Université Paris Descartes
UFR de Mathématiques et Informatique
Master 1 Informatique, Parcours : IAD

Rapport de projet de programmation web/distribuée

Réalisé par:

Assia BOURAÏ 21907038

Chargé de Cours/TD : Benoît CHARROUX

2019-2020

Tables des matières

Introduction

Ce projet a été conçu dans le cadre de l'évaluation des UEs PROGRAMMATION WEB et PROGRAMMATION DISTRIBUÉE.
Il met en pratique différentes notions étudiées tout au long du semestre, à savoir:

- la programmation web en utilisant Spring Boot.
- la sauvegarde des objets java dans des bases de données relationnelles en utilisant JPA.
- l'utilisation du framework Angular pour la construction des applications clients en utilisant HTML et TypeScript.
- la mise en place d'une architecture micro services dans les projets.
- l'utilisation de Docker et Kubernetes pour déployer les micro-services dans des containers et les faire communiquer entre eux.

Ce travail porte sur le développement d'un catalogue de livres en ligne. Ce dernier donne un aperçu sur la liste des livres disponibles, les utilisateurs de l'application ainsi qu'une liste détaillée de livres lus par chaque utilisateur.

1.Architecture générale

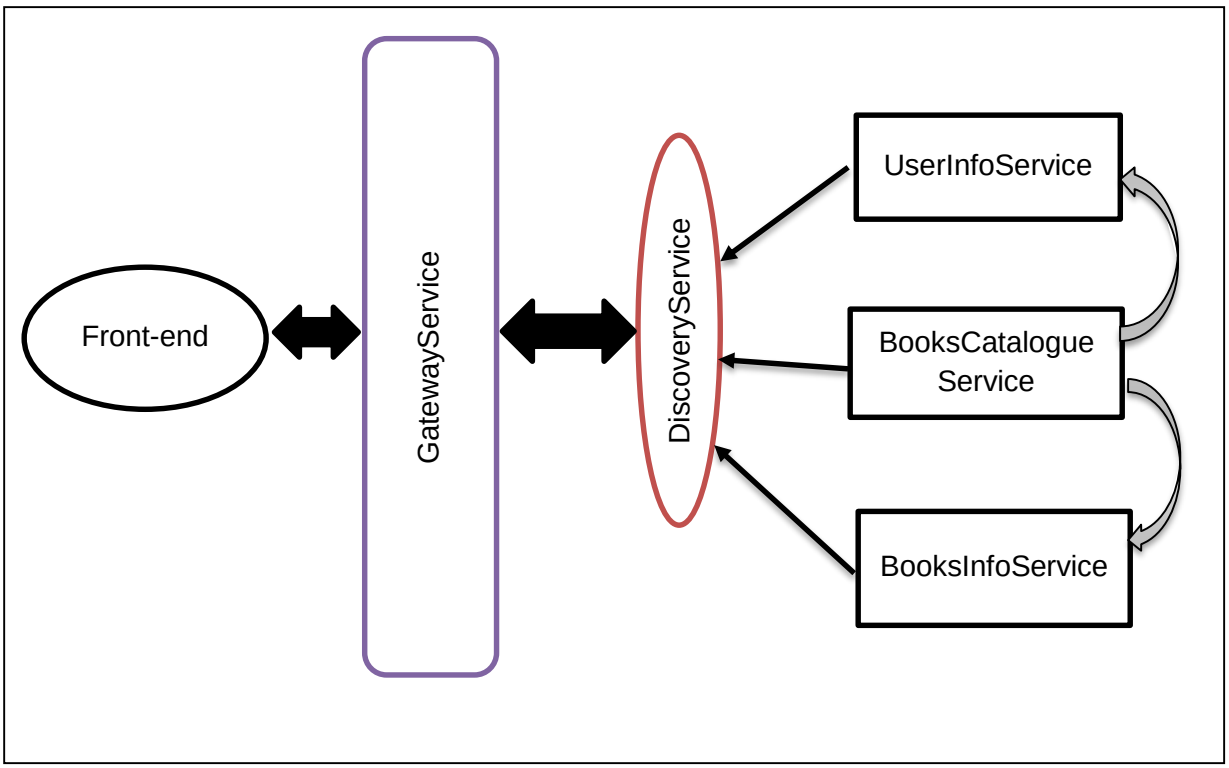
Concernant le côté serveur, le projet a été développé en suivant une architecture orientée micro-services, c'est à dire que l'application a été décomposée en plusieurs micro-services indépendants, à savoir:

- UserInfoService: web service qui permet de gérer les utilisateurs et possédant sa propre base de données. Il prend en entrée l'identifiant d'un utilisateur et renvoie les détails le concernant : nom, liste des identifiants des livres lus.
- BookInfoService: web service qui permet de gérer les livres et qui possède sa propre base de données. Il prend en entrée l'identifiant d'un livre et renvoie des détails à propos du livre: titre, nom de l'auteur, description.
- BooksCatalogueService: web service affichant la liste des livres de chaque utilisateur, il fait appel aux deux web services décrit précédemment, plus précisément, il prend en entrée l'identifiant d'un utilisateur, fait appel à "UserInfoService" pour obtenir la liste des livre lus, par la suite, il fait appel à "BookInfoService" pour obtenir la liste des détails de chaque livre lu.
- DiscoveryService: micro-service représentant l'annuaire d'enregistrement des micros services (registry), sauvegarde les noms des micro-services et les URLs de chacun.
- GatewayService: représente le point d'entrée de notre application, c'est ce micro-service qui permet la communication client-serveur. C'est avec ce service que le front codé en angular communique avec le côté serveur

Les bases de données relationnelles SQL ont été implémentées en utilisant Java Persistence API. Les deux bases sont in memory.

Pour ce qui est du front-end, c'est à dire le côté client de l'application. Ceci a été fait en HTML et TypeScript en utilisant le framework angular.

Le schéma ci-dessous résume le fonctionnement global de l'application.



