Projet API REST – Catalogue de Films & Commentaires

🎯Objectif pédagogique

L'objectif de ce projet est de concevoir et développer une **API REST complète** à partir d'un **jeu de données CSV réel**. Ce projet vise à évaluer votre capacité à :

- concevoir une API REST cohérente,
- manipuler et importer des données depuis un fichier CSV,
- implémenter des filtres de recherche,
- gérer des relations entre ressources (films / commentaires),
- documenter proprement une API,
- containeriser une application avec Docker.

Le langage, le framework et la base de données sont **libres**.

Jeu de données fourni

Vous disposez du fichier suivant (voir https://github.com/MadzMed/serviceWeb/blob/3945dbc60b395bd31904641c0a930e6c826f6774/tmdb_movies_dataset.csv) :

tmdb_movies_dataset.csv

Colonnes du CSV

Colonne	Description	Utilisation dans l'API
`movie_id`	Identifiant du film (TMDB)	`id` du film
`title`	Titre du film	`title`
`release_date`	Date de sortie (YYYY-MM-DD)	`release_date`
`vote_count`	Nombre de votes	optionnel
`popularity`	Indice de popularité	optionnel
`rating`	Note moyenne du film	`rating`

Vous devez **importer ce CSV** afin d'initialiser votre base de données.

Modélisation attendue

🎬Film

- `id` (int ou UUID)

- `title` (string, obligatoire)
- `release_date` (date, obligatoire)
- `rating` (float, obligatoire)
- `popularity` (float, optionnel)
- `vote_count` (int, optionnel)
- `created_at` (timestamp)
- `updated_at` (timestamp)

🎂 Commentaire

- `id`
- `film_id` (clé étrangère)
- `author` (string ou utilisateur authentifié)
- `content` (string, obligatoire)
- `created_at`
- `updated_at`

Fonctionnalités obligatoires

Lister les films avec filtres

Endpoint : `GET /films`

Paramètres de requête

- `title` : recherche partielle (insensible à la casse)
- `release_date` : date exacte (YYYY-MM-DD)
- `release_from` : date minimale
- `release_to` : date maximale
- `min_rating` : note minimale
- `max_rating` : note maximale
- `page` : pagination (défaut = 1)
- `page_size` : taille de page (défaut = 20)
- `sort` : tri (`rating`, `-rating`, `release_date`, etc.)

Consulter un film

Endpoint : `GET /films/{filmId}`

- Retourne les informations détaillées d'un film
- Retourne `404` si le film n'existe pas

Créer un film

Endpoint : `POST /films`

Body JSON :

```
{
  "title": "Mon film",
  "release_date": "2022-01-15",
  "rating": 7.5,
  "popularity": 12.3,
  "vote_count": 340
}
```

Validations attendues :

- `title` non vide
- `rating` compris entre 0 et 10
- `release_date` valide

🎂 Gestion des commentaires

Lister les commentaires d'un film

Endpoint : `GET /films/{filmId}/comments`

Ajouter un commentaire

Endpoint : `POST /films/{filmId}/comments`

```
{
  "author": "alice",
  "content": "Excellent film"
}
```

Modifier un commentaire

Endpoint : `PATCH /comments/{commentId}`

```
{
  "content": "Avis modifié"
}
```

Supprimer un commentaire

Endpoint : `DELETE /comments/{commentId}`

🎋 Bonus – Sécurité (optionnel mais valorisé)

Vous pouvez implémenter une authentification (JWT, API key, etc.) :

- Un utilisateur authentifié peut créer des films/commentaires
- **Seul le créateur** peut modifier ou supprimer ses ressources
- Les autres utilisateurs reçoivent une erreur `403 Forbidden`

🪇 Docker (obligatoire)

Exigences

- Un `Dockerfile` fonctionnel
- L'API doit être lançable via :

```
docker build -t movies-api .
docker run -p 8000:8000 movies-api
```

Recommandé

- `docker-compose.yml`
- Base de données séparée (PostgreSQL, MySQL, etc.)
- Variables d'environnement

README.md (obligatoire)

Votre README doit contenir :

1. Présentation du projet
2. Instructions de lancement (Docker)
3. Méthode d'import du CSV
4. Documentation complète des endpoints
5. Exemples `curl`
6. Choix techniques
7. (Bonus) Sécurité & authentification

🧨 Tests (recommandés)

- Tests d'intégration sur les filtres
- Gestion des erreurs (400, 404, 403)

• Collection Postman / Insomnia appréciée

📷Barème indicatif (/20)

Critère	Points
Filtres & listing des films	6
CRUD commentaires	5
Import CSV	1
Docker fonctionnel	3
README clair et complet	3

Bonus sécurité : +2 points

Livrables attendus

• Code source du projet
• `Dockerfile` (+ `docker-compose.yml` si utilisé)
• `README.md`
• (Optionnel) Tests / documentation OpenAPI

📌**Tout comportement non documenté ou non justifié sera pénalisé.**

Bon courage 🗜️