



SUJET D'EXAMEN : GESTION SIMPLIFIÉE D'UNE BIBLIOTHÈQUE

ENCADRÉ PAR :

❖ **MR SENY MBAYE**

PRÉSENTÉ PAR :

**SOKHNA DIEYE
BACARY SANO**

INTRODUCTION:

Dans le cadre de l'apprentissage des langages et frameworks backend, ce projet a pour objectif la conception d'une API REST permettant de gérer une bibliothèque de manière simplifiée. Il s'inscrit dans le cadre de l'apprentissage des concepts backend avec Spring Boot, notamment la manipulation des entités JPA, la création d'API REST, et la structuration d'un projet selon les bonnes pratiques du développement Java.

1: Objectifs pédagogiques atteints :

- ❖ Manipulation des entités JPA avec Hibernate
- ❖ Compréhension et implémentation des relations **@OneToMany** et **@ManyToOne**
- ❖ Création de endpoints REST (GET, POST, PUT, DELETE)
- ❖ Structuration du projet Spring Boot (controllers, services, repositories, models)

2: Technologies Utilisés :

- Java 17+
- Spring Boot 3+
- Spring Data JPA
- H2 Database
- Spring Web
- Maven
- Lombok (optionnel)
 - Postman pour les tests API

3: Structure attendue du projet :

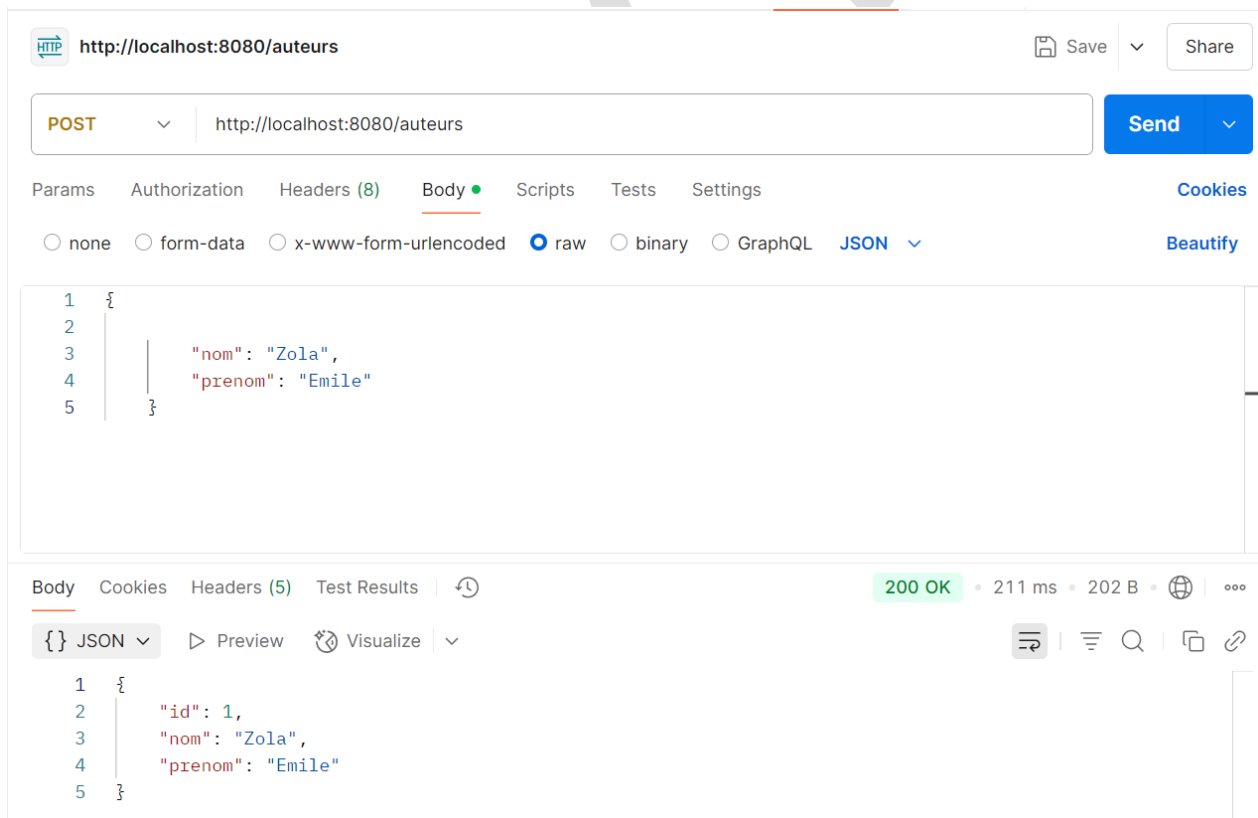
```
src/  
└─ main/  
    └─ java/com/monappli/  
        └─ controller/  
            └─ service/  
            └─ repository/  
            └─ model/
```


