Documentation

1. Le projet en question

- **Bookly+** : une API qui gère
  - les **users** et les **books** en SQL (PostgreSQL)
  - les **profils** (préfs, historique) en NoSQL (MongoDB)
- On combine le tout pour renvoyer un **user + son profil** d'un seul coup.

2. Module SQL (Users)

1. **User.model.js**
   a. Méthodes statiques :

• findAll(), findById(id), create(name, email).

   b. Exécute des requêtes paramétrées avec pool.query.
2. **userController.js**
   a. Vérifie que name et email sont là.
   b. Contrôle l'unicité de l'email.
   c. Appelle User.create(...).
   d. Envoie le JSON au client.
3. **userRoutes.js**
   a. GET /
   b. GET /:id
   c. POST /
   d. PUT /:id
   e. DELETE /:id

Pourquoi SQL ? transactions ACID, contraintes, JOIN faciles, schéma strict.

3. Module NoSQL (Profiles)

1. **Profile.model.js**
   a. Schéma Mongoose :

• userId (Number)

• preferences (genres, auteurs)

• history (tableau d'objets)

    b. Middleware pour mettre à jour updatedAt.
2. **profileController.js**
    a. createProfile :
        i. Vérifie que l'utilisateur existe en SQL.
        ii. Crée un doc Mongo si OK.
    b. addHistoryEntry :
        i. $push dans le tableau history.
        ii. $set pour updatedAt.
3. **profileRoutes.js**
    a. GET /:userId
    b. POST /
    c. PUT /:userId
    d. POST /:userId/history

Pourquoi NoSQL ? schéma flexible, docs imbriqués, pas de migration, sharding.

4. Route Hybride (getUserFull)

- **Endpoint** : GET /api/user-full/:id
- **Logique** :
    - Récupère user en SQL.
    - Récupère profile en NoSQL.
    - Combine dans { user: {...}, profile: {...} }.

Option simple :

javascript

```
const [user, profile] = await Promise.all([
  User.findById(id),
  Profile.findOne({ userId: +id })
]);
```

