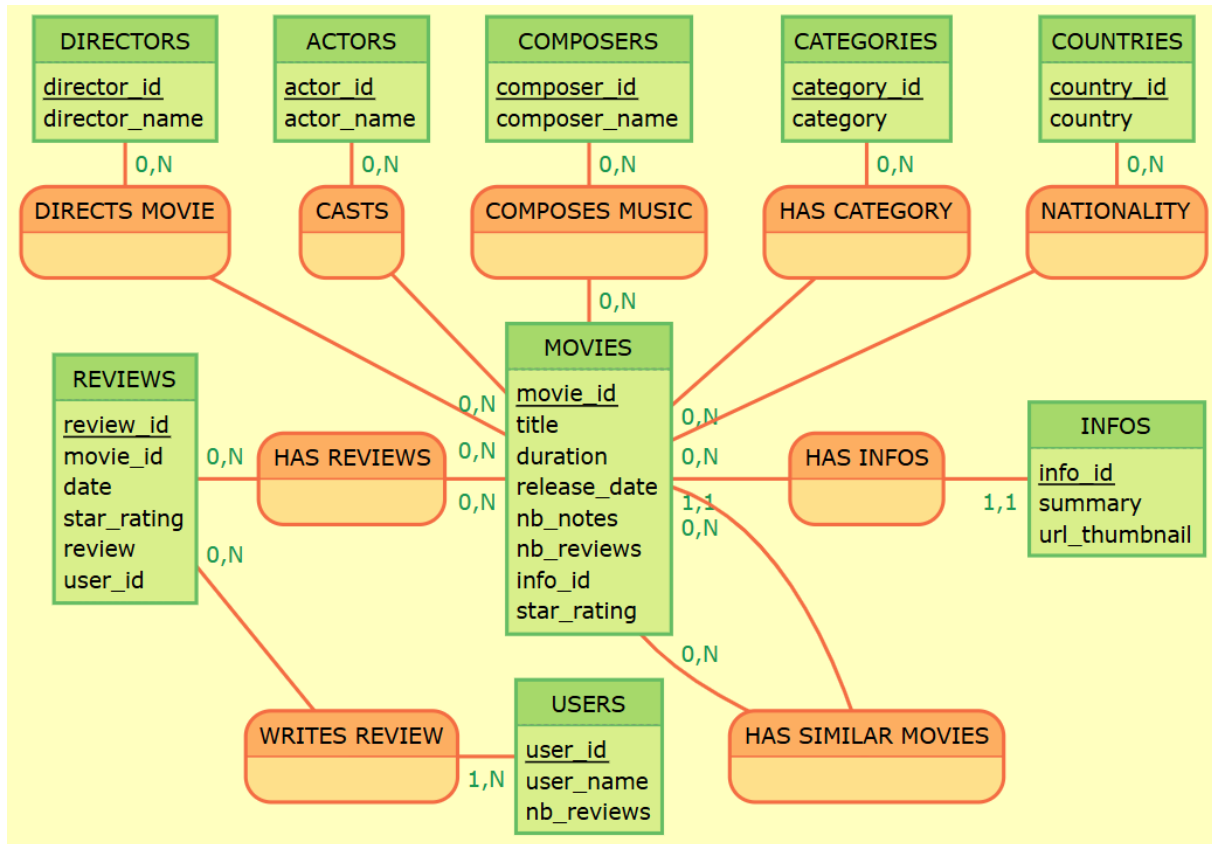
```
MySQL localhost:33060+ ssl SQL > SHOW DATABASES;
+-----+
| Database |
+-----+
| employees |
| games |
| gamesfromdumps |
| information_schema |
| list_books |
| movies |
| mysql |
| onetoone |
| performance_schema |
| sakila |
| sys |
| world |
+-----+

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 Automatisation

### 6.1 Création d'une tâche Crontab

Blabla



## 7 Création d'une API

### 7.1 Fast API

### 7.2 ...