

Rapport TP5

Louis Prigneaux 304

Etape 1 :

- Installation de MongoDB et Test de "show dbs" :

```
test> show dbs
admin      40.00 KiB
config     48.00 KiB
local      72.00 KiB
test       52.15 MiB
```

Etape 2 :

- Etablissement des certificat https :
openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out cert.pem
- Mise en place de la connexion avec mongoose et des schémas mongoDB :

```
import mongoose from 'mongoose';

const bookSchema = new mongoose.Schema({
  title: { type: String, required: true },
  author: { type: String, required: true },
  description: String,
  format: { type: String, enum: ['poche', 'manga', 'audio'], default: 'poche' }
});

const Book = mongoose.model('Book', bookSchema);
export default Book;
```

```
{"level":30,"time":1739892673698,"pid":80562,"hostname":"MacBook-Pro-de-Louis.local","msg":"Server listening at https://127.0.0.1:3000"}
Serveur en écoute sur https://localhost:3000
{"level":30,"time":1739892673698,"pid":80562,"hostname":"MacBook-Pro-de-Louis.local","msg":"Server listening at https://172.20.10.9:3000"}
MongoDB connecté
```

Etape 3 :

- Mise en place de toutes les routes :

```
const bookRoutes = async (fastify, opts) => {
  fastify.post('/books', { schema: bookSchema }, createBook);
  fastify.get('/books', getBooks);
  fastify.get('/books/:id', getBookById);
  fastify.put('/books/:id', updateBook);
  fastify.delete('/books/:id', deleteBook);
};
```

- Utilisation de ces routes dans le controllers :

```
export const createBook = async (request, reply) => {
  try {
    const book = new Book(request.body);
    await book.save();
    reply.code(201).send({ titre: book.titre, auteur: book.auteur,
description: book.description, format: book.format });
  } catch (error) {
    reply.code(500).send({ message: error.message });
  }
};
```

[...]

- Schema Json :

```
export const bookSchema = {
  body: {
    type: 'object',
    required: ['titre', 'auteur'],
    properties: {
      titre: { type: 'string' },
      auteur: { type: 'string' },
      description: { type: 'string' },
      format: { type: 'string', enum: ['poche', 'manga', 'audio'], default:
'poche' }
    }
  }
};
```

- Tests :

Create Valide :

```
{
  "titre": "Le Petit Prince",
  "auteur": "Antoine de Saint-Exupéry",
  "description": "Un conte philosophique et poétique.",
  "format": "poche"
}
```

Cookies Headers (6) Test Results | ↻ 201 Created

JSON ▾ ▶ Preview 🔗 Visualize ▾

```
{
  "titre": "Le Petit Prince",
  "auteur": "Antoine de Saint-Exupéry",
  "description": "Un conte philosophique et poétique.",
  "format": "poche"
}
```

Create Non Valide :

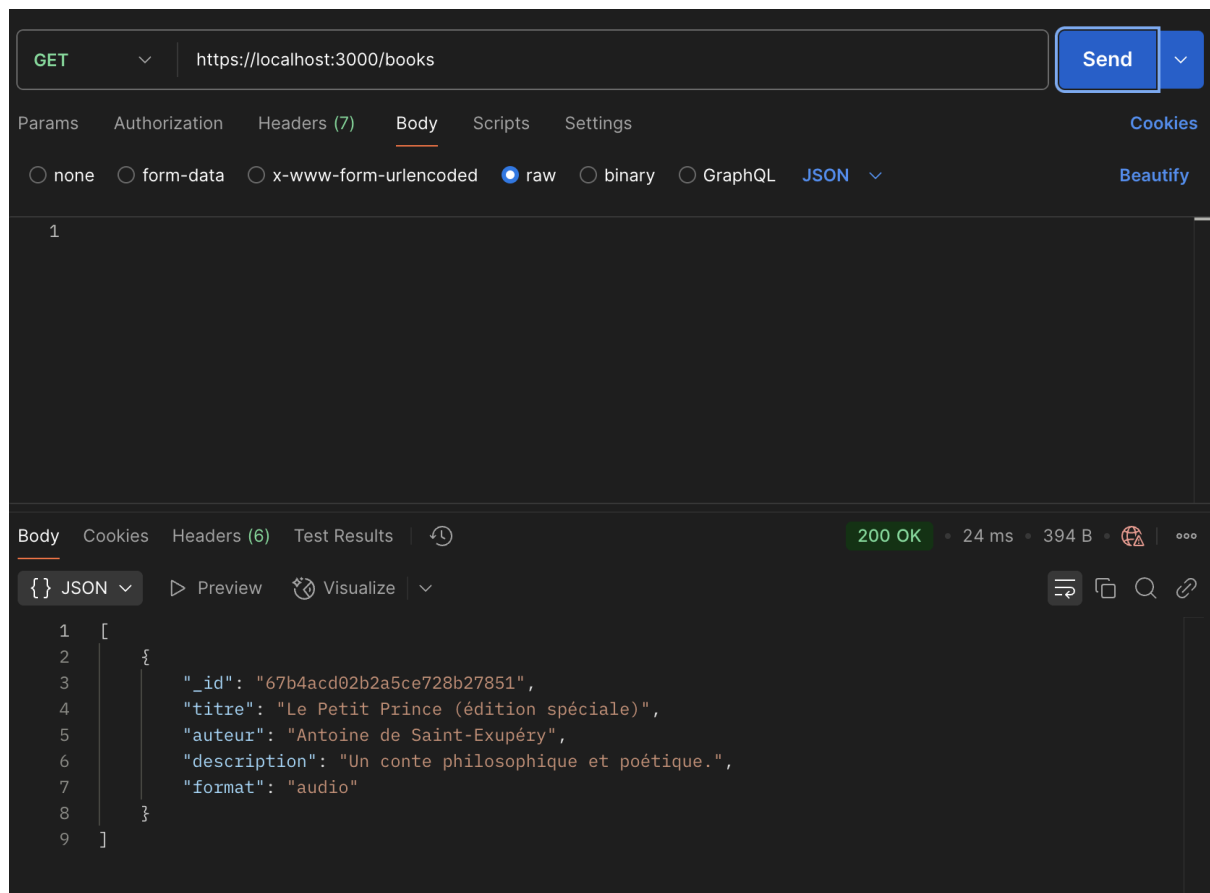
```
1 {
2   "titre": "Le Petit Prince",
3   "description": "Un conte philosophique et poétique.",
4   "format": "poche"
5 }
6
```

