

# Système de Recommandation de Films de Niveau Production

Découvrez vos prochains films préférés grâce à une architecture MLOps robuste et reproductible.

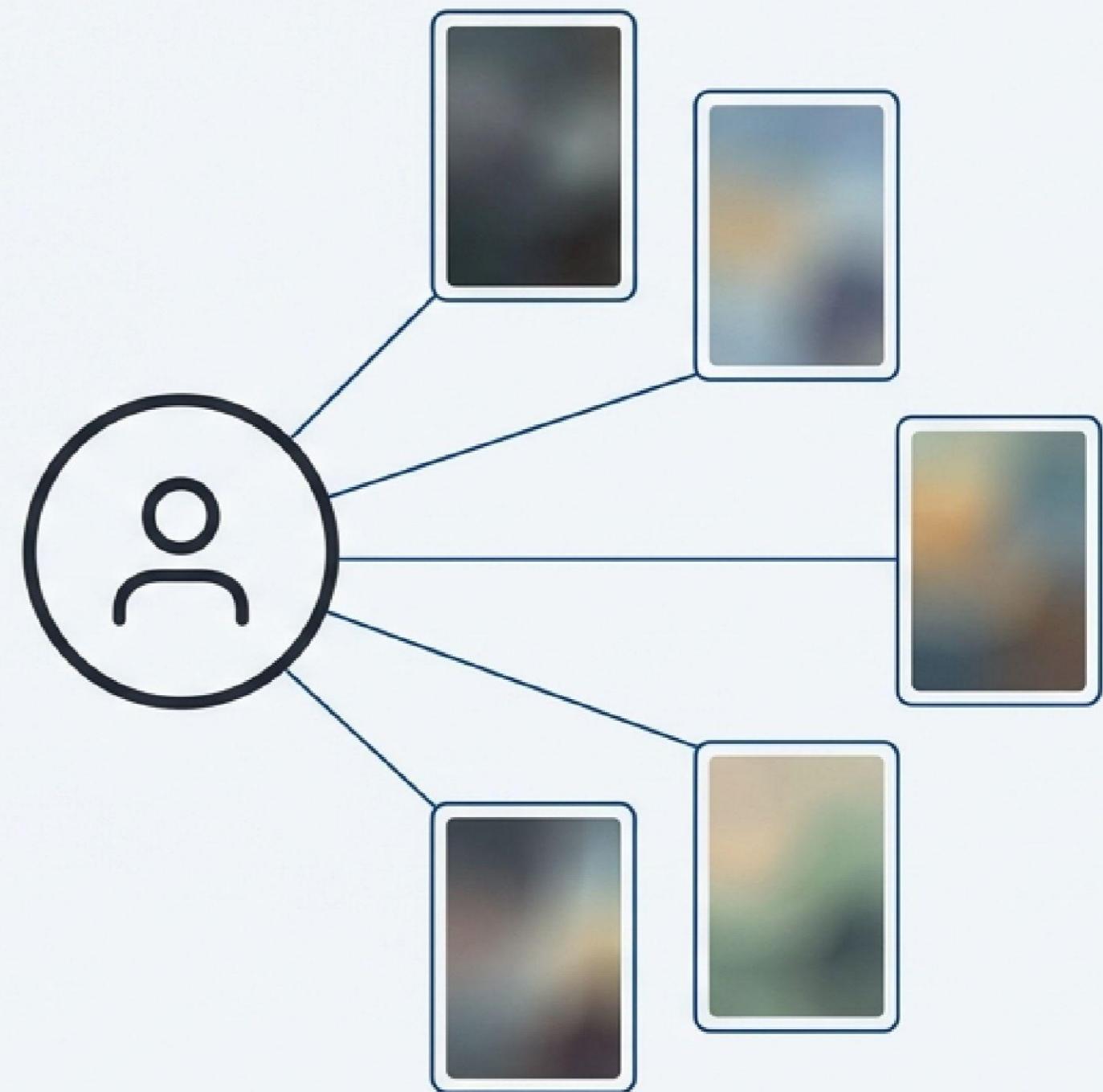
[Démo Live](#)[Documentation API](#)[Python 3.11](#)

Un projet de Souleymane SALL, Data Scientist / ML Engineer. 

