Compte Rendu Technique - Livrable 4

Projet CinéExplorer : Architecture Hybride SQL/NoSQL

DRAVET Timothée - FISA INFO 4A

Janvier 2026

Résumé

Ce rapport présente l'aboutissement du projet CinéExplorer. L'enjeu technique réside dans l'exploitation simultanée d'une base relationnelle (SQLite) et d'une base orientée documents (MongoDB) configurée en Replica Set, afin d'optimiser les performances de recherche et la disponibilité des données.

Table des matières

1 Introduction

Le projet CinéExplorer est une application web développée avec Django. Sa particularité repose sur une architecture hybride exploitant les forces de deux paradigmes de stockage : le relationnel pour la structure des recherches et le NoSQL pour la performance sur la lecture d'objets complexes.

Le code source complet est accessible sur GitHub : github.com/timotheedvt/cinexplorer.

2 Architecture Globale du Système

L'écosystème repose sur trois piliers technologiques majeurs :

— **Django** : Orchestre la logique métier et assure l'interface entre les différentes sources de données.

— **SQLite** : Utilisé pour les fonctionnalités de recherche textuelle (`LIKE`) et les listes filtrées (genres, années), où l'indexation relationnelle est nativement efficace.

— **MongoDB Replica Set** : Une grappe de 3 nœuds stocke les documents structurés. Cette approche évite les jointures SQL coûteuses lors de l'affichage détaillé d'un film.

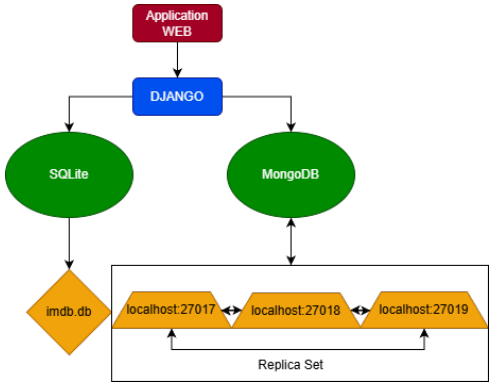


FIGURE 1 – Architecture hybride et réplication MongoDB

3 Choix Technologiques Justifiés

3.1 Haute Disponibilité

L'implémentation d'un **Replica Set** pour MongoDB garantit la tolérance aux pannes. En cas de défaillance du nœud primaire, une élection automatique désigne un nouveau leader, assurant la continuité de service.

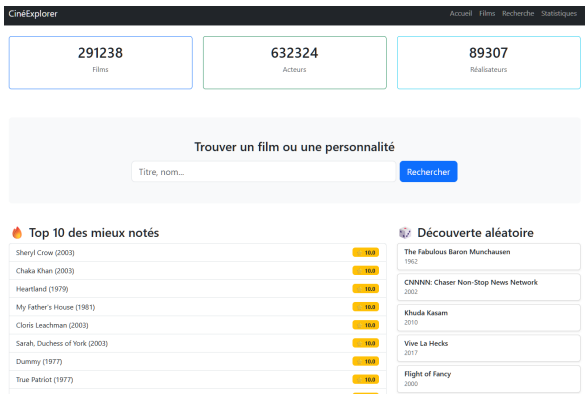
3.2 Performance et Dénormalisation

La stratégie multi-bases permet de passer d'un modèle normalisé (SQLite) à un modèle dénormalisé (MongoDB). Pour afficher une fiche film complète, SQLite nécessiterait plusieurs jointures complexes, tandis que MongoDB récupère l'intégralité des données (casting, titres, détails) en une seule lecture.

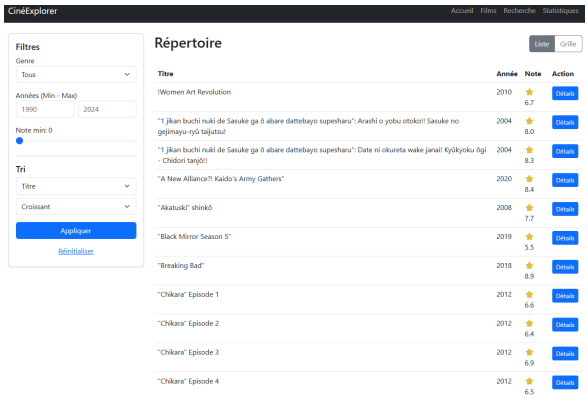
4 Description des Fonctionnalités

4.1 Interface Utilisateur

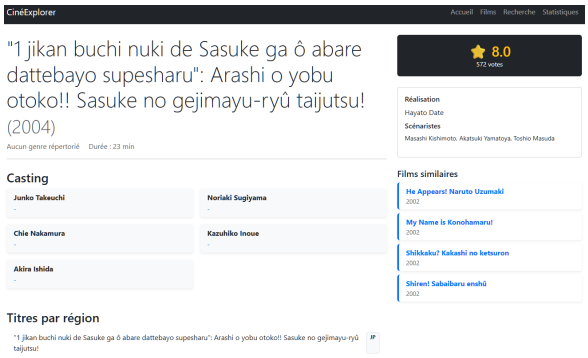
Conçue avec **Bootstrap 5**, l'interface est totalement responsive. Le tableau de bord statistique intègre **Chart.js** pour une visualisation dynamique des données agrégées.