| └─ MonappliApplication.java
└─ resources/
└─ application.properties


✓ 4: Fonctionnalités principales :



◆ Créer un auteur :


Nous avons créé 5 auteurs sur Postman dont Emile Zola, Leopold Sedar Senghor, Mariama Ba, Victor Hugo et Albert Camus.




 **http://localhost:8080/auteurs**



Save  Share









POST  http://localhost:8080/auteurs Send 

Params Authorization Headers (8) **Body**  Scripts Tests Settings Cookies


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


```
1 {
2   |
3   |   "nom": "Senghor",
4   |   "prenom": "Leopold Sedar"
5   | }
```



Body Cookies Headers (5) Test Results  **200 OK** • 11 ms • 213 B •  ...


{} JSON  Preview  Visualize      


```
1 {
2   |   "id": 5,
3   |   "nom": "Senghor",
4   |   "prenom": "Leopold Sedar"
5   | }
```

 **http://localhost:8080/auteurs**



Save  Share









POST  http://localhost:8080/auteurs Send 

Params Authorization Headers (8) **Body**  Scripts Tests Settings Cookies

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

```
1 {
2   |
3   |   "nom": "Ba",
4   |   "prenom": "Mariama"
5   | }
```

Body Cookies Headers (5) Test Results  **200 OK** • 11 ms • 202 B •  ...

{} JSON  Preview  Visualize      

```
1 {
2   |   "id": 4,
3   |   "nom": "Ba",
4   |   "prenom": "Mariama"
5   | }
```

HTTP <http://localhost:8080/auteurs> Save Share

POST <http://localhost:8080/auteurs> Send

Params Authorization Headers (8) **Body** Scripts Tests Settings Cookies

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

```
1 {
2   |
3   |   "nom": "Hugo",
4   |   "prenom": "Victor"
5   | }
```

Body Cookies Headers (5) Test Results 200 OK • 20 ms • 203 B

{ } JSON Preview Visualize

```
1 {
2   |   "id": 2,
3   |   "nom": "Hugo",
4   |   "prenom": "Victor"
5   | }
```

HTTP <http://localhost:8080/auteurs> Save Share

POST <http://localhost:8080/auteurs> Send

Params Authorization Headers (8) **Body** Scripts Tests Settings Cookies

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

```
1
2 {
3   |
4   |   "nom": "Camus",
5   |   "prenom": "Albert"
6   | }
```