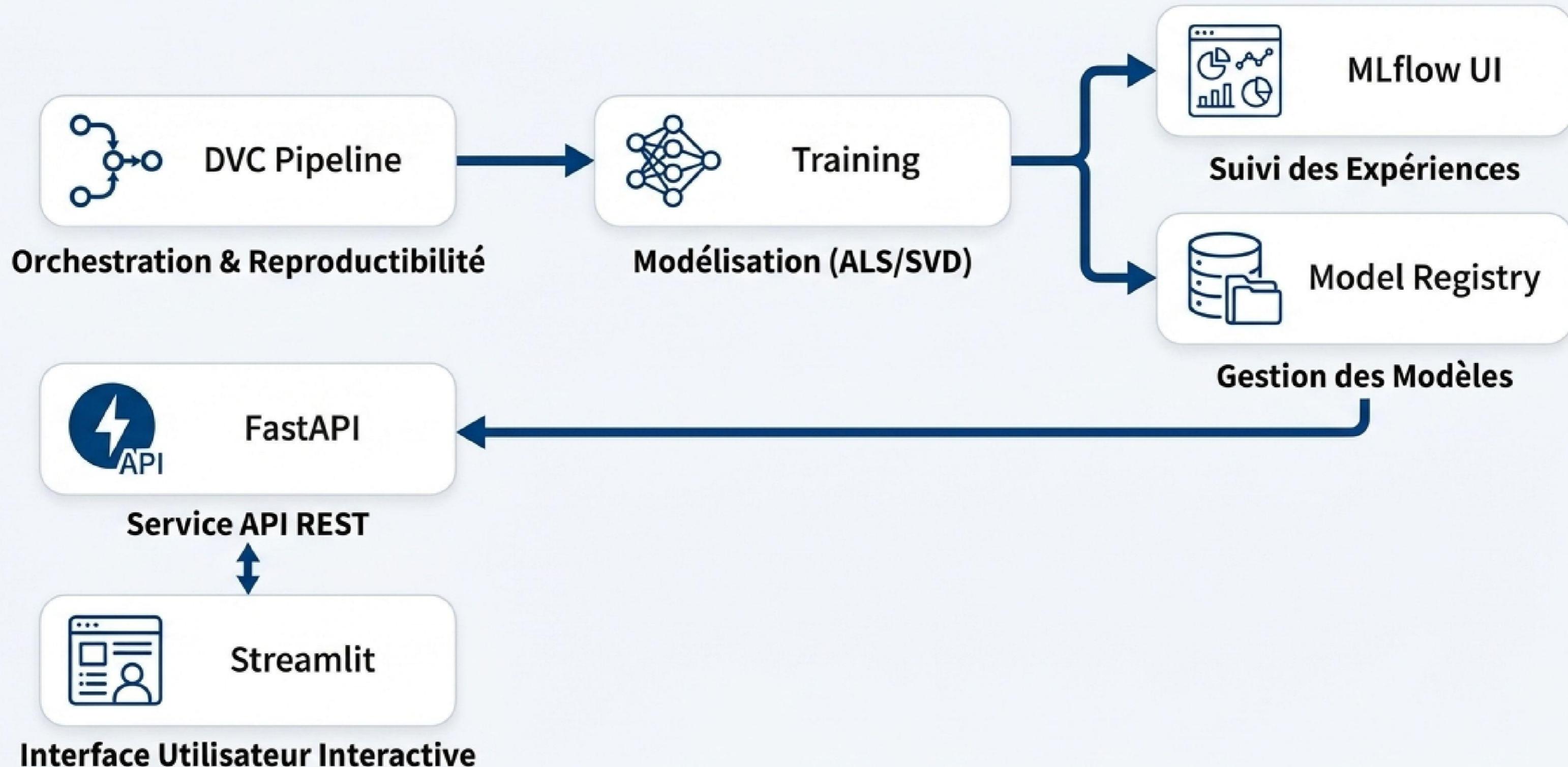
# Un moteur de suggestion intelligent qui apprend de vos goûts

Ce projet est un moteur de recommandation qui analyse les goûts des utilisateurs pour suggérer des films personnalisés. Le système apprend des préférences et recommande des contenus pertinents, à l'image des systèmes utilisés par Netflix ou Amazon.

| Scénario                          | Fonctionnement du système   |
|-----------------------------------|---|
| Un utilisateur a aimé "Inception" | Recommande des thrillers de science-fiction conceptuels similaires.             |
| Un nouvel utilisateur arrive      | Propose les films les plus populaires du moment pour initier son parcours.      |
| Un utilisateur explore            | Suggère : "Les gens qui ont aimé ce film ont aussi aimé..." pour la découverte. |



