

# Projet Bookstore

## Introduction

Le projet consiste à créer un service web REST de type CRUD pour la recherche et l'emprunt de livres. Les ressources manipulées par les services sont des utilisateurs, des livres et des fiches d'emprunt.

En complément de ce service web REST, une application WEB utilisant le service web REST est fournie.

## Pourquoi choisir Spring Boot dans notre projet ?

Spring Boot est un framework open source pour le développement d'applications en Java. Il est conçu pour simplifier la création d'applications robustes et évolutives en fournissant un ensemble d'outils et de fonctionnalités prêts à l'emploi.

Spring Boot est basé sur le framework Spring, qui est un framework très populaire pour le développement d'applications en Java. Cependant, contrairement à Spring, Spring Boot est axé sur la création d'applications autonomes, c'est-à-dire des applications qui peuvent être exécutées en tant que processus indépendants, sans nécessiter de serveur d'application.

Spring Boot facilite la configuration de nombreuses fonctionnalités courantes d'une application, telles que la gestion de la configuration, la sécurité, la gestion des erreurs, la journalisation et bien plus encore. Il utilise également une architecture modulaire qui permet aux développeurs de choisir les fonctionnalités dont ils ont besoin et d'intégrer facilement d'autres frameworks et bibliothèques de leur choix.

## Installation et configuration

Le projet BookStore nécessite les éléments suivants :

- JDK 17 ou supérieur
- Apache Maven 3.2 ou supérieur
- IDE Eclipse ou IntelliJ IDEA

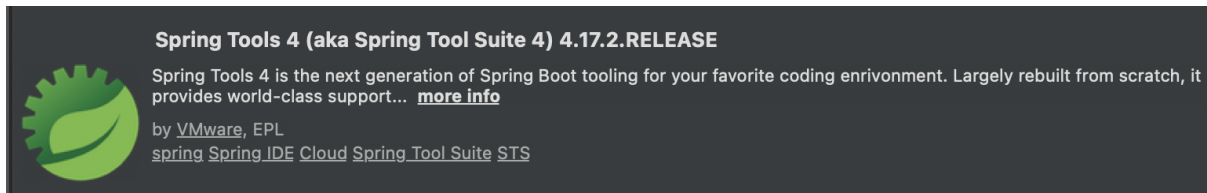
Pour installer et configurer le projet BookStore, suivez les étapes suivantes :

1. Téléchargez le code source de l'application BookStore depuis le dépôt Git.
2. Importez le projet dans votre IDE en tant que projet Maven.
3. Configurez les propriétés de la base de données dans le fichier "application.properties". Les propriétés à configurer sont :
  - spring.datasource.url : l'URL de la base de données
  - spring.datasource.username : le nom d'utilisateur de la base de données
  - spring.datasource.password : le mot de passe de la base de données

- `spring.mail.username` : l'email d'envoi
  - `spring.mail.password` : le mot de passe de l'email
4. Compilez le projet et générez le jar en exécutant la commande "mvn clean install" dans le terminal de votre IDE.

Sur IntelliJ, vous avez déjà Spring Boot Launcher d'installé par défaut dans l'IDE, vous avez juste besoin de lancer le projet.

Sur Eclipse, vous pouvez installer l'extension suivante permettant de lancer l'application facilement :



Sinon vous avez aussi la possibilité de déployer le projet sur le serveur d'application en exécutant la commande "mvn tomcat7:run" dans le terminal de votre IDE.

## Architecture de l'application

Les principales technologies utilisées dans l'application sont :

- Spring MVC : pour la gestion des requêtes HTTP et des réponses.
- Spring Data JPA : ORM (Object Relation Object) pour la persistance des objets en base de données.
- MySQL : pour le stockage des données.

Le schéma de base de données est le suivant :

- User : contient les informations sur les utilisateurs.
- Book : contient les informations sur les livres.
- BorrowingCard : contient les informations sur les fiches d'emprunt.

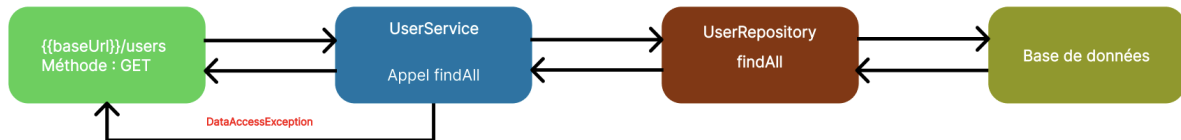
Chaque entité est associée à un repository pour la manipulation des données.