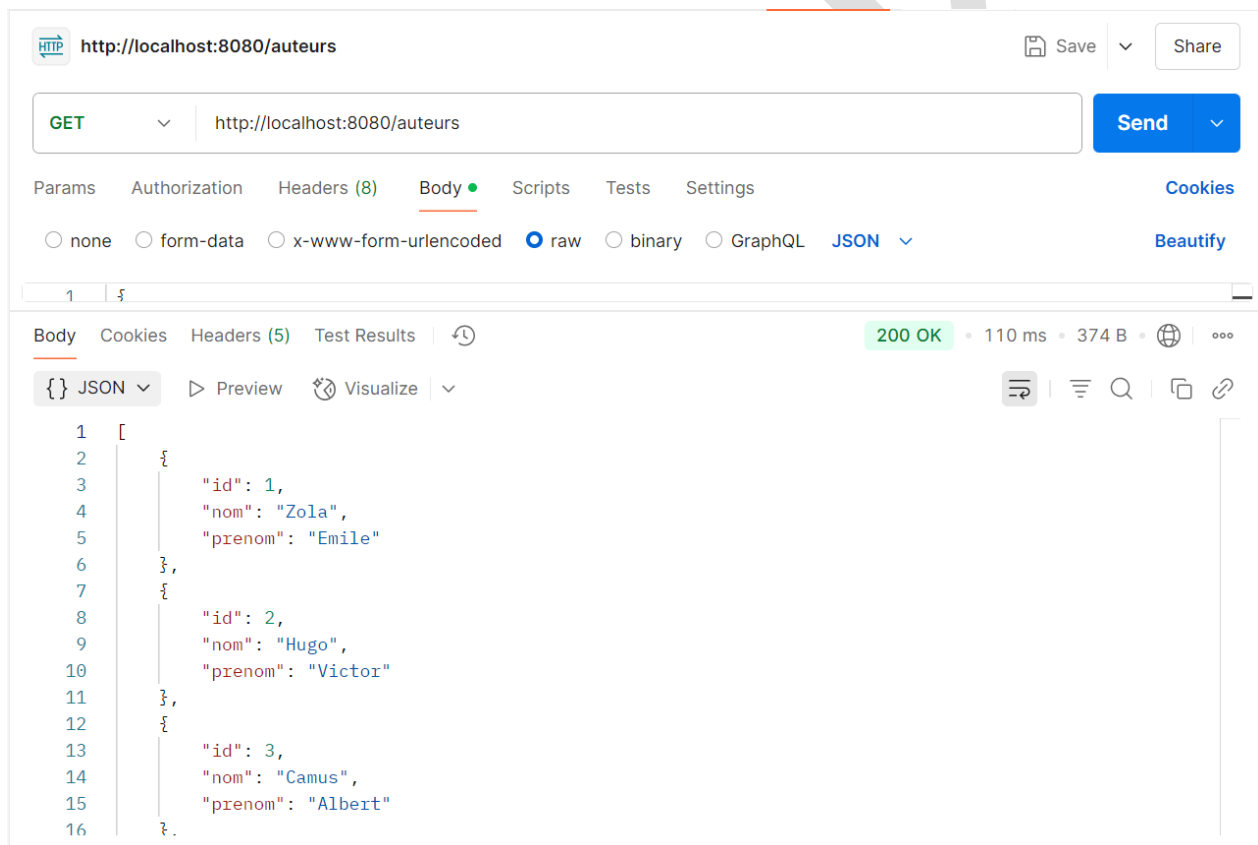
Body Cookies Headers (5) Test Results 200 OK • 13 ms • 204 B

{ } JSON Preview Visualize

```
1 {
2   |   "id": 3,
3   |   "nom": "Camus",
4   |   "prenom": "Albert"
5   | }
```

◆ Lister tous les auteurs:

Sur Postman:



Sur la Console H2:

localhost:8080/h2-console/login.do?jsessionId=644b596e148b9fe52872dd6e3f322eec

Auto commit ☒ Max rows: 1000 Auto complete Off Auto select On

jdbc:h2:mem:testdb

- AUTEUR
- LIVRE
- INFORMATION_SCHEMA
- Users

H2 2.1.214 (2022-06-13)

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM AUTEUR;

SELECT * FROM AUTEUR;

| ID | NOM | PRENOM |
|----|---------|---------------|
| 1 | Zola | Emile |
| 2 | Hugo | Victor |
| 3 | Camus | Albert |
| 4 | Ba | Mariama |
| 5 | Senghor | Leopold Sedar |