y Cookies Headers (6) Test Results | ↻ 400 Bad Request

JSON ▾ ▶ Preview 🔗 Visualize ▾

```
1 {
2   "statusCode": 400,
3   "code": "FST_ERR_VALIDATION",
4   "error": "Bad Request",
5   "message": "body must have required property 'auteur'"
6 }
```

Get Books :



GET https://localhost:3000/books Send

Params Authorization Headers (7) Body Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

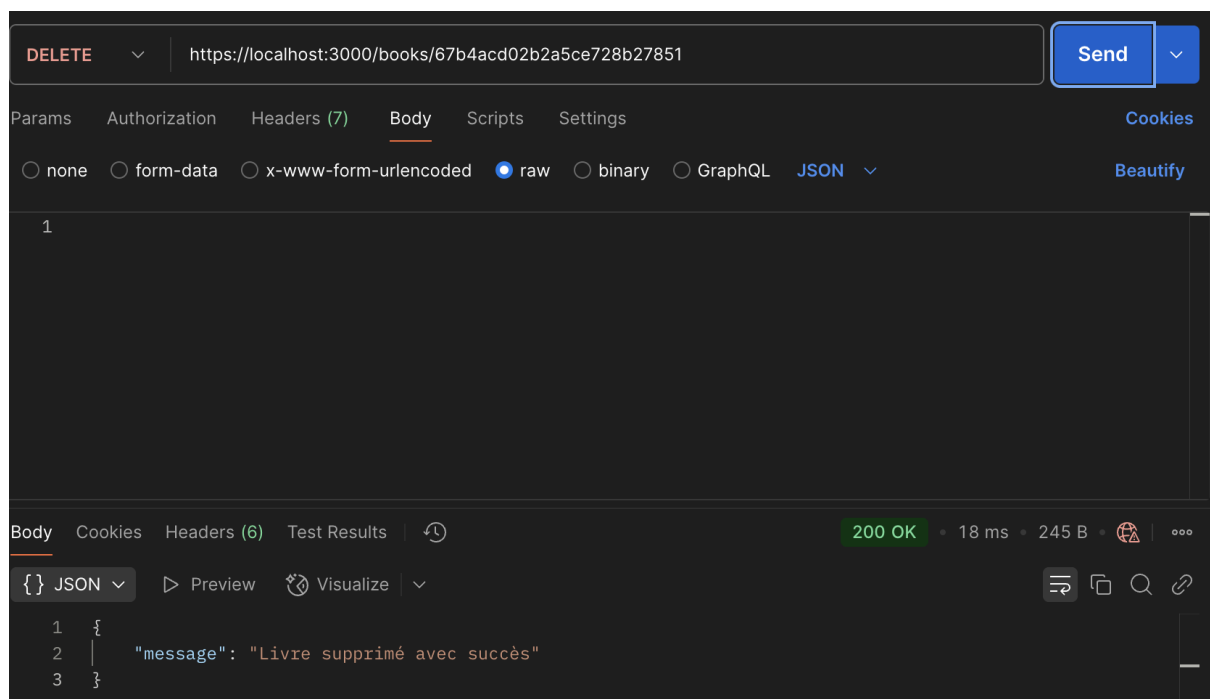
1

Body Cookies Headers (6) Test Results 200 OK • 24 ms • 394 B • 🌐 ⋮

{} JSON Preview Visualize ⌵ 🔍 🔗

```
1  [
2    {
3      "_id": "67b4acd02b2a5ce728b27851",
4      "titre": "Le Petit Prince (édition spéciale)",
5      "auteur": "Antoine de Saint-Exupéry",
6      "description": "Un conte philosophique et poétique.",
7      "format": "audio"
8    }
9  ]
```

Delete :



DELETE https://localhost:3000/books/67b4acd02b2a5ce728b27851 Send

Params Authorization Headers (7) Body Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

1

Body Cookies Headers (6) Test Results 200 OK • 18 ms • 245 B • 🌐 ⋮

{} JSON Preview Visualize ⌵ 🔍 🔗

```
1  {
2    "message": "Livre supprimé avec succès"
3  }
```