## Fonctionnalités de l'application

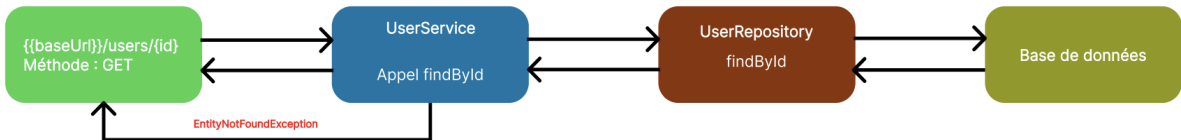
L'application BookStore permet de réaliser les opérations CRUD sur les entités User, Book et BorrowingCard.

Les endpoints REST exposés par l'application ainsi que leurs fonctionnements sont les suivants :

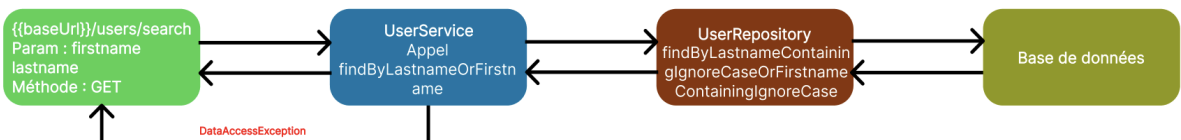
- /users (GET)
  - But : Récupérer tous les utilisateurs.



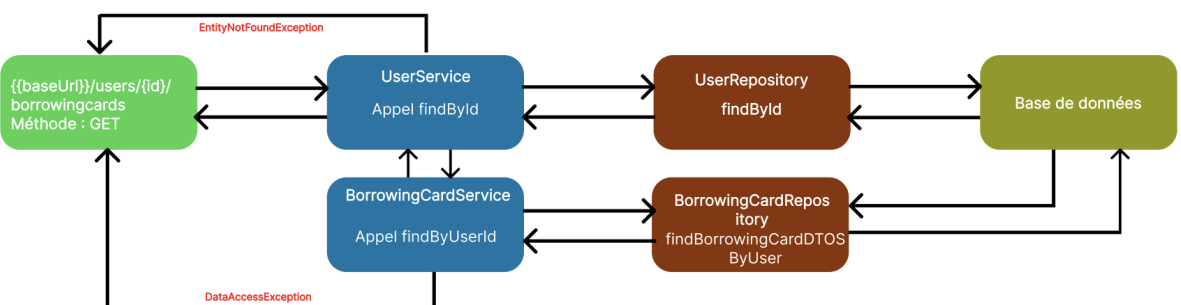
- /users/{id} (GET)
  - Path : {id} : Integer : Id : Id de l'utilisateur
  - But : Récupérer un utilisateur à partir de son ID.



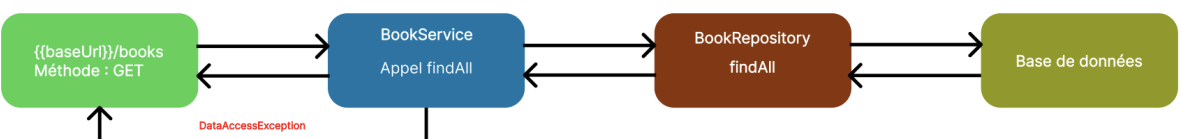
- /users/search (GET)
  - Param : Lastname (String) & Firstname (String)
  - But : Rechercher tous les utilisateurs à partir de son nom et prénom



- /users/{id}/borrowingcards (GET)
  - Path : {id} : Integer : Id de l'utilisateur
  - But : Rechercher les livres empruntés par un utilisateur

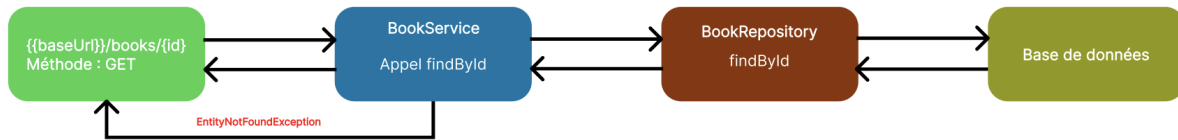


- /books (GET)
  - But : Récupérer tous les livres dans la bibliothèque



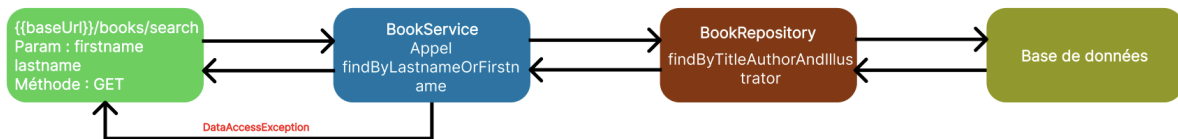
- /books/{id} (GET)
  - Path : {id} : String : Ean13 du livre

