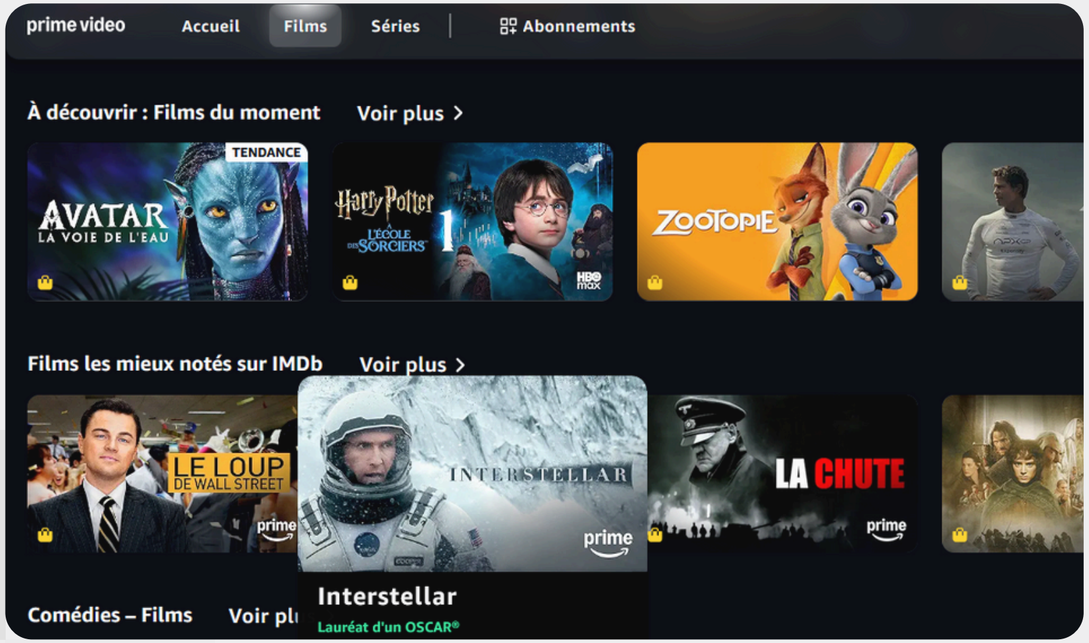
Recommandation de films



BRUXELLES FORMATION
former pour l'emploi

Présentation

Elena Rottin

- Historienne de l’art de formation
- Longue parenthèse en gestion administrative
- Reconversion vers Data et Développement

Guillaume Konen

- Diplômé en sciences mathématiques
- Développeur Junior Android
- Reconversion en Développeur Python IA

Nos objectifs

Nous voulions un projet applicatif complet abordant la plupart des matières vues durant la formation dans un cadre concret réalisable dans un délai de 10 jours.

→ Nous avons créer un site local où un utilisateur pouvait se connecter et noter les films du catalogue.

→ Nous avons créer un serveur local pour récupérer les données ainsi que fournir une liste de recommandation de films

Pourquoi ce sujet ?

→ réalisable dans les temps, tout en étant assez complexe pour pouvoir mettre à défi toutes les compétences apprises

→ bien documenté et transposable à n'importe quel autre sujet

→ concret, car se réfère à des techniques utilisées par plusieurs plateformes

Contexte

Nous cherchions un jeu de données assez important pour être confronté à une situation réelle et relever les défis techniques associés durant notre temps imparti.

Nous avons choisi la recommandation de films grâce à **GroupLens** qui a mis à disposition un jeu de données important provenant du site MovieLens. Tout notre projet s'est organisé autour de leurs données.