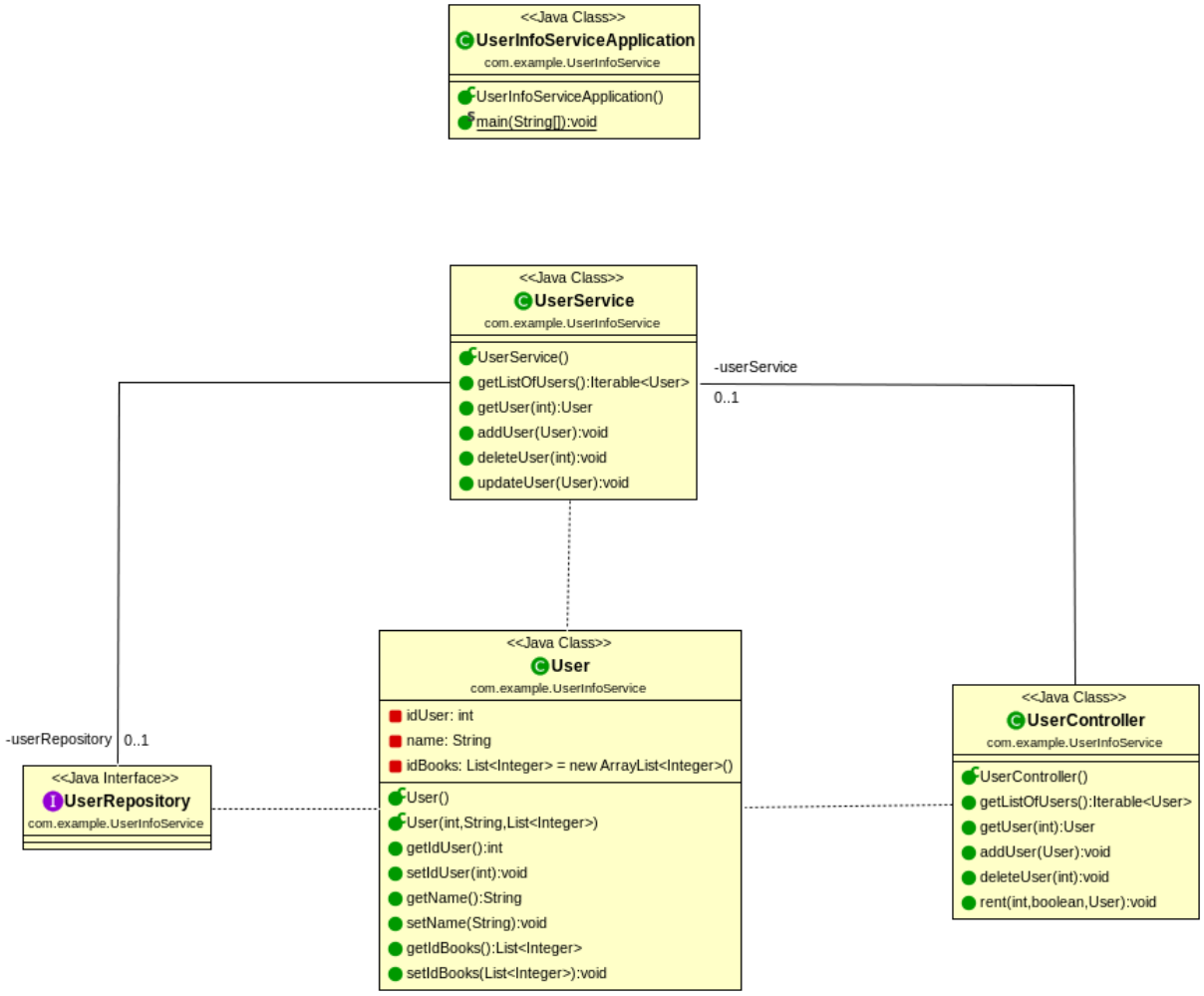
2.Diagrammes de classes UML

Dans ce qui suit, les diagrammes de classe de chaque micro-service sont représentés.

1. Micro-service “UserInfoService”

Ce micro-service se compose d’une interface, UserRepository qui hérite de CrudRepository et de quatre classes java:

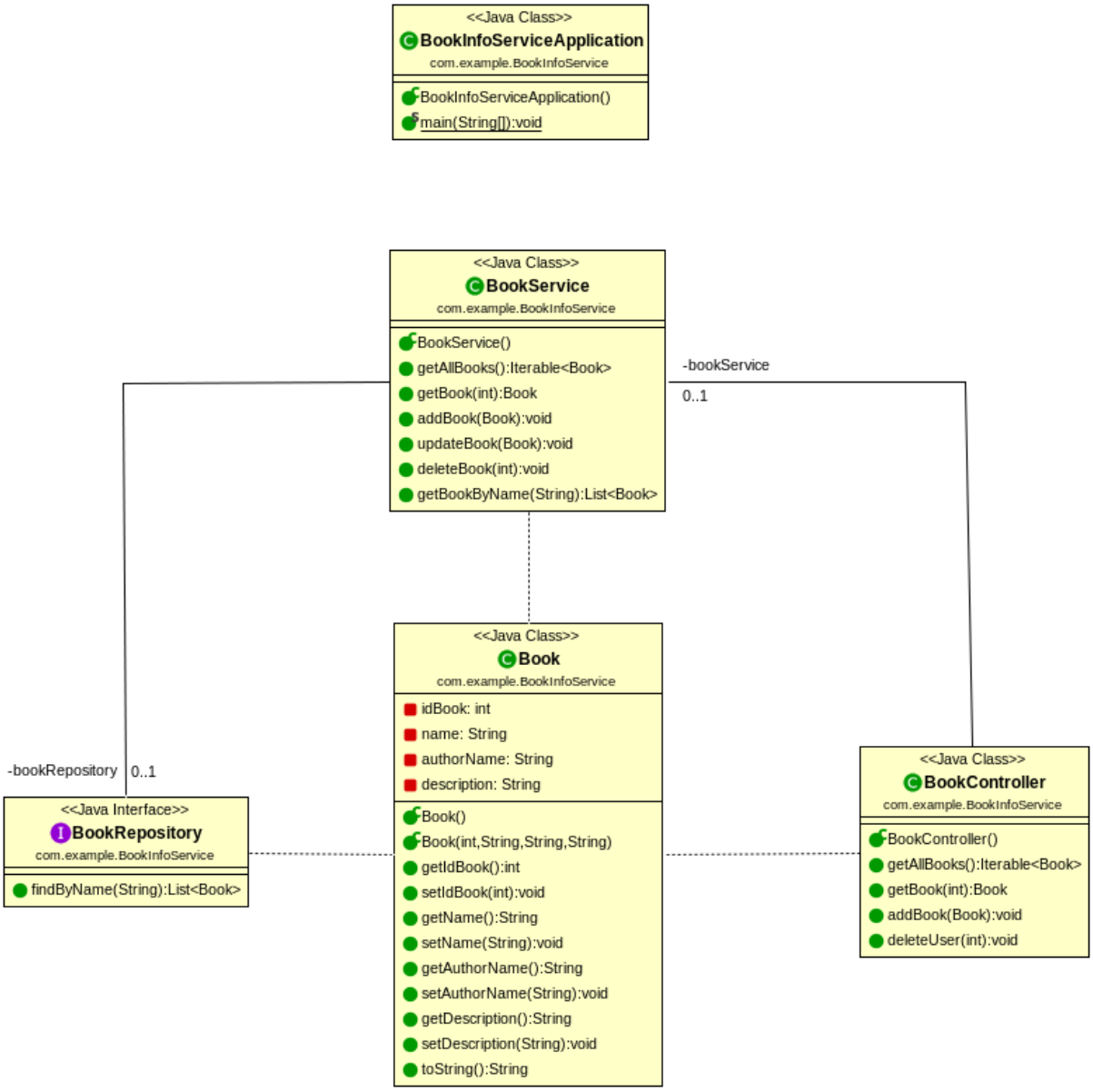
- UserInfoServiceApplication: classe disposant du main et qui permet de lancer l’application sur le port 8081.
- User: classe représentant le model du web service, il définit les attributs principaux.
- UserController: classe qui effectue les contrôles et traitements, c’est dans cette classe que sont définies les différentes méthodes GET, POST et DELETE.
- UserService: classe qui implémente l’interface citée précédemment, elle effectue les traitements en rapport avec la base de données.