(5 rows, 3 ms)

- ◆ **Créer un livre lié à un auteur:**

HTTP <http://localhost:8080/livres/auteur/1> Save Share

POST <http://localhost:8080/livres/auteur/1> Send

Params Authorization Headers (8) **Body** Scripts Tests Settings Cookies

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

```
1 {
2   |   "titre": "Germinal",
3   |   "annee": 1885
4   | }
```

Body Cookies Headers (5) Test Results 200 OK • 62 ms • 252 B • 🌐 ⋮

{} **JSON** Preview Visualize ⋮

```
1 {
2   |   "id": 1,
3   |   "titre": "Germinal",
4   |   "annee": 1885,
5   |   "auteur": {
6   |     |   "id": 1,
7   |     |   "nom": "Zola",
8   |     |   "prenom": "Emile"
9   |   }
10  | }
```

HTTP <http://localhost:8080/livres/auteur/5> Save Share

POST <http://localhost:8080/livres/auteur/5> Send

Params Authorization Headers (8) **Body** Scripts Tests Settings Cookies

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

```
1
2
3 {
4   |   "titre": "Ethiopiennes",
5   |   "annee": 1956
6   | }
```

Body Cookies Headers (5) Test Results 200 OK • 13 ms • 266 B • 🌐 ⋮

{} **JSON** Preview Visualize ⋮

```
1 {
2   |   "id": 5,
3   |   "titre": "Ethiopiennes",
4   |   "annee": 1956,
5   |   "auteur": {
6   |     |   "id": 5,
7   |     |   "nom": "Senghor",
8   |     |   "prenom": "Leopold Sedar"
9   |   }
10  | }
```

HTTP

http://localhost:8080/livres/auteur/4

Save

Share

POST

http://localhost:8080/livres/auteur/4

Send

Params

Authorization

Headers (8)

Body

Scripts

Tests

Settings

Cookie:

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

Beautify

```
1
2
3 {
4   "titre": "Une Si Longue Lettre",
5   "annee": 1979
6 }
```

Body

Cookies

Headers (5)

Test Results

200 OK

14 ms

264 B

{}

JSON

Preview

Visualize

```
1 {
2   "id": 4,
3   "titre": "Une Si Longue Lettre",
4   "annee": 1979,
5   "auteur": {
6     "id": 4,
7     "nom": "Ba",
8     "prenom": "Mariama"
9   }
10 }
```

HTTP <http://localhost:8080/livres/auteur/2> Save Share

POST ▼ <http://localhost:8080/livres/auteur/2>

Send ▼

Params Authorization Headers (8) **Body** ● Scripts Tests Settings

Cookies

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

Beautify

```
1 {  
2   |  
3   |   "titre": "Les Misérable",  
4   |   "annee": 1862  
4   | }  
4 }
```