- But : Recherche un livre



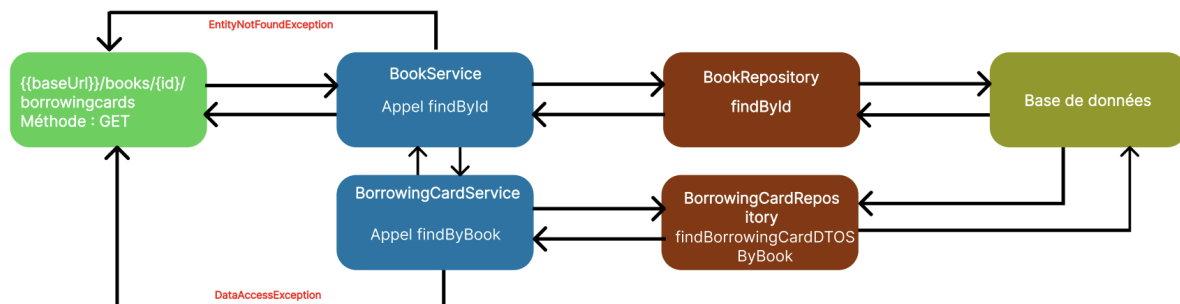
- /books/search (GET)

- Param : Title (String) & Author (String) & Illustrator (String)
- But : Rechercher des livres à partir de son titre, auteur ou illustrateur



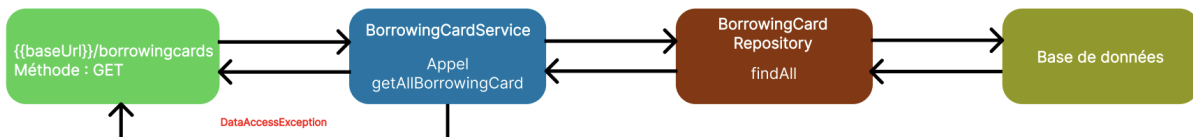
- /books/{id}/borrowingcards (GET)

- Path : String : Integer : Ean13 du livre
- But : Rechercher les réservations effectuées ou en cours sur ce livre



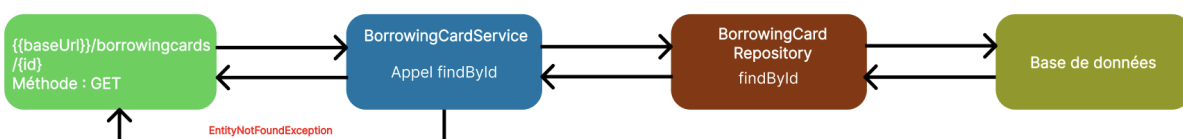
- /borrowingcards (GET)

- But : Récupérer toutes les emprunts

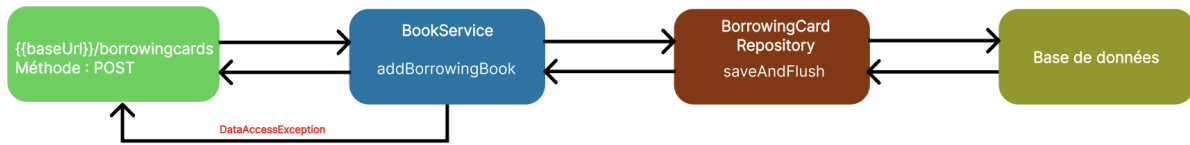


- /borrowingcards/{id} (GET)

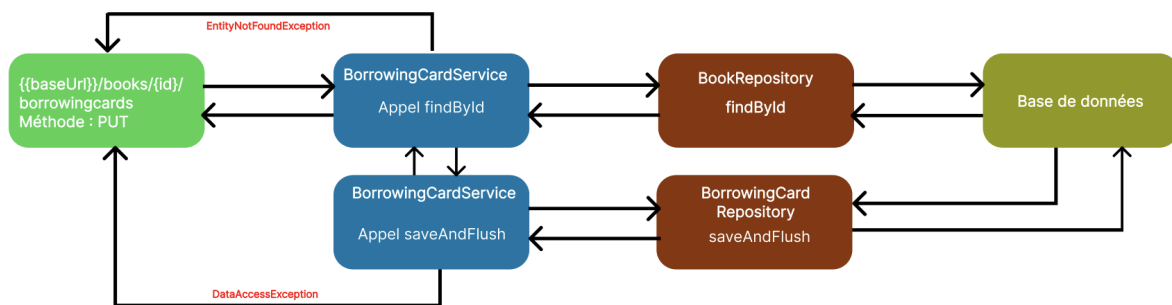
- Path : {id} : Integer : Id de l'emprunt
- But : Rechercher un emprunt à partir d'un id