Plus particulièrement, nous avons pris le jeu de données **MovieLens 32M** :

- 32 millions de notes
- 2 millions de tags
- 87 mille films
- 200 mille utilisateurs

Collecté par le groupe le 10/2023 publié 05/2024
Téléchargé 12/2025

Stack technologique

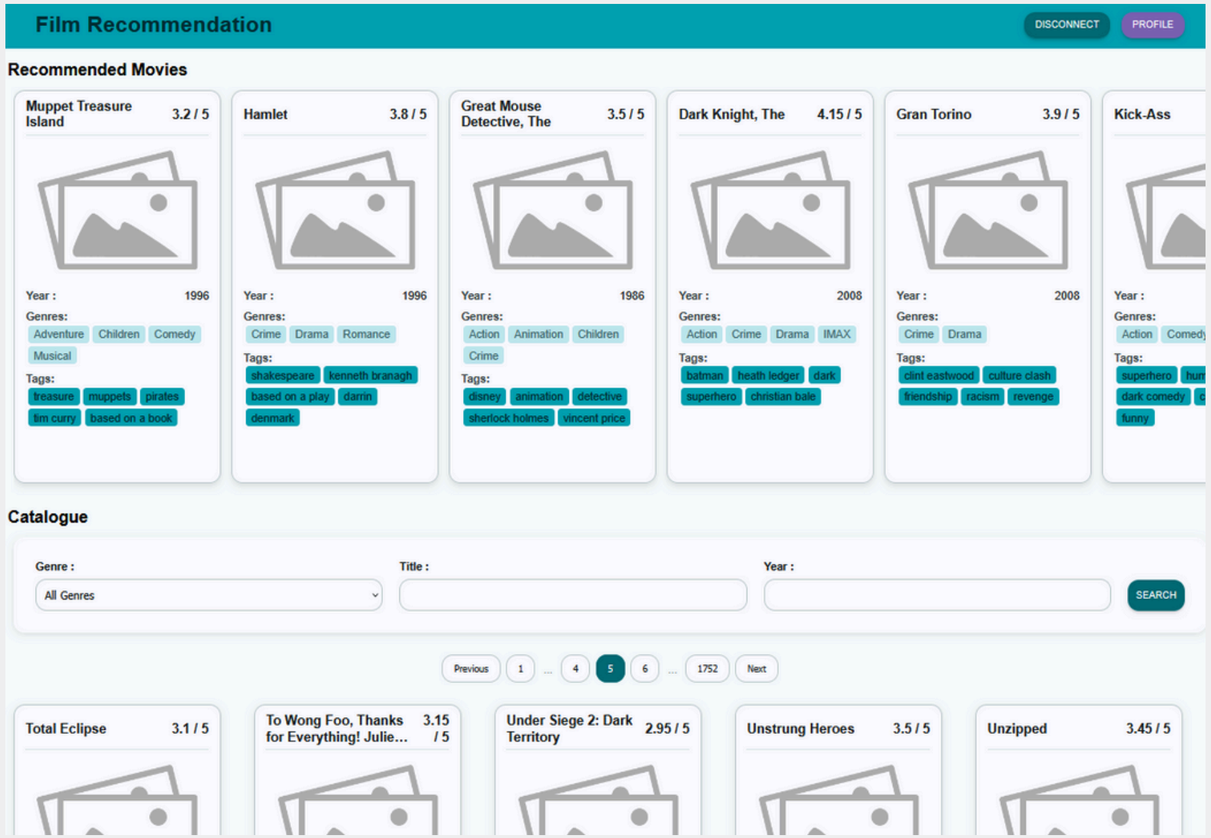
Le projet se compose de plusieurs couches avec leurs propres technologies.

- **Front** Angular 21 (*TypeScript*)
- **Back** Flask (*Python*)
- **Base de données**

  PostgreSQL
- **Librairies Python**
  - SQLAlchemy
  - scikit-learn
  - Pandas
  - NumPy

