

MINI PROJECT NoSQL (MongoDB)

1. Intorduction

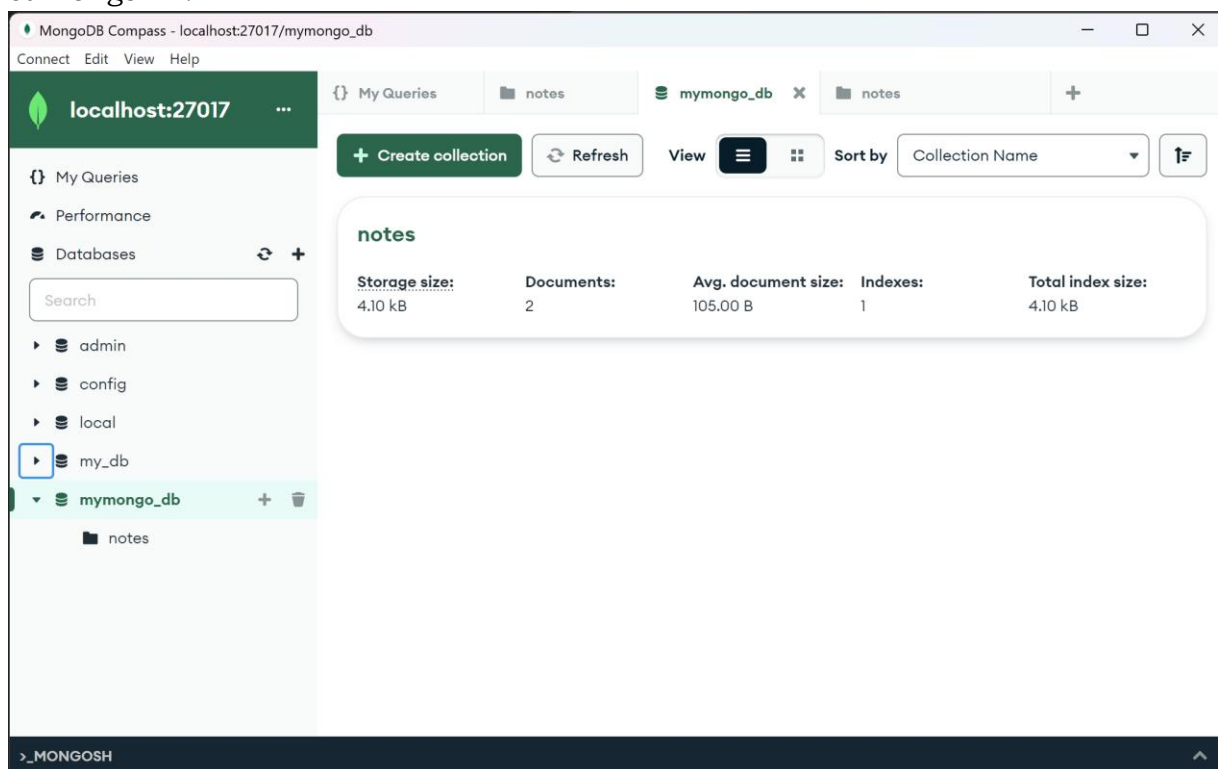
L'objectif principal du projet était de créer une application utilisant **Node.js** et **Express.js** pour fournir des **API CRUD** (**C**reate, **R**ead, **U**ppdate, **D**eleter) pour gérer des notes avec un titre et un corps, en utilisant une base de données **MongoDB** pour le stockage.

2. Configuration de l'environnement de développement :

- **Node.js** et **MongoDB** ont été installés sur le système de développement.
- Le projet **Node.js** a été initialisé en utilisant **npm**.
- **Express.js** a été configuré en tant que framework web pour l'application.

3. Mise en place de la base de données MongoDB :

- Une base de données MongoDB a été créée.
- Une collection "notes" a été mise en place pour stocker les notes.
- Mongoose a été utilisé pour établir une connexion entre l'application Node.js et MongoDB.