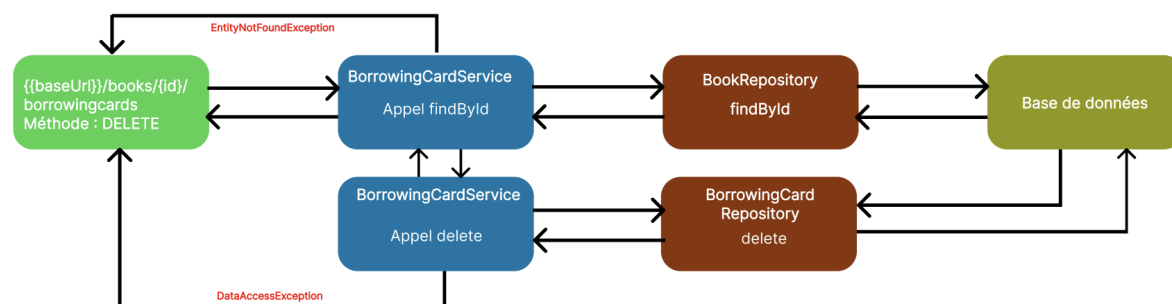
- /borrowingcard (POST)
  - Body : BorrowingcardModel : Information d'enregistrement d'emprunt
  - But : Enregistrer un emprunt



- /borrowingcard/{id} (PUT)
  - Path : id : Integer : Id de l'emprunt
  - Body : BorrowingcardModel : Information de mise à jour
  - But : Mettre à jour la date de rendu de l'emprunt



- /borrowingcard/{id} (DELETE)
  - Path : Id : Integer : Id de l'emprunt
  - But : Supprimer un emprunt



Dans le cadre de l'application BookStore, nous avons également ajouté la fonctionnalité d'envoi d'emails pour informer les abonnés lorsqu'un livre qu'ils ont emprunté doit être rendu.

Lorsqu'un livre est emprunté, une fiche d'emprunt (BorrowingCard) est créée avec la date d'emprunt et la date de retour prévue. Si la date de retour prévue est dépassée, un email est envoyé à l'abonné pour lui rappeler de rendre le livre.

Nous avons utilisé la dépendance spring starter mail pour envoyer les emails à partir de l'application. La fonctionnalité d'envoi de mails est implémentée dans la classe EmailSenderService, qui utilise l'interface JavaMailSender pour envoyer les mails.

Cette fonctionnalité permet de mieux communiquer avec les abonnés et de leur rappeler les dates de retour de leurs livres empruntés, ce qui contribue à améliorer la gestion de la bibliothèque.

## Books are late !



michalik.alexandre.17@gmail.com  
à : michalik.alexandre@orange.fr

10/03/23 21:59

détails



Hi Alexandre Michalik

This books are late !

Title	EAN 13	Loan Date	Max return date
sociis natoque penatibus	C1ECBB63-BD30-C2A1-6915-D8FA82C53DF8	2023-02-06	2023-02-10

## Tests des APIs sur POSTMAN

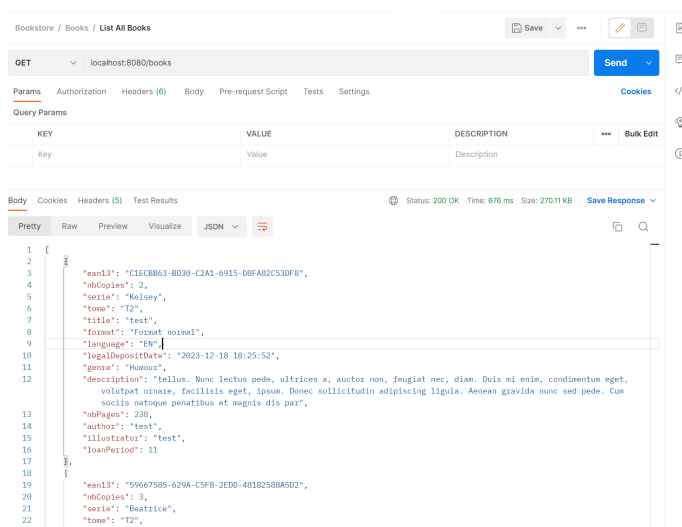
Pour tester les fonctionnalités de l'application BookStore, nous avons utilisé Postman, un outil permettant de tester les services web REST.

Un fichier Postman est présent dans le projet pour tester toutes les requêtes par vous même.