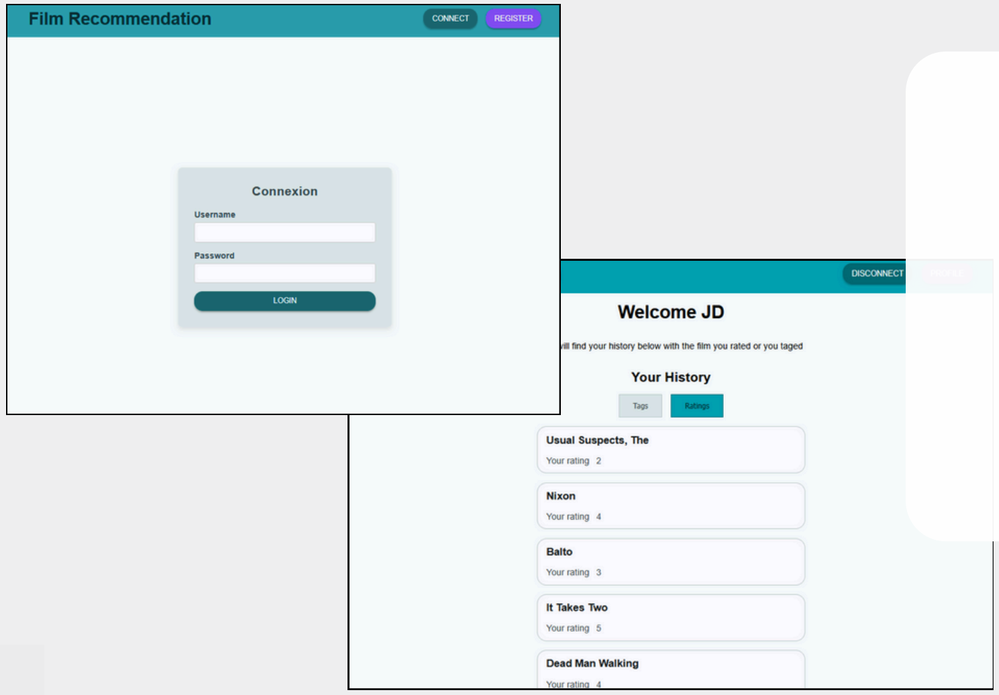
Front-end

Site local développé en Angular 21 en respectant les pratiques abordées durant la formation : *Atomic design*.



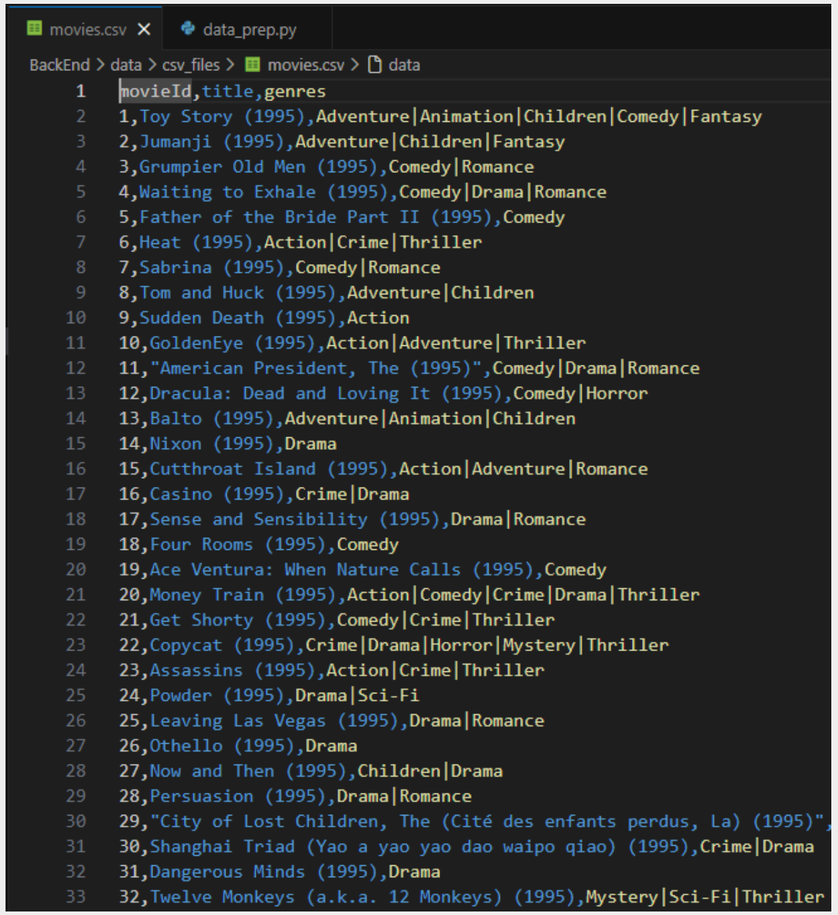
Front-end

Notre objectif était d'avoir un site pour exposer nos données, nous avons implémentées les fonctionnalités minimales pour respecter les délais. Beaucoup de travail est encore nécessaire.