NB : Une Foix j'ai Définir le nom de la base de donnée dans le fichier de configuration du projet. La collection notes à été crée automatiquement

4. Création des API CRUD :

✓ Create (POST) :

- Une route POST a été mise en place pour permettre la création de nouvelles notes.

```
//La route
noteRoutes.post('/notes/add', addNote); // La route

//le controller
const addNote = (req, res) => {
  const {title, body} = req.body

  const newNote = new Note({ title, body });

  newNote.save().then(() => {
    res.json("note added");
  }).catch(err => console.log(err));
}
```

- Les données entrantes ont été validées avant d'être enregistrées dans la base de données.

✓ Read (GET) :

- Une route GET a été implémentée pour récupérer toutes les notes ou une note spécifique.

```
//la route
noteRoutes.get('/notes', getAllNotes);
noteRoutes.get('/notes/:id', noteById);

//Le controller
const getAllNotes = async (req, res) => {
  try {
    const notes = await Note.find().select('title body createdAt');
    res.json({
      notesData : notes
    });
  } catch (error) {
    res.status(500).json({ message: err.message });
  }
}

const noteById = (req, res) => {
  Note.findById(req.params.id).then(note => {
    res.json({ note });
  }).catch(err => console.log(err));
}
```

✓ **Update (PUT) :**

- Une route PUT a été créée pour mettre à jour une note existante.
- Les données entrantes ont été validées avant la mise à jour de la note dans la base de données.

```
La Route
noteRoutes.put('/notes/:id', updateNote);

const updateNote = (req, res) => {
  Note.findById(req.params.id).then(note => {
    const {title, body} = req.body;
    note.title = title;
    note.body = body;
    note.save().then(() => {
      res.json("note updated");
    }).catch(err => console.log(err));
  }).catch(err => console.log(err));
}
```

✓ **Delete (DELETE) :**

- Une route DELETE a été établie pour permettre la suppression d'une note.

```
noteRoutes.delete('/notes/:id', deleteNote);

const deleteNote = (req, res) => {
  Note.findByIdAndDelete(req.params.id).then(note => {
    res.json("note deleted");
  }).catch(err => console.log(err));
}
```

- La note a été supprimée de la base de données.

5. Tests :

Des tests unitaires et/ou d'intégration ont été développés pour garantir le bon fonctionnement des API CRUD.

L'outil **Postman** a été utilisé pour tester manuellement les différentes fonctionnalités de l'API.

6. Conclusion :

- Le projet a été réalisé avec succès, fournissant des **API CRUD** fonctionnelles pour gérer des notes dans une base de données **MongoDB**.

7. Annexes :

- Le code source de l'application, y compris le fichier de configuration, les routes et le contrôleur
- Des captures d'écran de l'application en cours d'exécution et des tests effectués avec **Postman**. et vérifier le contenu avec le **mongoDb Compass**

Create

The image displays two screenshots related to a REST API test and database verification.

Top Screenshot (Postman): Shows a POST request to `http://localhost:4200/notes/add` with a JSON body: `{ "title": "Title 1", "body": "Body 1" }`. The response status is `200 OK` with a response time of `7 ms` and a body of `"note added"`.

Bottom Screenshot (MongoDB Compass): Shows the `mymongo_db` database with the `notes` collection. The document stored is:

```
{
  "_id": ObjectId('66063747bcde0099b996dfd6'),
  "title": "Title 1",
  "body": "Body 1",
  "createdAt": 2024-03-29T03:36:39.673+00:00,
  "updatedAt": 2024-03-29T03:36:39.673+00:00,
  "__v": 0
}
```

Read

- Récupérer tous les notes

The screenshot shows a REST client interface for a MongoDB application. The request is a GET to `http://localhost:4200/notes/`. The response is a 200 OK status with a response time of 15 ms and a body size of 499 B. The response body is a JSON array containing two note objects.

```
1 {
2   "notesData": [
3     {
4       "_id": "66063747bcde0099b996dfd6",
5       "title": "Title 1",
6       "body": "Body 1",
7       "createdAt": "2024-03-29T03:36:39.673Z"
8     },
9     {
10      "_id": "6608863e3da8adda146fa862",
11      "title": "title 2",
12      "body": "body 2",
13      "createdAt": "2024-03-30T21:38:06.258Z"
14    }
15  ]
16 }
```