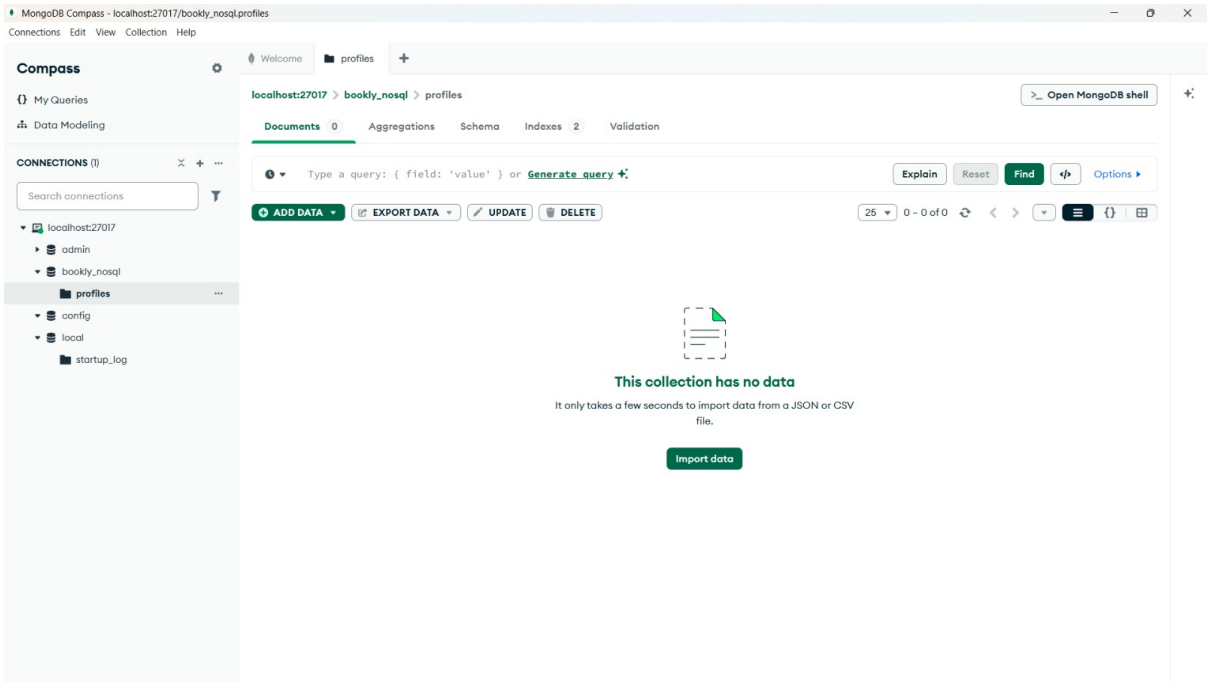
Attention

- Pas de transaction cross-DB (pas de rollback global).
- Deux requêtes, un peu plus de latence (on peut paralléliser).

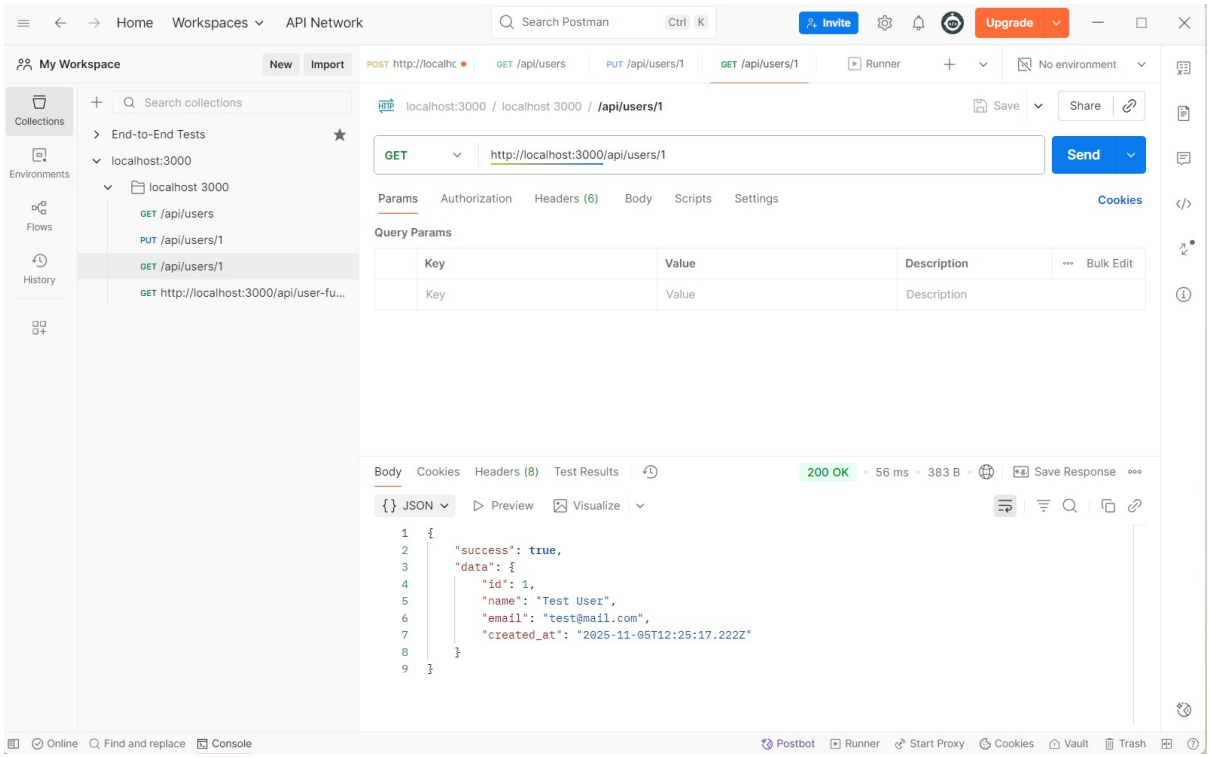
5. Pourquoi c’est cool ?

- On utilise le meilleur de chaque monde :
  - **SQL** pour les données structurées et critiques
  - **NoSQL** pour la flexibilité et les gros volumes
- Réponse **tout-en-un** pour le frontend.
- Architecture claire, chaque module est indépendant.

Capture mongoDB



Capture postman



Schéma