- page principale avec le catalogue des films et les recommandations (pour l'utilisateur connecté)
- page de profil de l'utilisateur, avec son historique des interactions.
- pages de création / connexion compte utilisateur

Back-end & base de données

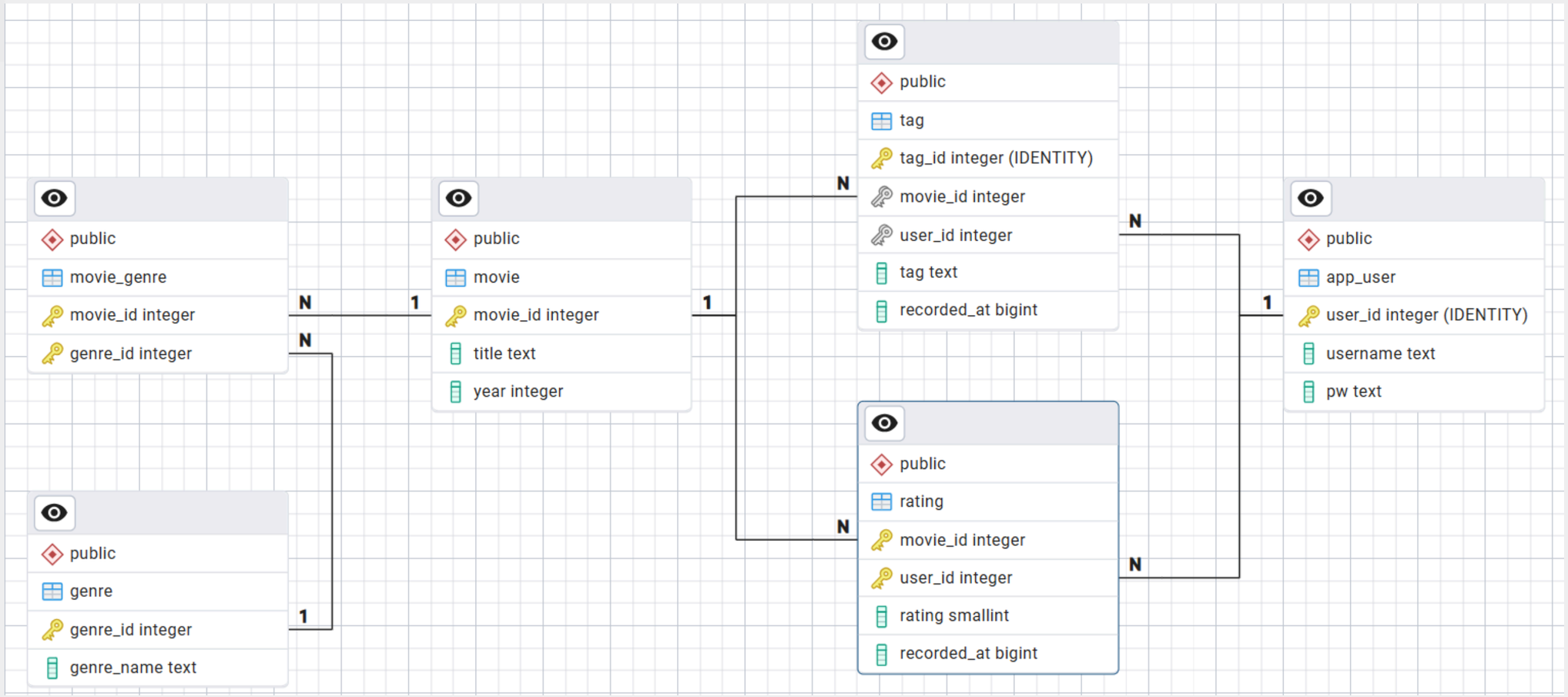


Méthodologie

- Exploration des données
- Définition du schéma entité-association
- Prétraitement des données
- Script de création de la base de données et de remplissage des tables de la BD