- Récupérer une note par l'identifiant ID

The screenshot shows a REST client interface for a MongoDB application. The request is a GET to `http://localhost:4200/notes/66063747bcde0099b996dfd6`. The response is a 200 OK status with a response time of 14 ms and a body size of 431 B. The response body is a JSON object representing a single note.

```
1 {
2   "note": {
3     "_id": "66063747bcde0099b996dfd6",
4     "title": "Title 1",
5     "body": "Body 1",
6     "createdAt": "2024-03-29T03:36:39.673Z",
7     "updatedAt": "2024-03-29T03:36:39.673Z",
8     "__v": 0
9   }
10 }
```

Update

The screenshot shows a REST client interface for a project named 'MongoDb_Notes'. The active tab is 'Update', and the HTTP method is set to 'PUT'. The URL is 'http://localhost:4200/notes/66063747bcde0099b996dfd6'. The 'Body' tab is selected, and the request body is in JSON format, containing a document with 'title' and 'body' fields, both updated to '101'. The status bar at the bottom indicates a successful '200 OK' response with a response time of 34 ms and a body size of 280 B. Below the status bar, the response body is displayed in a 'Pretty' JSON view, showing the updated note document.

```
PUT http://localhost:4200/notes/66063747bcde0099b996dfd6
```

Params Auth Headers (8) **Body** Pre-req. Tests Settings Cookies

raw JSON Beautify

```
1 {
2   "title": "title 101",
3   "body": "body 101"
4 }
```

Body 200 OK 34 ms 280 B Save as example

Pretty Raw Preview Visualize JSON

```
{
  "note": {
    "_id": "66063747bcde0099b996dfd6",
    "title": "title 101",
    "body": "body 101",
    "createdAt": "2024-03-29T03:36:39.673Z",
    "updatedAt": "2024-03-30T21:45:50.692Z",
    "__v": 0
  }
}
```

- Récupérer une note par l'identifiant ID

The screenshot shows the same REST client interface, but the active tab is 'ReadById' and the HTTP method is set to 'GET'. The URL remains the same. The status bar indicates a successful '200 OK' response with a response time of 9 ms and a body size of 435 B. The response body is displayed in a 'Pretty' JSON view, showing the full note document with its metadata.

```
GET http://localhost:4200/notes/66063747bcde0099b996dfd6
```

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Key	Value	Description
Key	Value	Description

Body 200 OK 9 ms 435 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "note": {
3     "_id": "66063747bcde0099b996dfd6",
4     "title": "title 101",
5     "body": "body 101",
6     "createdAt": "2024-03-29T03:36:39.673Z",
7     "updatedAt": "2024-03-30T21:45:50.692Z",
8     "__v": 0
9   }
10 }
```

Delete

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** `http://localhost:4200/notes/66063747bcde0099b996dfd6`
- Response Status:** 200 OK, 15 ms, 280 B
- Response Body (Pretty):**

```
1 "note deleted"
```

- Récupérer une note par l'identifiant ID

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:4200/notes/66063747bcde0099b996dfd6`
- Response Status:** 200 OK, 14 ms, 279 B
- Response Body (Pretty):**

```
1 {  
2   "note": null  
3 }
```

- Récupérer une note par l'identifiant ID

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:4200/notes/`
- Response Status:** 200 OK, 10 ms, 391 B
- Response Body (Pretty):**

```
1 {  
2   "notesData": [  
3     {  
4       "_id": "6608863e3da8adda146fa862",  
5       "title": "title 2",  
6       "body": "body 2",  
7       "createdAt": "2024-03-30T21:38:06.258Z"  
8     }  
9   ]  
10 }
```