Voici les réponses aux requêtes POST, GET, PUT et DELETE pour chaque entité :

### Tests des requêtes pour les livres :

Liste de tous les livres :



Recherche d'un livre via ID: (ici on sélectionne d'abords un livre qui existe dont l'ID est 2 puis ensuite pour montrer notre gestion d'erreur nous avons choisi de sélectionner un livre qui n'existe pas)

Bookstore / Books / **Books by id** • From Successful operation

GET `{{baseUrl}}/books/2` Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 561 ms Size: 270.11 KB Save Response

Pretty Raw Preview Visualize JSON ...

```
1 {
2   "ean13": "C1ECBB63-8030-C2A1-6915-D8F82C530F8",
3   "nbCopies": 2,
4   "serie": "Kelsey",
5   "tome": "T2",
6   "title": "Test",
7   "format": "Format normal",
8   "language": "EN",
9   "legalDepositDate": "2023-12-10 18:25:52",
10  "genre": "Humour",
11  "description": "tellus. Nunc lectus pede, ultrices a, auctor non, feugiat nec, diam. Duis mi enim, condimentum eget,
12  voluptat ornare, facilisis eget, ipsum. Donec sollicitudin adipiscing ligula. Aenean gravida nunc sed pede. Cum
    sociis natoque penatibus et magnis dis par",
```

Nous avons pensé à faire une gestion d'erreur sur nos requêtes afin de prévenir lorsque la requête ne peut pas aboutir.

GET `{{baseUrl}}/books/1` Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 404 Not Found Time: 72 ms Size: 231 B Save Response

Pretty Raw Preview Visualize JSON ...

```
1 {
2   "status": "NOT_FOUND",
3   "code": 404,
4   "message": "Book not found"
5 }
```

Listes des livres par author ou/et par booktitle :

Juste avec les auteurs :

Bookstore / Books / Book search

GET `((baseUrl))/books/search?author=test&title=` Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION		Bulk Edit
<input checked="" type="checkbox"/>	author	test			
<input type="checkbox"/>	illustrator	test			
<input checked="" type="checkbox"/>	title				
	Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 567 ms Size: 270.11 KB Save Response

Pretty Raw Preview Visualize JSON

```
1  {
2    "ean13": "C1ECB863-B030-C2A1-6915-D8FA82C53DF8",
3    "nbCopies": 2,
4    "serie": "Kelsey",
5    "tone": "T2",
6    "title": "test",
7    "format": "Format normal",
8    "language": "EN",
9    "legalDepositDate": "2023-12-18 18:25:52",
10   "genre": "Humour",
11   "description": "Tellus. Nunc lectus pede, ultrices a, auctor non, feugiat nec, diam. Duis mi enim, condimentum eget,
12     volutpat ornare, facilisis eget, ipsum. Donec sollicitudin adipiscing ligula. Aenean gravida nunc sed pede. Cum
13     sociis natoque penatibus et magnis dis par",
14   "nbPages": 238,
15   "author": "test",
```

Juste par titre :

Bookstore / Books / Book search

GET `((baseUrl))/books/search?author=&title=test` Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION		Bulk Edit
<input checked="" type="checkbox"/>	author				
<input type="checkbox"/>	illustrator	test			
<input checked="" type="checkbox"/>	title	test			
	Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 9 ms Size: 709 B Save Response

Pretty Raw Preview Visualize JSON

```
1  {
2    "ean13": "C1ECB863-B030-C2A1-6915-D8FA82C53DF8",
3    "nbCopies": 2,
4    "serie": "Kelsey",
5    "tone": "T2",
6    "title": "test",
7    "format": "Format normal",
8    "language": "EN",
9    "legalDepositDate": "2023-12-18 18:25:52",
10   "genre": "Humour",
11   "description": "Tellus. Nunc lectus pede, ultrices a, auctor non, feugiat nec, diam. Duis mi enim, condimentum eget,
12     volutpat ornare, facilisis eget, ipsum. Donec sollicitudin adipiscing ligula. Aenean gravida nunc sed pede. Cum
13     sociis natoque penatibus et magnis dis par",
```

Avec les deux :



GET `{{baseUri}}/books/search?author=test&title=test` Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> author	test			
<input type="checkbox"/> illustrator	test			
<input checked="" type="checkbox"/> title	test			
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 11 ms Size: 709 B Save Response

Pretty Raw Preview Visualize JSON

```
3 {
4   "ean13": "C1ECB63-8030-C2A1-6915-D8FA82C53DF8",
5   "nbCopies": 2,
6   "serie": "Kelsey",
7   "tome": "T2",
8   "title": "test",
9   "format": "Format normal",
10  "language": "EN",
11  "legalDepositDate": "2023-12-18 18:25:52",
12  "genre": "Humour",
13  "description": "tellus. Nunc lectus pede, ultrices a, auctor non, feugiat nec, diam. Duis mi enim, condimentum eget,
14  volutpat ornare, facilisis eget, ipsum. Donec sollicitudin adipiscing ligula. Aenean gravida nunc sed pede. Cum
15  sociis natoque penatibus et magnis dis par",
16  "nbPages": 238,
17  "author": "test",
```

