TP : Créer une API hybride SQL + NoSQL

Nathan Deroeck — MixtAPI — Efrei 2025

1. Introduction

L'objectif de ce TP est de concevoir et déployer une API REST hybride capable d'exploiter à la fois une base de données relationnelle (PostgreSQL) et une base NoSQL (MongoDB). L'application, développée en Node.js avec Express, permet de gérer des utilisateurs, des livres et des profils utilisateurs enrichis stockés dans MongoDB. Cette approche illustre la complémentarité des modèles SQL et NoSQL au sein d'une même architecture.

2. Architecture de l'application

Le projet repose sur une architecture MVC (Model – View – Controller) :

- config/ : connexions à PostgreSQL et MongoDB
- models/ : schémas et fonctions d'accès aux données (SQL et NoSQL)
- controllers/ : logique métier et traitement des requêtes
- routes/ : définition des endpoints REST

PostgreSQL gère les entités structurées (users, books), tandis que MongoDB stocke les profils utilisateurs plus dynamiques, contenant des préférences et un historique de lecture.

3. Principales routes et extraits de code

Routes implémentées :

- GET /api/users : liste des utilisateurs (PostgreSQL)
- POST /api/users : ajout d'un utilisateur SQL
- GET /api/books : liste des livres (PostgreSQL)
- POST /api/profiles : création d'un profil MongoDB
- PUT /api/profiles/:userId : mise à jour (ajout historique)

Exemple d'insertion d'un utilisateur (PostgreSQL) :

```
curl -X POST http://localhost:4000/api/users \
  -H 'Content-Type: application/json' \
  -d '{"name":"Alice","email":"alice@mail.com"}'
```

Exemple de création de profil (MongoDB) :

```
curl -X POST http://localhost:4000/api/profiles \
  -H 'Content-Type: application/json' \
  -d '{"userId":1,"preferences":["Fantasy"],"history":[{"book":"Harry Potter","rating":5}]}'
```

Lecture mixte (SQL + NoSQL) :

```
curl http://localhost:4000/api/users/full/1
```

4. Partie 7 — Tests et validations

Les tests ont été réalisés avec Postman et curl pour valider :
1. L'insertion d'utilisateurs et de livres dans PostgreSQL.
2. La création de profils dans MongoDB.
3. La lecture d'un utilisateur complet via /api/users/full/:id.
4. La mise à jour d'un profil avec ajout d'historique.

Extraits de résultats :

POST /api/users → 201 Created
{ "id": 1, "name": "Alice", "email": "alice@mail.com" }

POST /api/profiles → 201 Created
{ "userId": 1, "preferences": ["Fantasy"], "history": [...] }

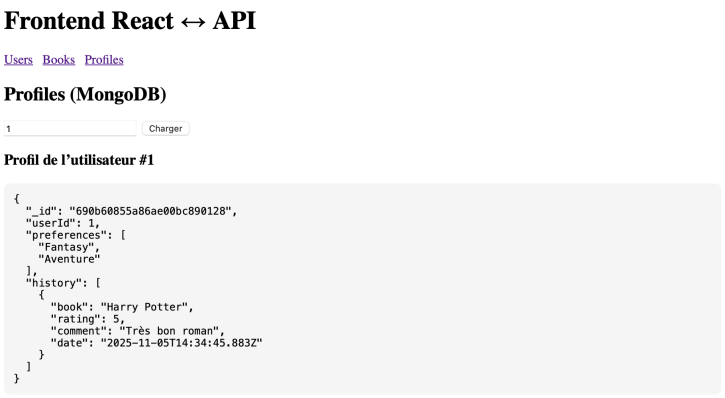
GET /api/users/full/1 → 200 OK
{ "user": {...}, "profile": {...} }

5. Résultat après lancement du front et du back

Liste des Users avec la possibilité d'ajouter, modifier et supprimer un user

Liste Books avec toujours la possibilité d'ajouter un livre et de mettre sa disponibilité ou pas.

Profiles : qui va chercher un profil avec un ID dans la BDD mangoDB pour l'afficher et pouvoir la modifier



6. Conclusion

Ce TP a permis de concevoir une API Node/Express connectée à PostgreSQL et MongoDB. Le modèle hybride illustre la complémentarité entre la rigueur du SQL et la flexibilité du NoSQL. Les tests Postman ont confirmé la stabilité et la cohérence de l'application. Ce type d'architecture est pertinent pour des systèmes combinant données structurées et contenus évolutifs.

PS: Je viens d'avoir mon Mac j'ai mis du temps a faire les installation et comprendre donc désolé pour le travail un peux maché. :(