2. Micro-service “BookInfoService”

Ce micro-service se compose d’une interface, BookRepository qui hérite de CrudRepository et de quatre classes java:

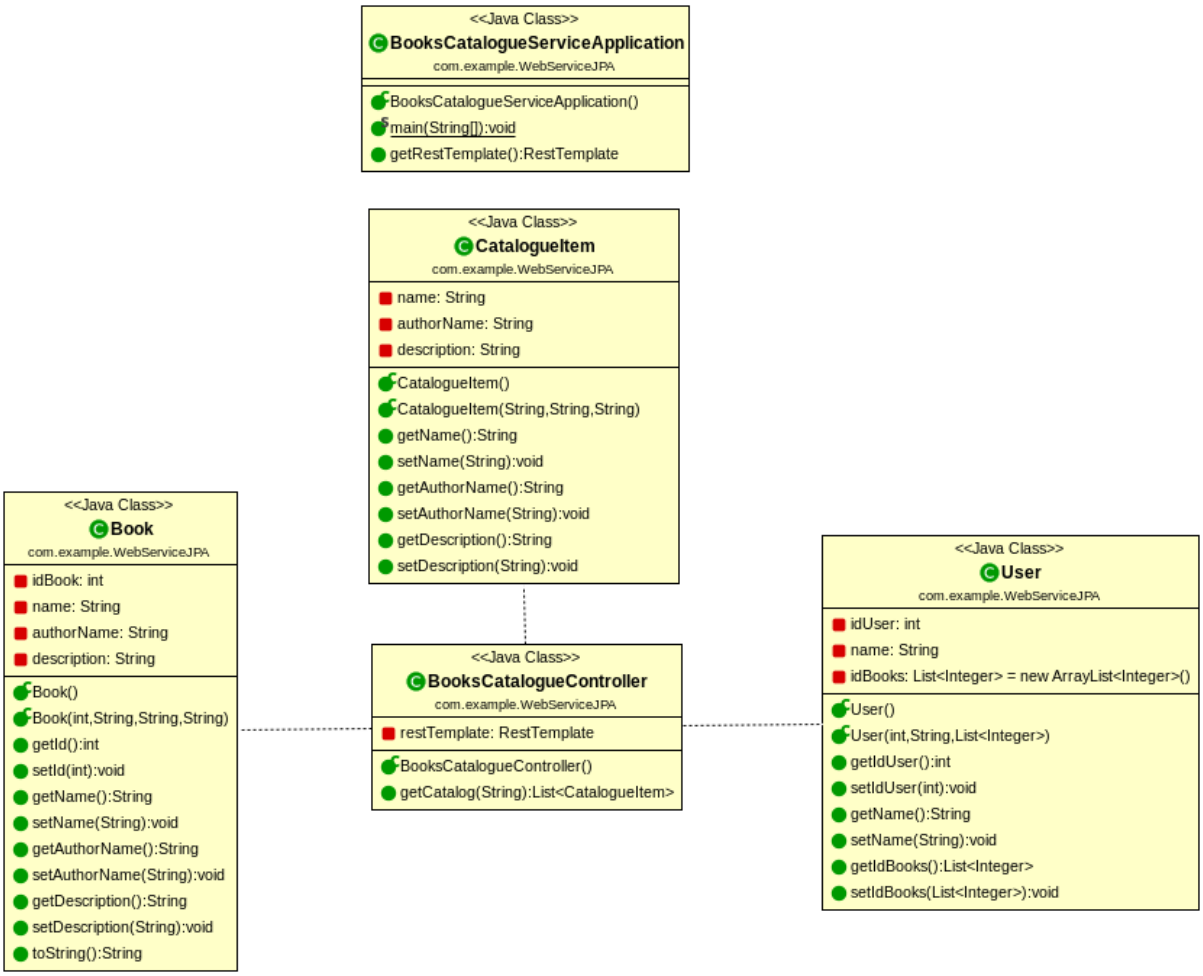
- BookInfoServiceApplication: classe disposant du main et qui permet de lancer l’application sur le port 8082.
- Book: classe représentant le model du web service, il définit les attributs principaux.
- BookController: classe qui effectue les contrôles et traitements, c’est dans cette classe que sont définies les différentes méthodes GET, POST et DELETE.
- BookService: classe qui implémente l’interface citée précédemment, elle effectue les traitements en rapport avec la base de données.



3. Micro-service “BooksCatalogueService”

Ce micro-service fait appel aux deux micro-services cités plus haut. Il est composé de:

- BooksCatalogueServiceApplication: classe disposant du main et qui permet de lancer l’application sur le port 8083.
- User et Book: classes représentant le model du web service, il définit les attributs principaux.
- BooksCatalogueController: classe qui effectue les contrôles et traitements, c’est dans cette classe que sont définies les différentes méthodes GET, POST et DELETE.
- CatalogueItem: classe qui définit la structure de l’objet renvoyé lors des requêtes html .



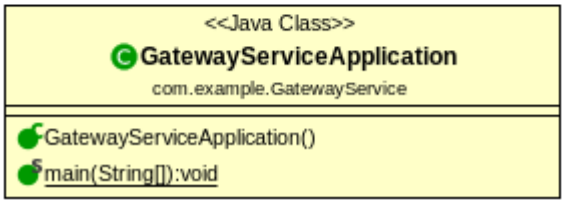
4. Micro-service “DiscoveryService”

Ce micro-service se lance sur le port 8761, il représente l’annuaire ou registry sur lequel tous les autres micro-services s’enregistrent.