## Ajout de l'illustrateur :

GET `{{baseUri}}/books/search?author=test&illustrator=test&title=test` Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> author	test			
<input checked="" type="checkbox"/> illustrator	test			
<input checked="" type="checkbox"/> title	test			
Key	Value	Description		

Body Cookies Headers (5) Test Results Status: 200 OK Time: 23 ms Size: 709 B Save Response

Pretty Raw Preview Visualize JSON

```
5   "serie": "Kelsey",
6   "tome": "T2",
7   "title": "test",
8   "format": "Format normal",
9   "language": "EN",
10  "legalDepositDate": "2023-12-18 18:25:52",
11  "genre": "Humour",
12  "description": "tellus. Nunc lectus pede, ultrices a, auctor non, feugiat nec, diam. Duis mi enim, condimentum eget,
13  volutpat ornare, facilisis eget, ipsum. Donec sollicitudin adipiscing ligula. Aenean gravida nunc sed pede. Cum
14  sociis natoque penatibus et magnis dis par",
15  "nbPages": 238,
16  "author": "test",
17  "illustrator": "test",
18  "loanPeriod": 11
```

## Tests des requêtes pour les utilisateurs :

## Liste de tous les utilisateurs :

GET `{{baseUrl}}/users` [Send](#)

Params Authorization Headers (6) Body Pre-request Script Tests Settings [Cookies](#)

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results [Status: 200 OK](#) [Time: 1098 ms](#) [Size: 166.65 KB](#) [Save Response](#)

Pretty Raw Preview Visualize JSON [Send](#)

```
24 {
25   "id": 2,
26   "title": "M.",
27   "lastname": "Lynch",
28   "firstname": "Minerva",
29   "email": "nunc.interdum@yahoo.org",
30   "address": {
31     "id": 2,
32     "number": "59",
33     "lane": "Ap #790-4176 Et Ave",
34     "residence": "596-8111 Vivamus Street",
35     "building": "4576 In Rd.",
36     "apartmentNumber": "117",
37     "postalCode": "57636",
38     "town": "Kristiansund",
39     "country": "Norway"
40   }
41 },
42 {
43   "id": 3,
```

## Sélection d'un utilisateur par ID :

GET `{{baseUrl}}/users/id` [Send](#)

Params Authorization Headers (6) Body Pre-request Script Tests Settings [Cookies](#)

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Path Variables

KEY	VALUE	DESCRIPTION	...	Bulk Edit
id	1	Description		

Body Cookies Headers (5) Test Results [Status: 200 OK](#) [Time: 24 ms](#) [Size: 504 B](#) [Save Response](#)

Pretty Raw Preview Visualize JSON [Send](#)

```
1 {
2   "id": 1,
3   "title": "ME.",
4   "lastname": "test",
5   "firstname": "test",
6   "email": "ultrices.a@hotmail.edu",
7   "address": {
8     "id": 1,
9     "number": "235",
10    "lane": "5227 Faucibus Ave",
11    "residence": "6331 Non Ave",
12    "building": "Ap #947-9149 Tristique St.",
13    "apartmentNumber": "193",
14    "postalCode": "45417-28775",
15    "town": "Bremen",
16    "country": "Netherlands"
17  },
18   "borrowingCardIds": [
19     16,
20     479
21   ]
22 }
```

Recherche utilisateur par nom et ou prénom :

The screenshot shows a REST client interface with a GET request to `{{baseUrl}}/users/search?lastname=test&firstname=test`. The request is sent, and the response is displayed in JSON format. The response status is 200 OK, with a time of 15 ms and a size of 506 B.

**Query Params**

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> lastname	test	
<input checked="" type="checkbox"/> firstname	test	
Key	Value	Description