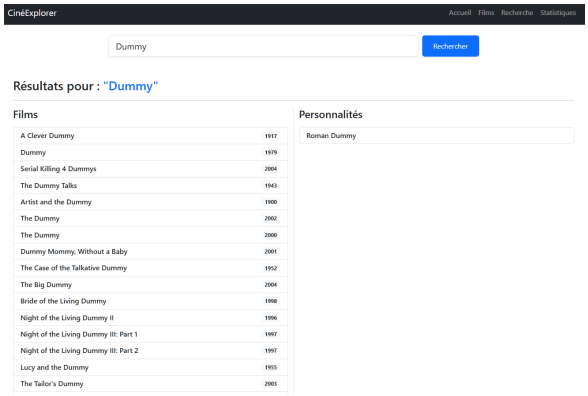
(a) Page d'accueil



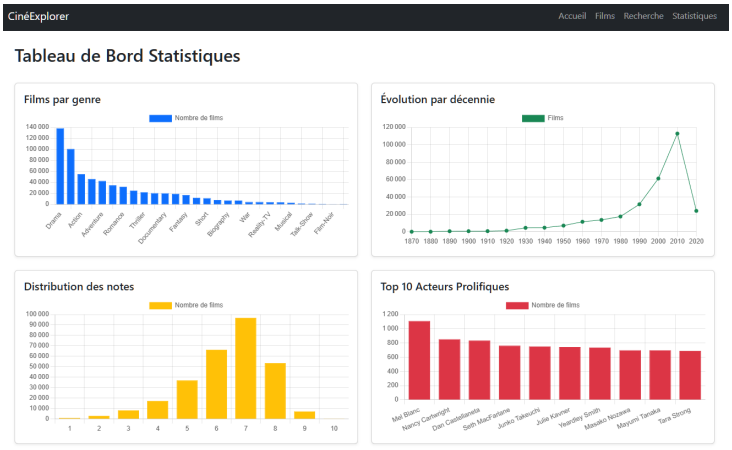
(b) Répertoire des films



(c) Détails d'un film (MongoDB)



(d) Moteur de recherche (SQLite)



(e) Statistiques (Chart.js)

FIGURE 2 – Aperçu des interfaces de l'application Django CinéExplorer

5 Implémentation Technique

```python
def get_rating_distribution():
    db = get_mongo_client()
    pipeline = [
        {"$match": {"rating.average": {"$exists": True}}},
        {"$group": {
            "_id": {"$floor": "$rating.average"},
            "count": {"$sum": 1}
        }},
        {"$sort": {"_id": 1}}
    ]
    results = list(db.MOVIE_COMPLETE.aggregate(pipeline))
    return [{'label': r['_id'], 'value': r['count']} for r in results]
```

Listing 1 – Exemple d'agrégation MongoDB pour les statistiques

6 Benchmarks de Performance

Les tests confirment l'intérêt de l'architecture hybride sur les temps de réponse.

Opération	**SQLite (ms)**	**MongoDB (ms)**	**Gain**
Détail complet (Casting + Titres)	~150ms	**~40ms**	x3.7
Agrégation statistique (Genres)	~800ms	**~200ms**	x4.0
Recherche textuelle simple	**~15ms**	N/A	-

TABLE 1 – Comparaison des performances moyennes constatées

7 Difficultés et Solutions

Problème : Lenteur des statistiques : Le calcul des agrégations bloquait le rendu de la page. *Solution* : Implémentation d'un point d'accès API asynchrone et d'un *skeleton screen* pour améliorer l'expérience utilisateur.

Problème : Accès aux IDs MongoDB dans Django : Les templates Django n'autorisent pas l'accès aux variables commençant par un souligné (`_id`). *Solution* : Création d'un template filter personnalisé `get_id` pour extraire proprement l'identifiant.

8 Conclusion

CinéExplorer démontre la complémentarité des mondes SQL et NoSQL. En combinant la flexibilité de MongoDB pour les objets complexes et la rapidité de SQLite pour le relationnel, l'application offre une solution performante, robuste et hautement disponible.