5. Micro-service “GatewayService”

Ce micro-service se lance sur le port 8084, c’est le point d’entrée de l’application. La communication client-serveur est établie grâce à ce service.

Dans le fichier appalication.yml, les différentes routes vers les autres micro-services sont définies.



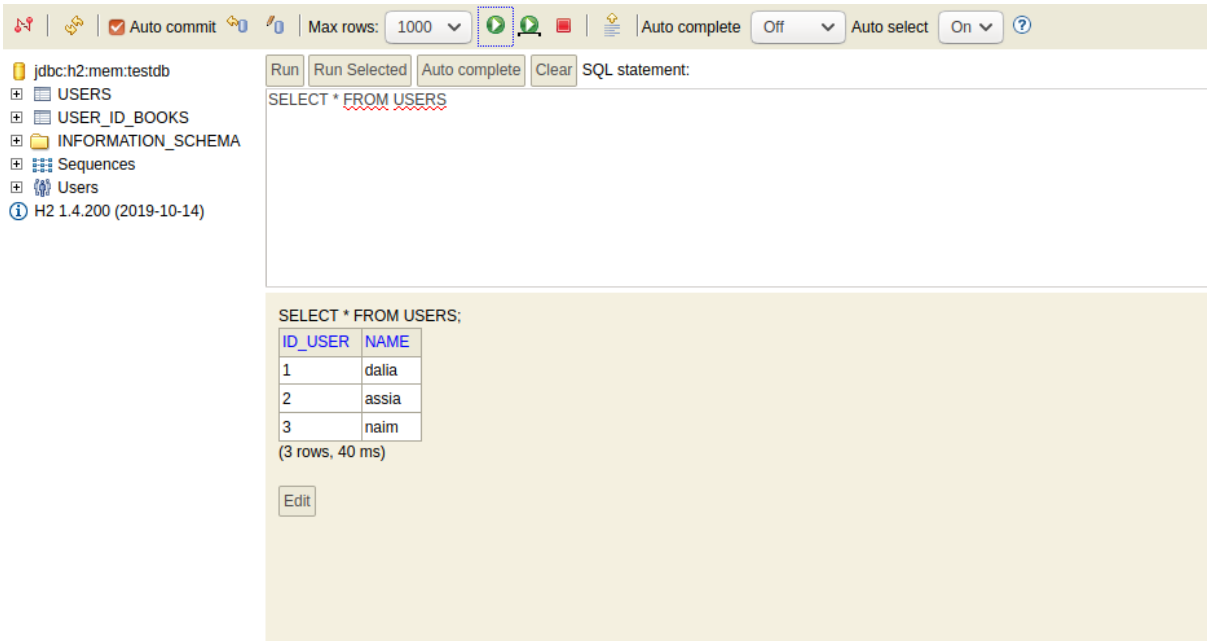
3.Bases de données

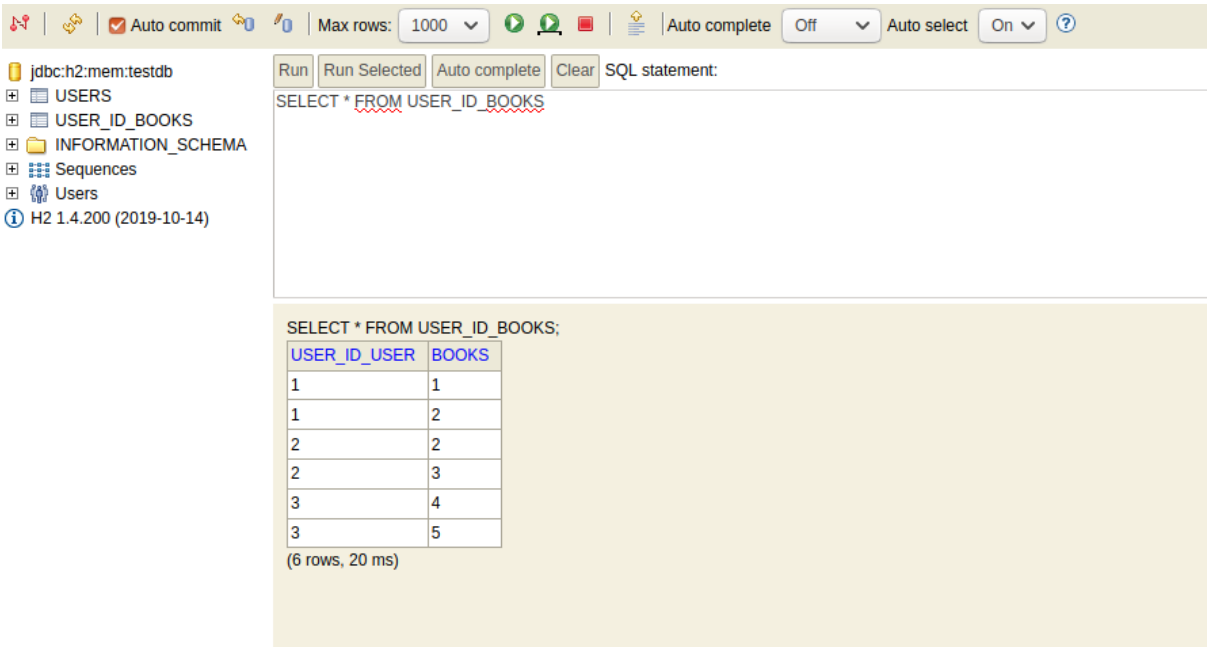
Comme je l’ai précédemment précisé, mon projet dispose de deux bases de données principales, une pour stocker les utilisateurs et une pour les livres.

La base de données des utilisateurs se compose de deux tables:

- USERS qui stocke l’identifiant de l’utilisateur (ID_USER) qui est généré automatiquement ainsi que son nom (NAME).
- USER_ID_BOOKS qui stocke pour chaque utilisateur l’identifiant du livre lli.