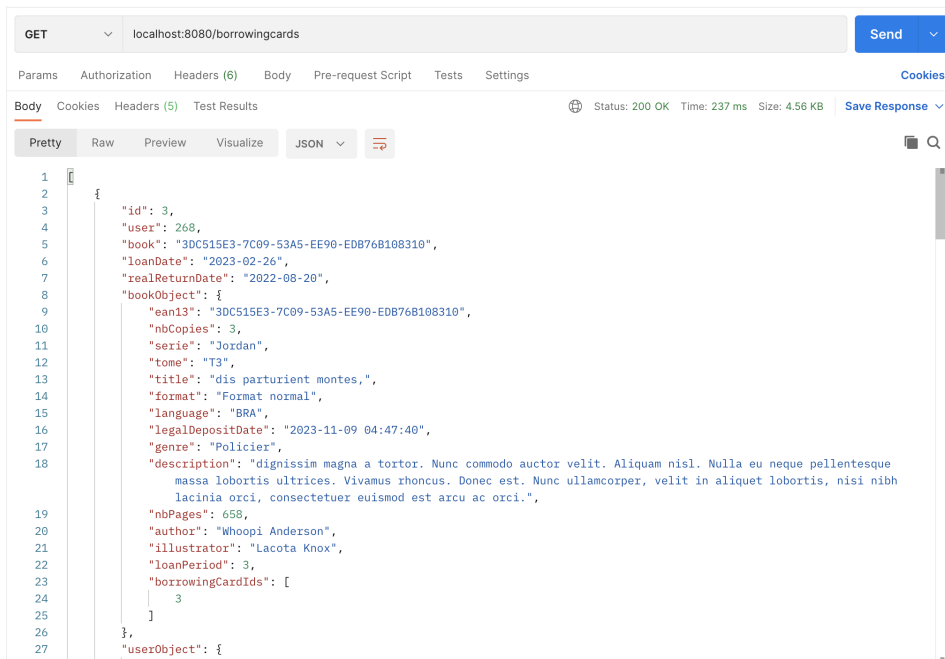
**Body**

JSON:

```
{
  "id": 1,
  "title": "MME.",
  "lastname": "test",
  "firstname": "test",
  "email": "ultrices.a@hotmail.edu",
  "address": {
    "id": 1,
    "number": "235",
    "lane": "5227 Faucibus Ave",
    "residence": "6331 Non Ave",
    "building": "Ap #947-9149 Tristique St.",
    "apartmentNumber": "193",
    "postalCode": "45417-20775",
    "town": "Bremen",
    "country": "Netherlands"
  },
  "borrowingCardIds": [
    14,
  ]
}
```

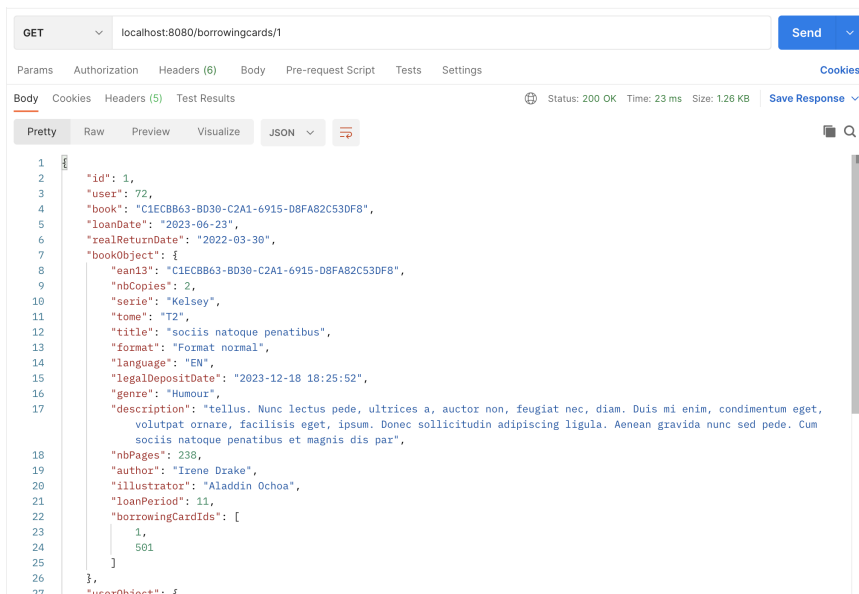
## Test des requêtes Borrowing Card :

Liste de tous les fiches emprunts d'un livre :



```
1 {
2   "id": 3,
3   "user": 268,
4   "book": "3DC515E3-7C09-53A5-EE90-EDB768108310",
5   "loanDate": "2023-02-26",
6   "realReturnDate": "2022-08-20",
7   "bookObject": {
8     "ean13": "3DC515E3-7C09-53A5-EE90-EDB768108310",
9     "nbCopies": 3,
10    "serie": "Jordan",
11    "tome": "T3",
12    "title": "dis parturient montes,",
13    "format": "Format normal",
14    "language": "BRA",
15    "legalDepositDate": "2023-11-09 04:47:40",
16    "genre": "Policier",
17    "description": "dignissim magna a tortor. Nunc commodo auctor velit. Aliquam nisl. Nulla eu neque pellentesque
18      massa lobortis ultrices. Vivamus rhoncus. Donec est. Nunc ullamcorper, velit in aliquet lobortis, nisi nibh
19      lacinia orci, consectetur euismod est arcu ac orci.",
20    "nbPages": 658,
21    "author": "Whoopi Anderson",
22    "illustrator": "Lacota Knox",
23    "loanPeriod": 3,
24    "borrowingCardIds": [
25      3
26    ]
27  },
28   "userObject": {
```