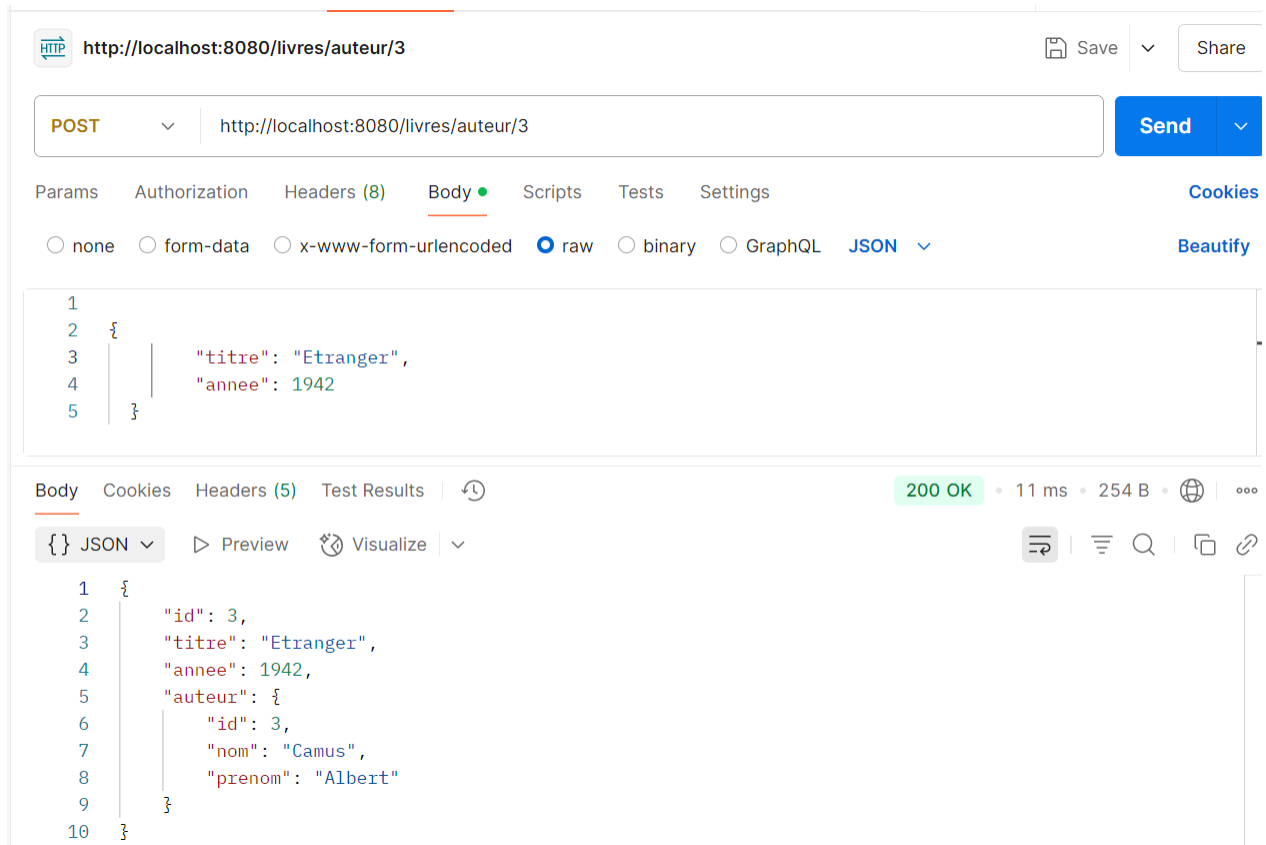
Body Cookies Headers (5) Test Results ↺

200 OK • 36 ms • 259 B • 🌐 ⋮

{ } JSON ▼ ▶ Preview 🔍 Visualize ▼

≡ ≡ 🔍 📄 🔗

```
1 {  
2   |  
3   |   "id": 2,  
4   |   "titre": "Les Misérable",  
5   |   "annee": 1862,  
6   |   "auteur": {  
7   |     |  
8   |     |   "id": 2,  
9   |     |   "nom": "Hugo",  
10  |     |   "prenom": "Victor"  
10  |     | }  
10  |   }  
10 }
```



◆ Lister tous les livres:

Sur Postman:

HTTP <http://localhost:8080/livres> Save Share

GET <http://localhost:8080/livres> Send

Params Authorization Headers (8) **Body** Scripts Tests Settings Cookies

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

1

Body Cookies Headers (5) Test Results 200 OK • 21 ms • 645 B

{ } JSON Preview Visualize

```
2 {
3   "id": 1,
4   "titre": "Germinal",
5   "annee": 1885,
6   "auteur": {
7     "id": 1,
8     "nom": "Zola",
9     "prenom": "Emile"
10  }
11 },
12 {
13   "id": 2,
14   "titre": "Les Misérable",
15   "annee": 1862,
16   "auteur": {
17     "id": 2,
```

```
16     "auteur": {
17       "id": 2,
18       "nom": "Hugo",
19       "prenom": "Victor"
20     }
21   },
22   {
23     "id": 3,
24     "titre": "Etranger",
25     "annee": 1942,
26     "auteur": {
27       "id": 3,
28       "nom": "Camus",
29       "prenom": "Albert"
30     }
31   }
32 }
```

```

32     {
33         "id": 4,
34         "titre": "Une Si Longue Lettre",
35         "annee": 1979,
36         "auteur": {
37             "id": 4,
38             "nom": "Ba",
39             "prenom": "Mariama"
40         }
41     },
42     {
43         "id": 5,
44         "titre": "Ethiopiennes",
45         "annee": 1956,
46         "auteur": {
47             "id": 5

```

```

48             "nom": "Senghor",
49             "prenom": "Leopold Sedar"
50         }
51     }
52 ]

```

Sur la Console H2:

jdbc:h2:mem:testdb

Auto commit ☒ Max rows: 1000 Auto complete Off Auto select On

Run Run Selected Auto complete Clear SQL statement:

SELECT * FROM LIVRE;

| ID | ANNEE | TITRE | AUTEUR_ID |
|----|-------|----------------------|-----------|
| 1 | 1885 | Germinal | 1 |
| 2 | 1862 | Les Misérables | 2 |
| 3 | 1942 | Etranger | 3 |
| 4 | 1979 | Une Si Longue Lettre | 4 |
| 5 | 1956 | Ethiopiennes | 5 |

(5 rows, 3 ms)

◆ Afficher les livres d'un auteur donné:

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

- URL:** `http://localhost:8080/livres/auteur/1`
- Method:** `GET`
- Body:** `raw` (selected)
- Response:** `200 OK`, `8 ms`, `254 B`
- JSON Body:**

```
1  [
2    {
3      "id": 1,
4      "titre": "Germinal",
5      "annee": 1885,
6      "auteur": {
7        "id": 1,
8        "nom": "Zola",
9        "prenom": "Emile"
10     }
11  }
12 ]
```

HTTP <http://localhost:8080/livres/auteur/5> Save Share

GET <http://localhost:8080/livres/auteur/5> Send

Params Authorization Headers (8) **Body** Scripts Tests Settings Cookies

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

1

Body Cookies Headers (5) Test Results 200 OK • 12 ms • 268 B

{ } JSON Preview Visualize

```
1 [
2   {
3     "id": 5,
4     "titre": "Ethiopiennes",
5     "annee": 1956,
6     "auteur": {
7       "id": 5,
8       "nom": "Senghor",
9       "prenom": "Leopold Sedar"
10    }
11  }
12 ]
```

◆ Modifier un livre:

HTTP <http://localhost:8080/livres/5> Save Share

PUT <http://localhost:8080/livres/5> Send

Params Authorization Headers (8) **Body** Scripts Tests Settings Cookies

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

```
1
2 {
3   "titre": "Ethiopiennes(édition modifier)",
4   "annee": 1956
5 }
```

Body Cookies Headers (5) Test Results 200 OK • 66 ms • 285 B • 🌐 ⋮

{} JSON Preview Visualize 🔍 📄 🔗

```
1 {
2   "id": 5,
3   "titre": "Ethiopiennes(édition modifier)",
4   "annee": 1956,
5   "auteur": {
6     "id": 5,
7     "nom": "Senghor",
8     "prenom": "Leopold Sedar"
9   }
10 }
```