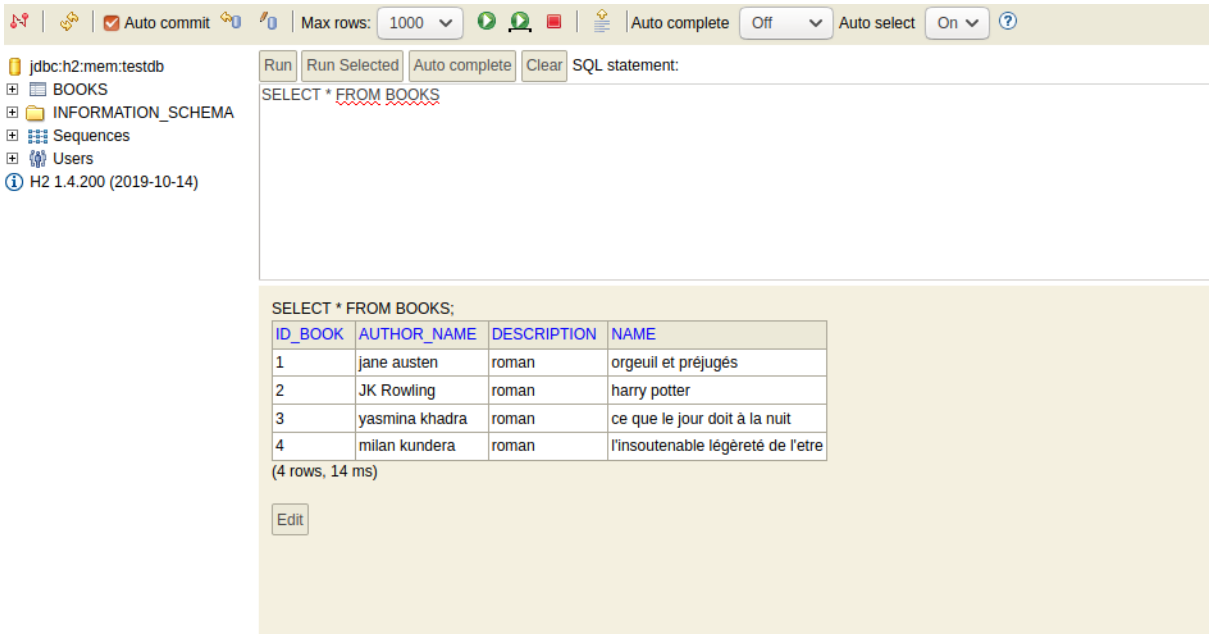
Les deux figures qui suivent donnent un aperçu de ces deux tables.





La BDD qui stocke les données des livres, illustrée dans la figure qui suit, est composée de quatre colonnes:

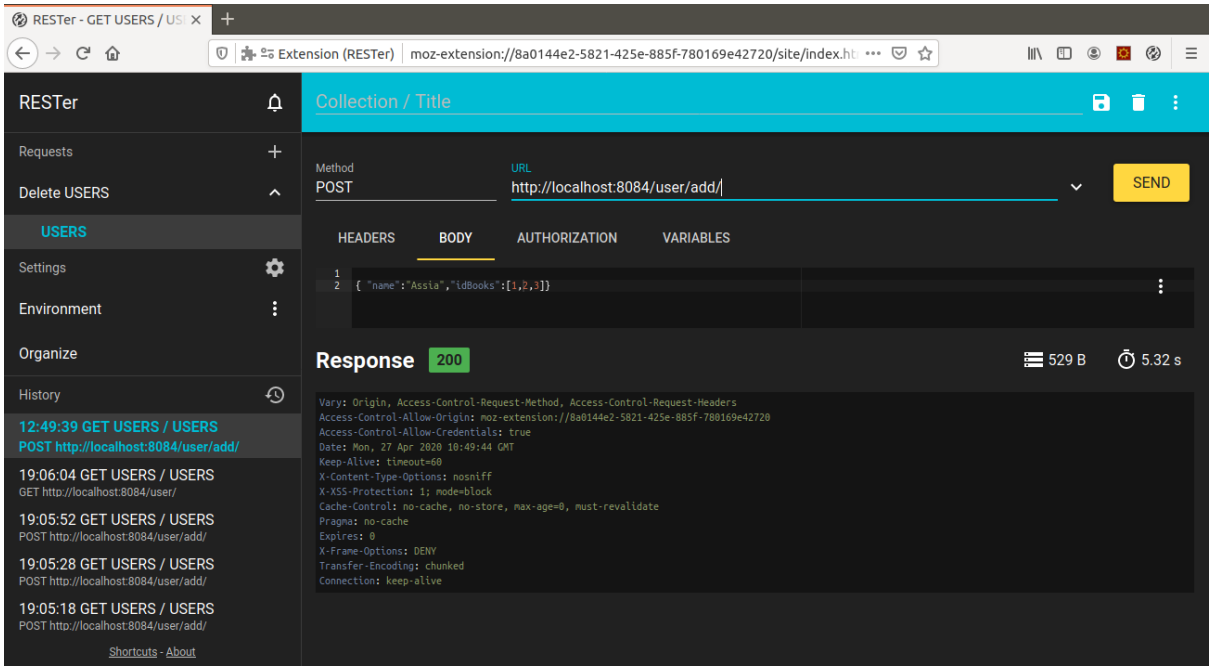
- ID_BOOK, identifiant du livre, entier généré automatiquement.
- AUTHOR_NAME, nom de l'auteur du livre.
- DESCRIPTION, description du livre.
- NAME, titre du livre.



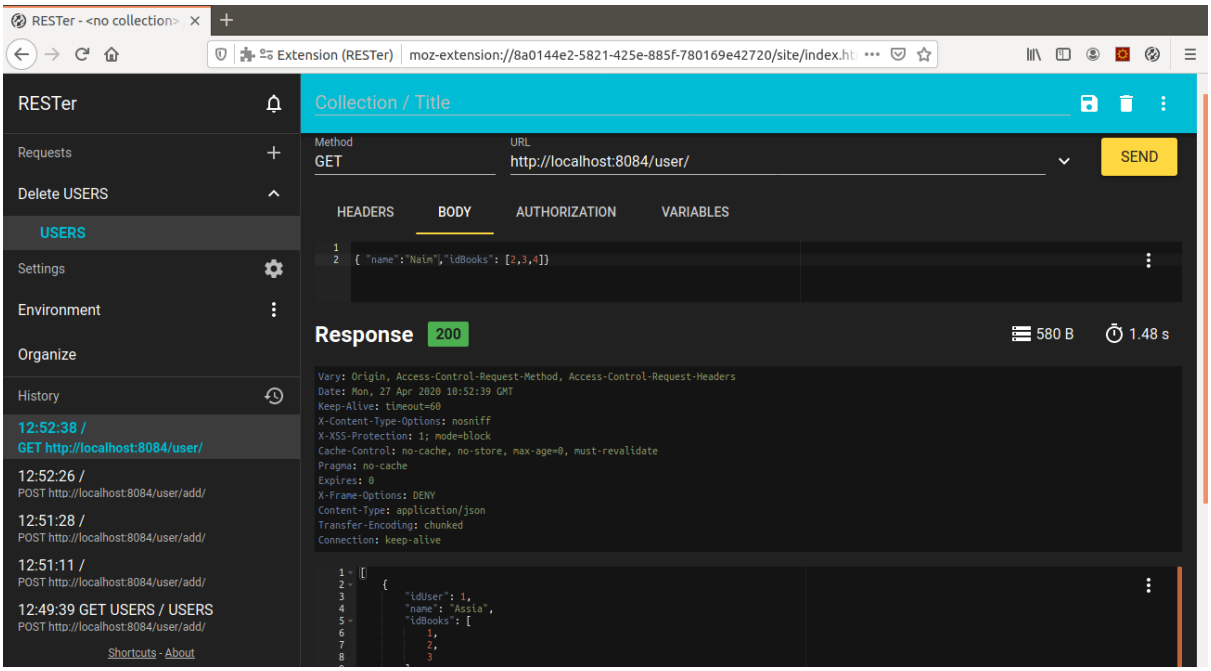
4.Réponses des web services aux requêtes http