# L'architecture complète : de l'orchestration au déploiement



# Une stack technique moderne conçue pour la production

## Data Science & ML



**Python**  
Langage principal



**Pandas**  
Manipulation des données



**Scikit-learn**  
Utilitaires ML



**Algorithmes**  
Filtrage Collaboratif (ALS)

## MLOps



**DVC**  
Gestion des données et des pipelines



**MLflow**  
Suivi des expériences et registre de modèles



**Optuna**  
Optimisation des hyperparamètres

## Backend & Frontend



**FastAPI**  
API REST haute performance



**Pydantic**  
Validation des données



**Streamlit**  
Interface web interactive

## DevOps & Déploiement



**Docker**  
Conteneurisation



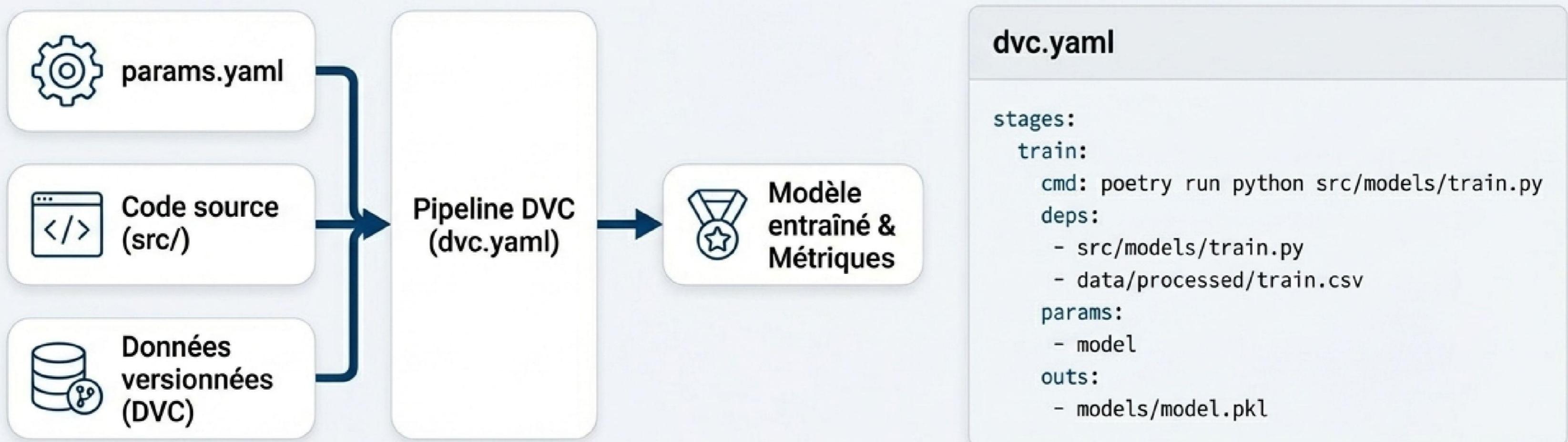
**GitHub Actions**  
Intégration et déploiement continu (CI/CD)



**Render**  
Hébergement Cloud

# La reproductibilité comme fondation avec DVC

Le pipeline est entièrement défini et versionné avec DVC, garantissant que chaque exécution, du traitement des données à l'entraînement, est 100% reproductible.



# Un suivi systématique des expériences avec MLflow et Optuna

Toutes les expériences sont tracées avec MLflow, capturant les paramètres, les métriques et les artefacts. Le registre de modèles MLflow gère le cycle de vie des modèles, de l'expérimentation à la production.

## Focus sur l'Optimisation

L'optimisation des hyperparamètres est automatisée avec Optuna pour trouver la meilleure configuration de modèle.

```
params.yaml
model:
  type: als
  embedding_dim: 64
  regularization: 0.01
  iterations: 15
```

```
optuna:
  enabled: true
  n_trials: 20
  metric: ndcg_at_10 # Cible d'optimisation
```

| Run Name         | embedding_dim | regularization | ndcg_at_10  |
|------------------|---------------|----------------|---|
| als-run-1        | 32            | 0.05           | 0.087    |
| als-run-2        | 64            | 0.05           | 0.091  |
| als-run-3 (best) | 64            | 0.01           | 0.102  |
| als-run-4        | 128           | 0.1            | 0.085  |

# Le cœur du système : le filtrage collaboratif

- **Principe:** Le système analyse les **comportements utilisateurs similaires** plutôt que le contenu des films. Si deux utilisateurs ont aimé les mêmes films, il est probable que leurs autres préférences soient également similaires.
- **Algorithme principal:** **ALS (Alternating Least Squares)**, une technique de factorisation de matrice optimisée pour les données de feedback implicite (ex: vues, clics).