Back-end & base de données

Schéma entité-association

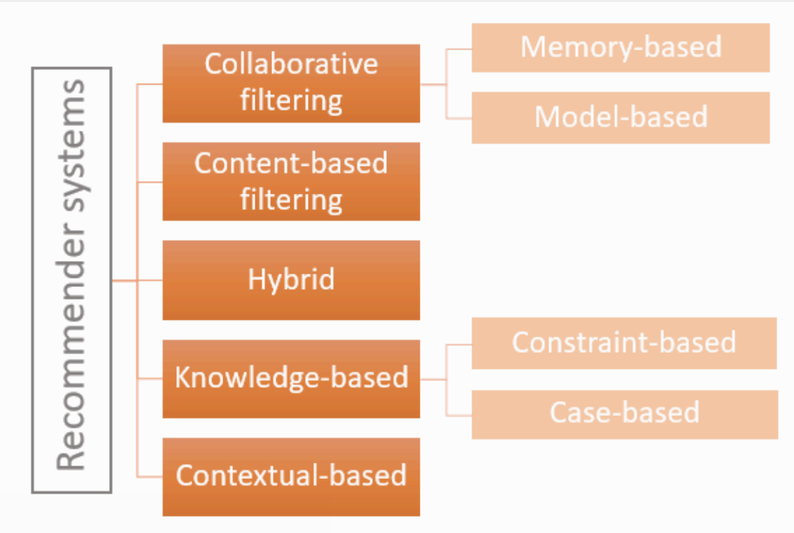


Back-end & base de données

Méthodologie

- Flask, micro-framework de développement web
- SQLAlchemy, toolkit SQL et ORM (mapping objet-relationnel)
- architecture par couches:
  - Model
  - Repository
  - Controller
  - Route
- problème de la pagination:
  - sans pagination, le serveur devrait envoyer tous les 87K+ films en une seule fois → très lent et inefficace
  - pagination implémentée grâce aux méthodes “offset” et “limit” de SQLAlchemy

Système de recommandation



Source: https://www.smals.be/fr/content/introduction-aux-systemes-de-recommandation

collaborative filtering

les recommandations sont basées sur les préférences (ratings, actions) de l'utilisateur et de ses pairs

content-based filtering

les recommandations sont basées sur les préférences/le profil de l'utilisateur et les caractéristiques de l'objet recommandé

knowledge-based

les recommandations sont basées sur les spécifications de l'utilisateur et les caractéristiques de l'objet recommandé

approche hybride

combinaison des approches précédentes

Système de recommandation

Méthodologie

Pour respecter les délais, nous avons travaillé par itération en terme de complexité.

1re Approche *Content-Based*

Nous avons, dans un premier temps, étudié uniquement les propriétés intrinsèques des films pour notre système de recommandation.

Plusieurs choix possibles : titre, années et genre. Nous nous sommes focalisés sur le genre.

Pour déterminer la similarité, l'approche standard est d'utiliser le *cosine similarity*. Vu que nous traitons une liste de catégorie, nous avons jugé plus pertinent l'emploi du *coefficient d'Ochiai*.

Notre premier système de recommandation est de proposer les 20 films les plus proches avec les films les mieux notés de l'utilisateur.

Système de recommandation

Méthodologie

2e Approche *Collaborative Filtering*

Nous utilisons les interactions des utilisateurs, plus précisément les notes qu'ils ont donnés, pour pouvoir construire notre deuxième système. L'objectif est de trouver des utilisateurs similaires pour pouvoir recommander les films qu'ils ont bien notés que notre utilisateur n'aurait pas vu.

Pour déterminer si deux utilisateurs sont similaires, nous avons collectés leurs notes pour déterminer quels genres ils ont tendances à mieux noter.
Nous obtenons un vecteur de préférence par genre qui est utilisé pour "profiler" l'utilisateur.

Nous avons alors utiliser le modèle *Nearest Neighbors* de *scikit-learn* pour déterminer les utilisateurs qui ont le profil le plus proche et recommander les films qu'ils ont le plus notés à notre utilisateur.

Q&A

Remerciements

Merci pour votre attention!

Et un tout grand merci à...

→ Bruxelles Formation et notre coordinateur pédagogique David Jelgersma

→ L'ensemble de nos formateurs de qualités de chez BStorm

→ Anouchka Walha, notre coach Cefora

BRUXELLES FORMATION
former pour l'emploi

bstorm

CEFORA