Le back end de mon application fonctionne très bien dans sa globalité.
Il est à noter que la communication entre les différents micro-services a été faire en utilisant RestTemplate.
Les requêtes GET, POST et DELETE ont été testées en utilisant le plugin RESTer.
Dans ce qui suit, je présente les différentes réponses obtenues lors des différentes requêtes lancées.

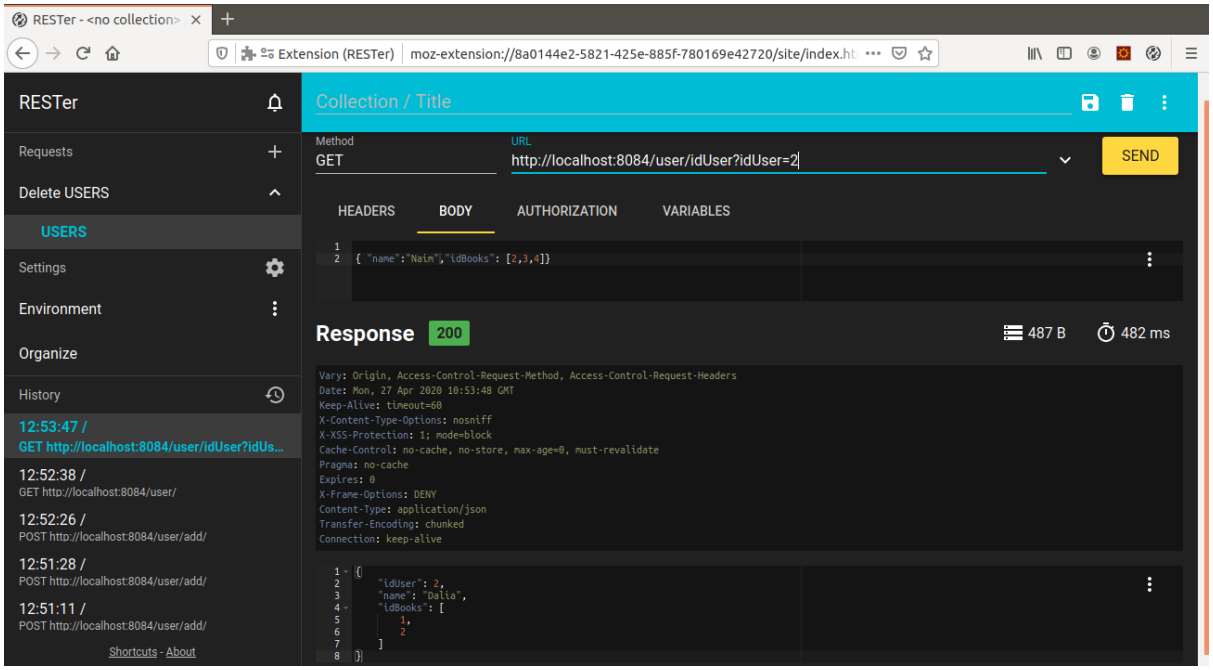
- **Ajout d'un utilisateur:**



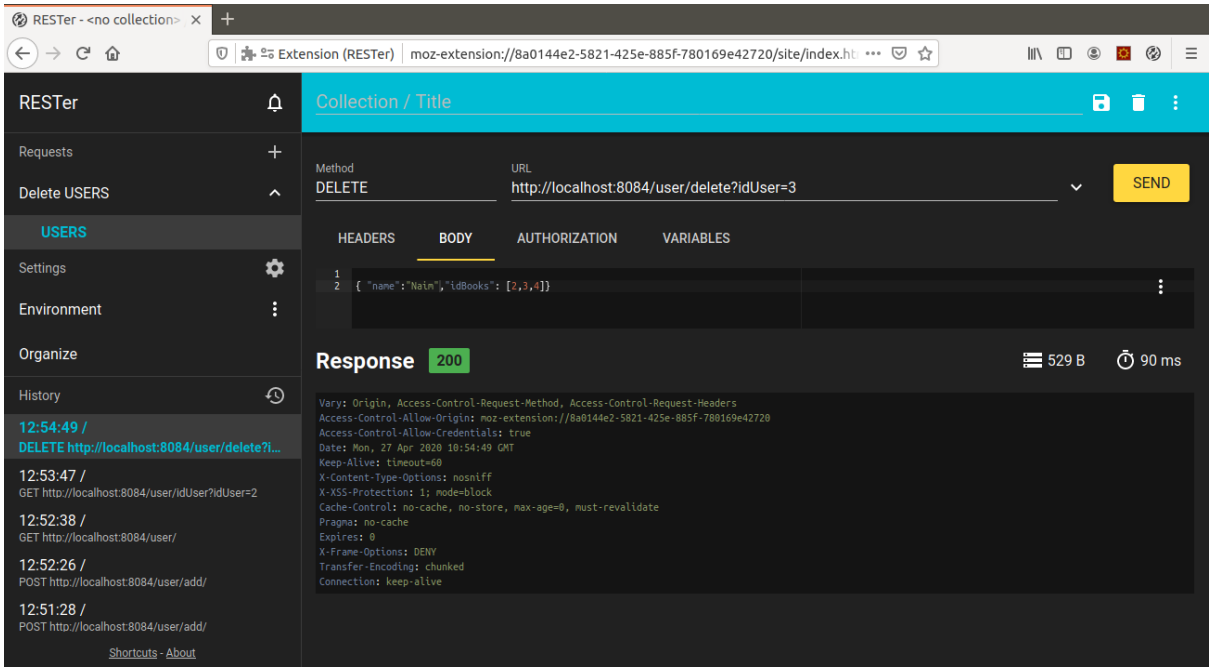
- **Afficher tous les utilisateurs**



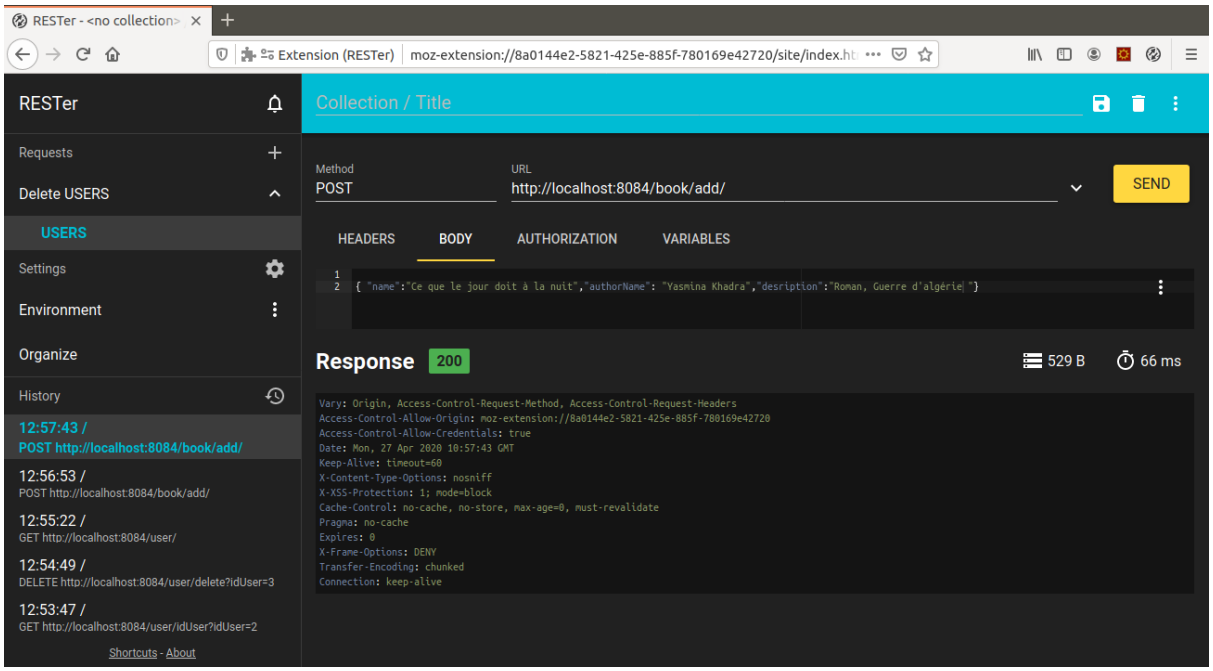
- **Afficher un utilisateur en spécifiant son identifiant**