Le système prédit que l'Utilisateur B aimera probablement le film Z.

# Une performance mesurée par des métriques de classement reconnues

Pour évaluer la qualité des recommandations, nous utilisons une série de métriques standards qui mesurent à la fois la pertinence et la qualité du classement des suggestions. Le modèle de popularité sert de baseline de comparaison.

| Métrique           | Description   |
|--------------------|---|
| <b>Precision@K</b> | Proportion d'items pertinents parmi les $K$ premiers recommandés.               |
| <b>Recall@K</b>    | Proportion d'items pertinents retrouvés dans le top $K$ .                       |
| <b>NDCG@K</b>      | Qualité du classement, avec une pénalité pour les items pertinents mal classés. |
| <b>MAP@K</b>       | Précision moyenne, considérant la position de chaque item pertinent.            |
| <b>MRR</b>         | Rang réciproque moyen du premier item pertinent trouvé.                         |

\*\*Note\*: Le jeu de données utilisé est **MovieLens**, contenant ~100 000 évaluations de ~600 utilisateurs sur ~10 000 films.

# Un service de prédiction accessible via une API REST performante

L'API est construite avec FastAPI, garantissant des temps de réponse rapides, une validation automatique des données et une documentation interactive (Swagger UI).

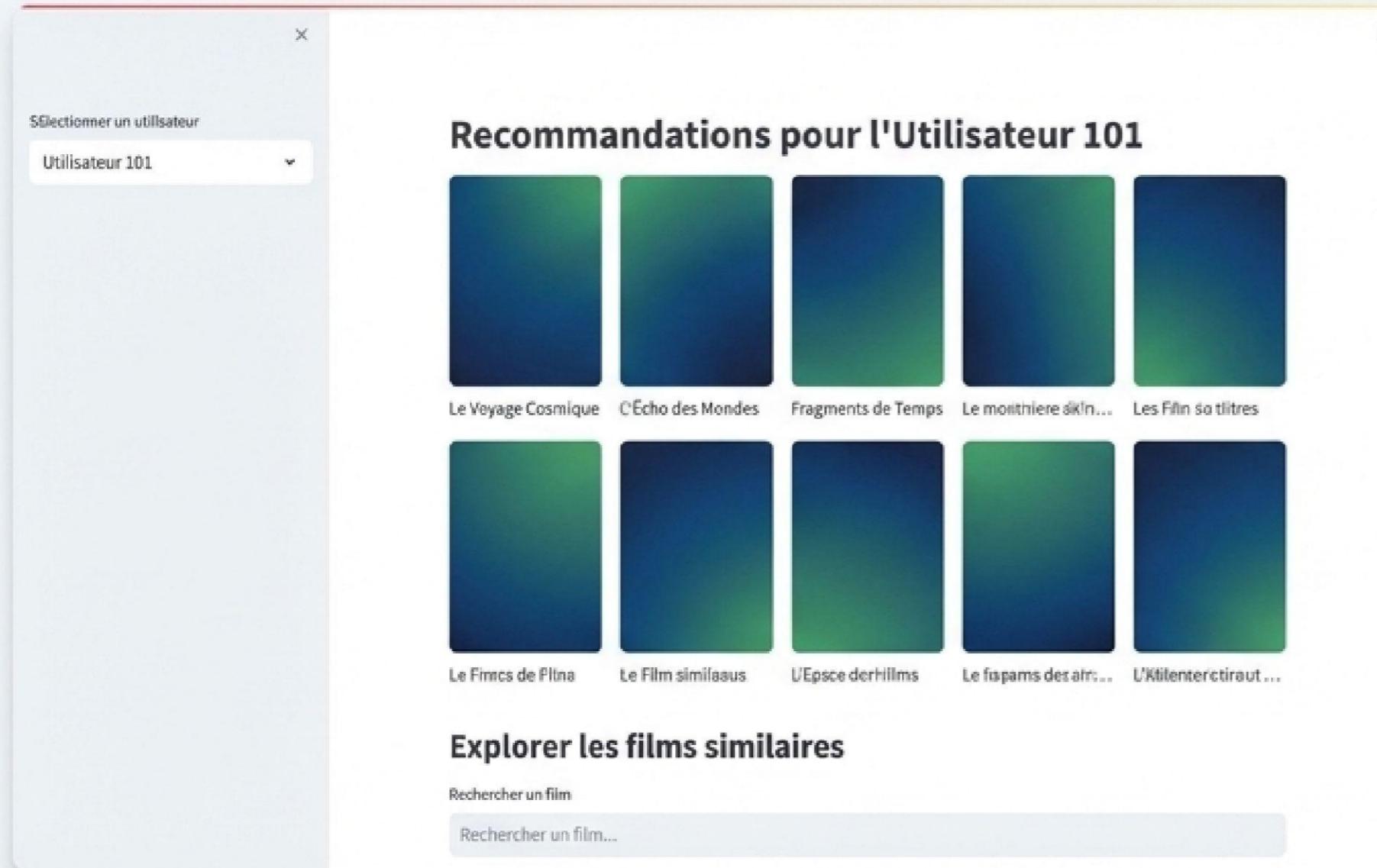
| Méthode | Endpoint       | Description   |
|---------|----------------|---|
| GET     | /health        | Vérification de l'état de santé du service.                     |
| POST    | /recommend     | Obtient des recommandations personnalisées pour un utilisateur. |
| POST    | /similar-items | Trouve des items similaires à un item donné.                    |

