Compte rendu

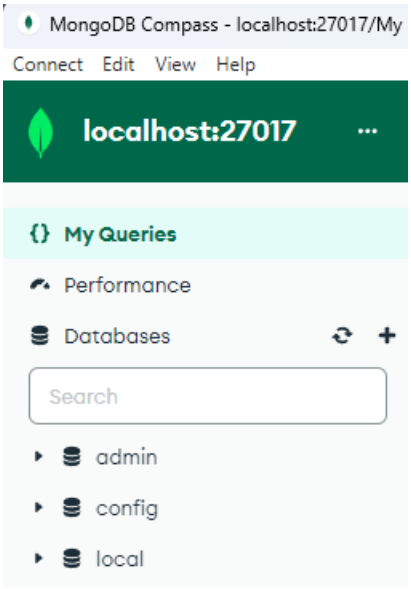
Développement Avancé : TP 5

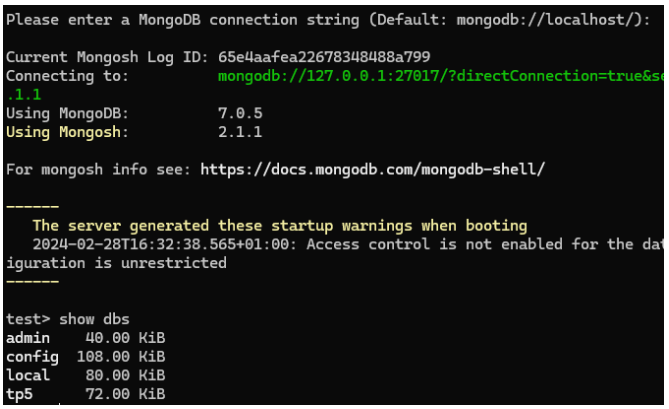
Laurent Giustignano

Par Steven TEA 303

Étape 1

Pour commencer, j'ai installé MongoDB et notamment MongoDB Compass pour pouvoir naviguer facilement dans notre base de données NoSQL avec une interface graphique. Mais sinon, avec Mongosh, nous pouvons voir les bases de données en utilisant la commande show dbs.





Étape 2

Pour ce qui est de la connexion à la base de données, comme l'URI et le nom de la base de données, j'ai créé un fichier d'environnement pour les stocker plutôt que de les enregistrer en dur dans le programme. J'ai ensuite commencé à tester l'insertion de données dans la base de données NoSQL et à mettre en place l'API REST avec Mangoose et Fastify.
Comme indiqué, j'ai créé un fichier à part, uniquement utilisé pour la connexion à la base de données.

Une fois la base de données en place, je m'occupe de la sécurité de l'API. Comme pour le TP précédent, j'ai créé une demande de signature de certificat, clé publique et privée. Puis, pour pouvoir faire des requêtes HTTPS, j'ai auto signé le certificat.

Étape 3

Contrôleur

Pour les routes de l'API et garantir les principes REST, les routes sont simplement le nom de la collection que nous allons manipuler, donc des albums. Les routes de l'API partagent la même URI : https://localhost:5000/album/, mais avec des méthodes différentes. Un album est affecté à un identifié noté _id, cela va nous permettre de sélectionner l'album que nous voulons manipuler. L'identifiant est saisi à partir de l'URI et le contenu de l'album est à envoyer dans le corps de la requête.

Donc nous avons :

- https://localhost:5000/album/ en méthode POST : ajoute un nouvel album dans la base de données. Dans le corps de la requête, il faut renseigner toutes les informations obligatoires d'un album,

- https://localhost:5000/album/ en méthode GET : récupère l'ensemble des albums dans la base de données
- https://localhost:5000/album/:id en méthode GET : récupère l'album avec l'identifiant saisi

- https://localhost:5000/album/:id en méthode PUT : modifie un album, si un album a été trouvé avec son _id, les informations modifiées doivent toujours être dans le format du json attendu.

- https://localhost:5000/album/:id en méthode DELETE : supprime l'album de l'identifiant saisie.

Dans le cas où une erreur a été levée, par exemple, lors de la manipulation de la base de données ou que l'identifiant de l'album n'a pas été saisi ou qu'il soit invalide, le contrôleur envoie un message d'erreur personnalisé en fonction de l'erreur et le code d'erreur 400.

J'ai également mis en place les Json Schema pour imposer un certain format du corps des requêtes et de réponses.

```
export const albumSchemaRequest = {
    type: 'object',
    properties: {
        titre: { type: 'string' },
        description: { type: 'string' },
        auteur: { type: 'string' },
        format: {
            type: 'string',
            enum: ['poche', 'manga', 'audio']
        }
    },
    required: ['titre', 'description', 'auteur'],
    additionalProperties: false
}
```

```
const albumSchemaResponse = {
    type: 'object',
    properties: {
        _id: {type: 'string'},
        titre: {type: 'string'},
        description: {type: 'string'},
        auteur: {type: 'string'},
        format: {type: 'string', enum: ['poche', 'manga', 'audio']}
    },
    required: ['_id', 'titre', 'description', 'auteur'],
    additionalProperties: false
};

export const getAlbums = {
    schema: {
        response: {
            200: {
                type: 'array',
                items: {
                    type: 'object',
                    properties: {
                        ...albumSchemaResponse.properties,
                        __v: { type: 'number', exclude: true }
                    },
                    required: albumSchemaResponse.required,
                    additionalProperties: albumSchemaResponse.additionalProperties
                }
            }
        }
    }
};
```