HTTP <http://localhost:8080/livres/1> Save Share

PUT <http://localhost:8080/livres/1> Send

Params Authorization Headers (8) **Body** Scripts Tests Settings Cookies

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

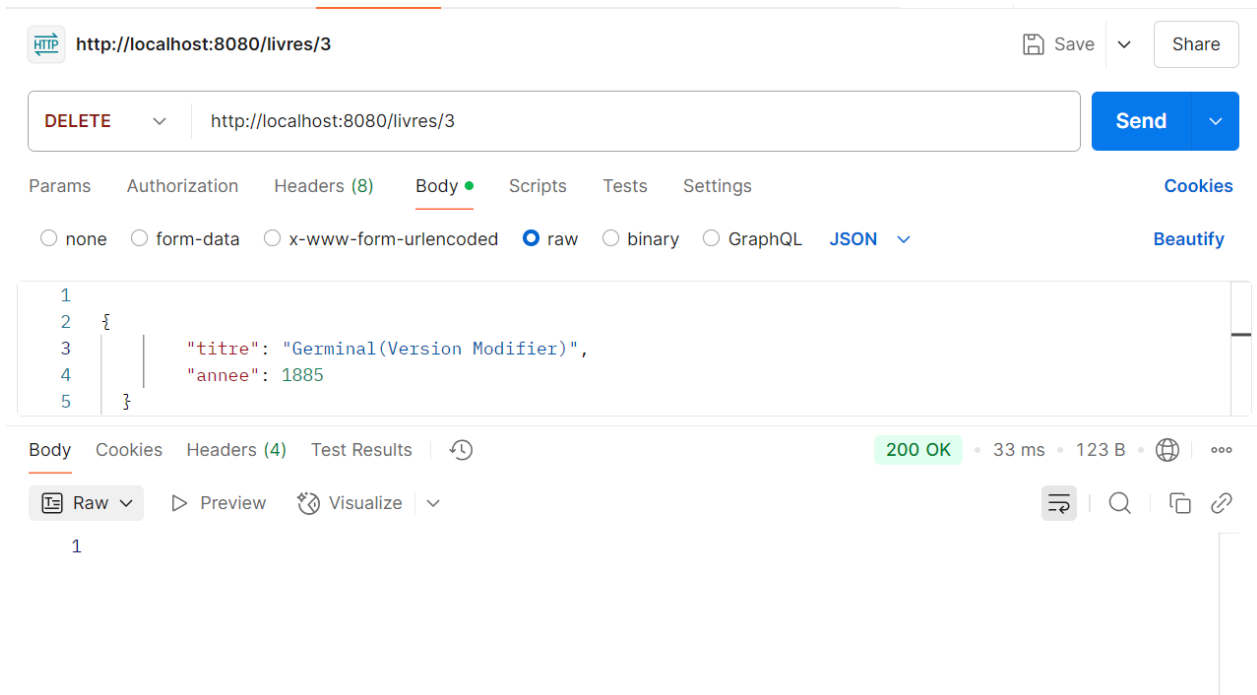
```
1
2 {
3   "titre": "Germinal(Version Modifier)",
4   "annee": 1885
5 }
```

Body Cookies Headers (5) Test Results 200 OK • 17 ms • 270 B • 🌐 ⋮

{} JSON Preview Visualize 🔍 📄 🔗

```
1 {
2   "id": 1,
3   "titre": "Germinal(Version Modifier)",
4   "annee": 1885,
5   "auteur": {
6     "id": 1,
7     "nom": "Zola",
8     "prenom": "Emile"
9   }
10 }
```

◆ Supprimer un livre:



CONCLUSION :

Ce projet m'a permis de mieux comprendre la création d'une API RESTful avec Spring Boot, la gestion des relations JPA, et l'importance de structurer proprement une application. Il a également renforcé mes compétences dans l'utilisation d'outils professionnels comme Swagger, Postman et Maven.

Les défis rencontrés, comme la gestion des relations entre entités et la pagination, m'ont permis de mieux comprendre la logique derrière Spring Data JPA. Ce projet constitue une base solide pour des évolutions futures,

notamment l'ajout de la sécurité avec Spring Security ou l'écriture de tests unitaires.

unchk

Unchik

Unchik