## Exemple de Requête

```
# Obtenir 10 recommandations pour l'utilisateur 1
curl -X POST http://localhost:8000/recommend \
-H "Content-Type: application/json" \
-d '{"user_id": 1, "k": 10}'
```

# Une interface interactive pour explorer les recommandations

Une application web construite avec **Streamlit** permet une exploration simple et intuitive des fonctionnalités du système de recommandation.



## Fonctionnalités de l'UI

- ✓ Sélectionner un profil utilisateur dans une liste déroulante.
- ✓ Obtenir et visualiser une liste de recommandations personnalisées.
- ✓ Explorer des films similaires en sélectionnant un titre.
- ✓ Consulter l'historique des interactions de l'utilisateur.

Démo live disponible sur : [mlops-recommender-ui.onrender.com](https://mlops-recommender-ui.onrender.com)

# Un déploiement automatisé et conteneurisé

## Automatisation (CI/CD)

Les workflows **GitHub Actions** sont utilisés pour automatiser les tests, le linting et la construction des images Docker à chaque commit, garantissant la qualité et la stabilité du code.

## Conteneurisation

L'ensemble de l'application (API et UI) est conteneurisé avec **Docker**, assurant un environnement d'exécution cohérent et portable. Le fichier `compose.yaml` orchestre les services localement.

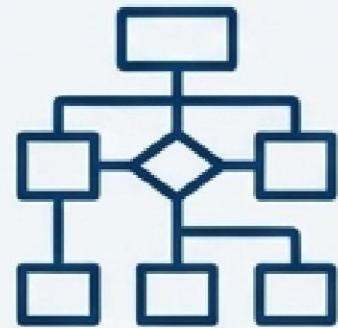
## Déploiement Cloud

Le service est déployé sur **Render** en tant que Web Service, directement depuis le repository GitHub.

Configuration de Déploiement Clé (sur Render)

```
Build Command: pip install poetry  
&& poetry install  
Start Command: poetry run uvicorn  
src.serving.api:app --host 0.0.0.0  
--port $PORT
```

# Plus qu'un projet : un blueprint pour des systèmes ML de production



## Architecture Robuste

Une conception modulaire et découpée qui sépare la logique ML, le service API et l'interface utilisateur.



## Reproductibilité Totale

Pipelines versionnés avec DVC et gestion des dépendances avec Poetry pour garantir des résultats cohérents.



## Expérimentation Systématique

Suivi rigoureux avec MLflow et optimisation automatique avec Optuna pour une amélioration continue.



## Déploiement Automatisé

Intégration continue (CI/CD) et conteneurisation pour des déploiements fiables et rapides.

*Ce projet sert de guide pratique pour implémenter les meilleures pratiques MLOps dans un cas d'usage concret.*