Liste des fiches emprunts d'un livre :



```
1 {
2   "id": 1,
3   "user": 72,
4   "book": "C1ECB863-BD30-C2A1-6915-D8FA82C530F8",
5   "loanDate": "2023-06-23",
6   "realReturnDate": "2022-03-30",
7   "bookObject": {
8     "ean13": "C1ECB863-BD30-C2A1-6915-D8FA82C530F8",
9     "nbCopies": 2,
10    "serie": "Kelsey",
11    "tome": "T2",
12    "title": "sociis natoque penatibus",
13    "format": "Format normal",
14    "language": "EN",
15    "legalDepositDate": "2023-12-18 18:25:52",
16    "genre": "Humour",
17    "description": "tellus. Nunc lectus pede, ultrices a, auctor non, feugiat nec, diam. Duis mi enim, condimentum eget,
18      volutpat ornare, facilisis eget, ipsum. Donec sollicitudin adipiscing ligula. Aenean gravida nunc sed pede. Cum
19      sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.",
20    "nbPages": 238,
21    "author": "Irene Drake",
22    "illustrator": "Aladdin Ochoa",
23    "loanPeriod": 11,
24    "borrowingCardIds": [
25      1,
26      501
27    ]
28  },
29   "userObject": {
```

Ajoute une réservation en prenant comme entrant un document JSON :

POST `{{baseUrl}}/borrowingcards...` Send

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

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

```
1 {
2   "id": 4,
3   "userId": 2,
4   "bookEAN13": "59667585-629A-C5F8-2EDD-48182588A5D2"
5 }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 71 ms Size: 1.26 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 502,
3   "user": 2,
4   "book": "59667585-629A-C5F8-2EDD-48182588A5D2",
5   "loanDate": "2023-03-18",
6   "bookObject": {
7     "ean13": "59667585-629A-C5F8-2EDD-48182588A5D2",
8     "nbCopies": 3,
9     "serie": "Beatrice",
10    "tome": "T2",
11    "title": "faucibus leo, in lobortis tellus",
12    "format": "Format normal",
13    "language": "FR",
14    "legalDepositDate": "2023-03-21 19:14:13",
15    "genre": "Dramatique",
16    "description": "tellus id nunc interdum feugiat. Sed nec metus facilisis lorem tristique aliquet. Phasellus fermentum convallis ligula. Donec luctus aliquet odio. Etiam ligula tortor, dictum eu, placerat eget, venenatis a, magna. Lorem ipsum dolor sit amet, consectetur ",
17  }
```

Met à jour la date de retour réelle en prenant en paramètre un document JSON :

PUT `{{baseUrl}}/borrowingcards/502` Send

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies

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

```
1 {
2   "realReturnDate": "2016-03-25"
3 }
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 307 ms Size: 1.28 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 502,
3   "user": 2,
4   "book": "59667585-629A-C5F8-2EDD-48182588A5D2",
5   "loanDate": "2023-03-18",
6   "realReturnDate": "2016-03-25",
7   "bookObject": {
8     "ean13": "59667585-629A-C5F8-2EDD-48182588A5D2",
9     "nbCopies": 3,
10    "serie": "Beatrice",
11    "tome": "T2",
12    "title": "faucibus leo, in lobortis tellus",
13    "format": "Format normal",
14    "language": "FR",
15    "legalDepositDate": "2023-03-21 19:14:13",
16    "genre": "Dramatique",
17    "description": "tellus id nunc interdum feugiat. Sed nec metus facilisis lorem tristique aliquet. Phasellus fermentum convallis ligula. Donec luctus aliquet odio. Etiam ligula tortor, dictum eu, placerat eget, venenatis a, magna. ",
18  }
```

Supprime la fiche d'emprunt dont l'id est celui passé en paramètre

DELETE

{{baseUrl}}/borrowingcards/502

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 20 ms

Size: 190 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1 SUCCESS DELETING BorrowingCard 502

## Répartition dans le projet

Tâches	Participants
création base de données (data)	Damien, Allan
création des dto	Damien, Paul-Emile
création des requêtes pour récupérer ce qu'on veut en bdd (repository)	Damien, Alexandre
création des endpoints (contrôleur)	Alexandre
gestion des erreurs	Damien
création différents services	Alexandre, Paul-Emile
création de la crontab + envoie d'email	Alexandre
création postman	Paul-Emile

création front	Allan
----------------	-------