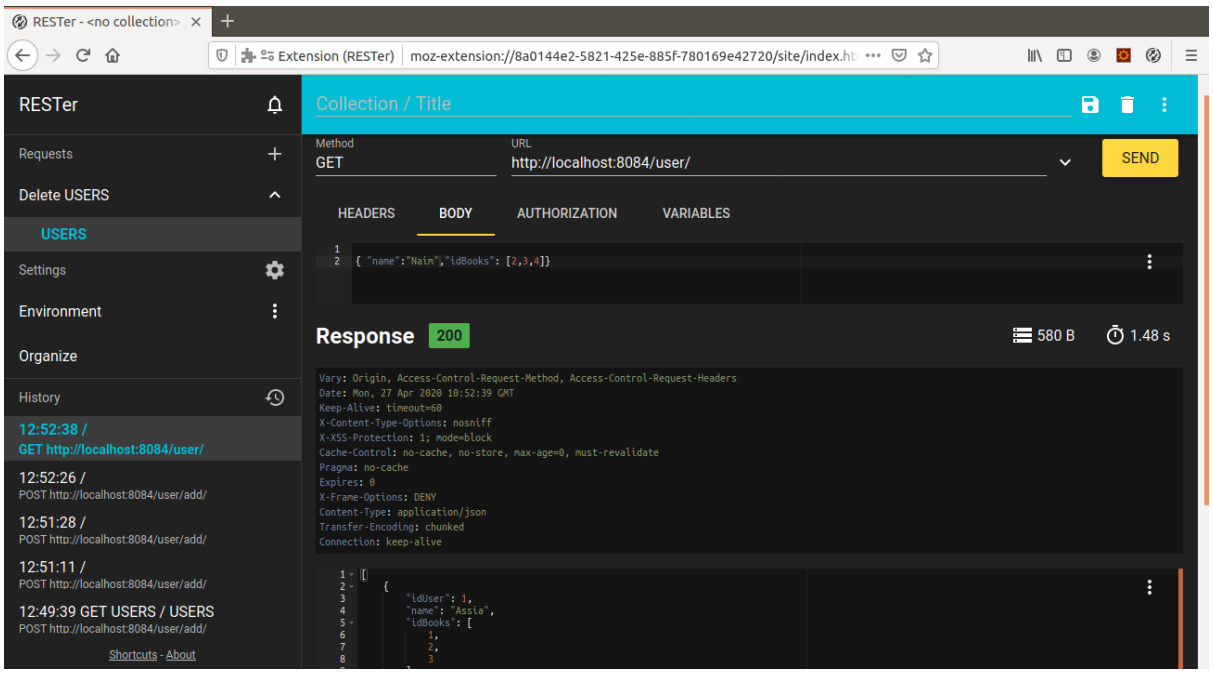
- **Supprimer un utilisateur**



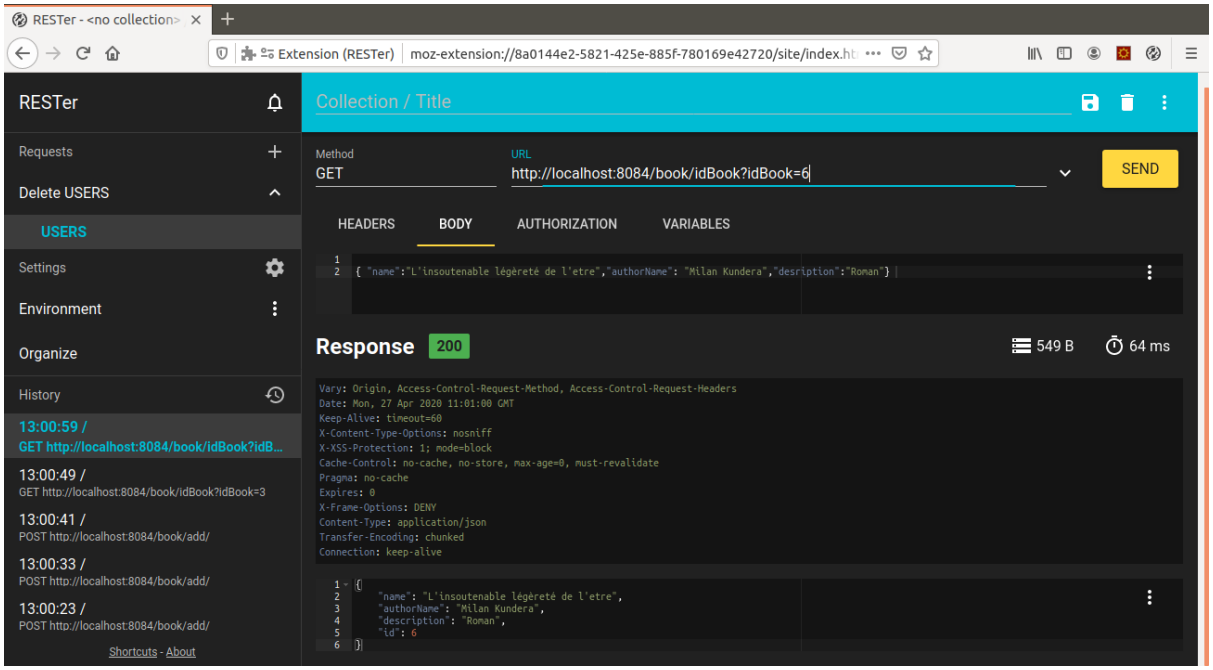
- **Ajouter un livre**



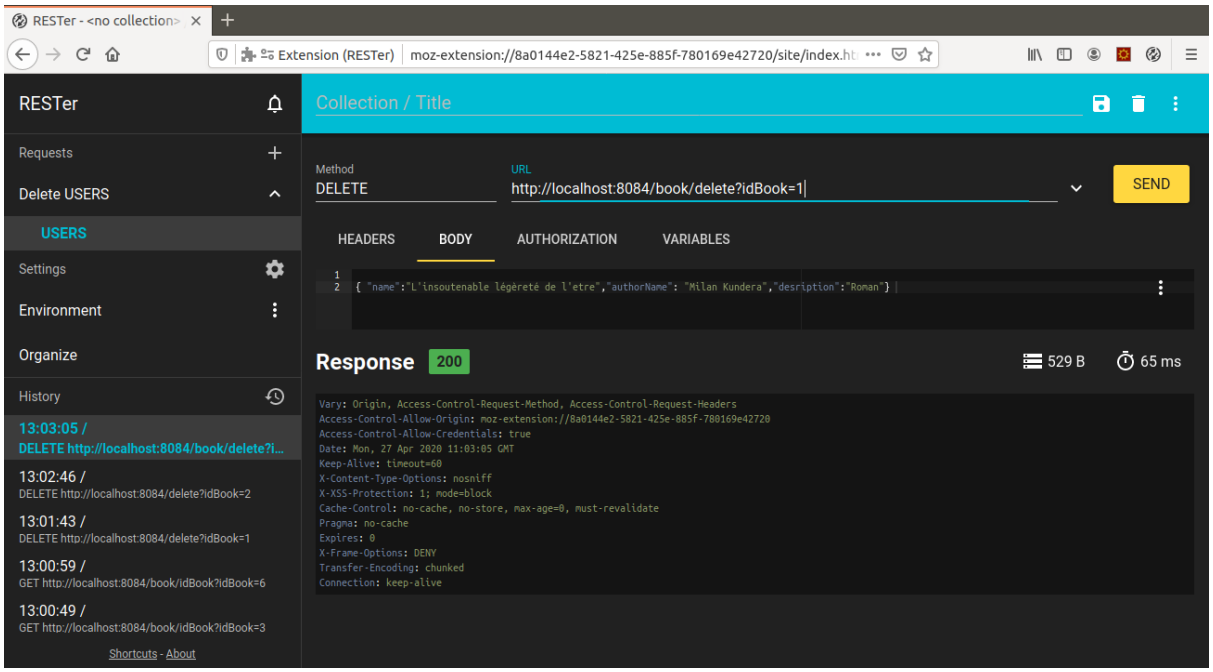
- **Afficher tous les livres**



- **Afficher un livre**



- **Supprimer un livre**



5.Interfaces côté client

Les interfaces de l'application ont été codées en Angular.

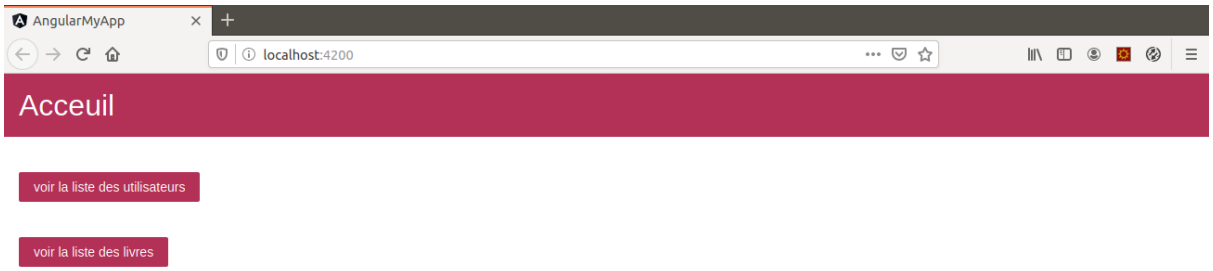
Etant débutante dans l'utilisation de ce framework j'ai suivi le tutoriel "Getting Started" d'Angular comme vous l'avez conseillé.

J'ai rencontré plusieurs difficultées à faire communiquer les micro-services ainsi que le serveur Angular (ordinateur pas du tout performant), c'est pour cela que je n'ai pas pu implémenter toutes les fonctionnalités du back-end correctement et que je n'ai pas présenté un travail très satisfaisant. Je vous ai d'ailleurs envoyé un mail pour vous faire part de mes difficultées.

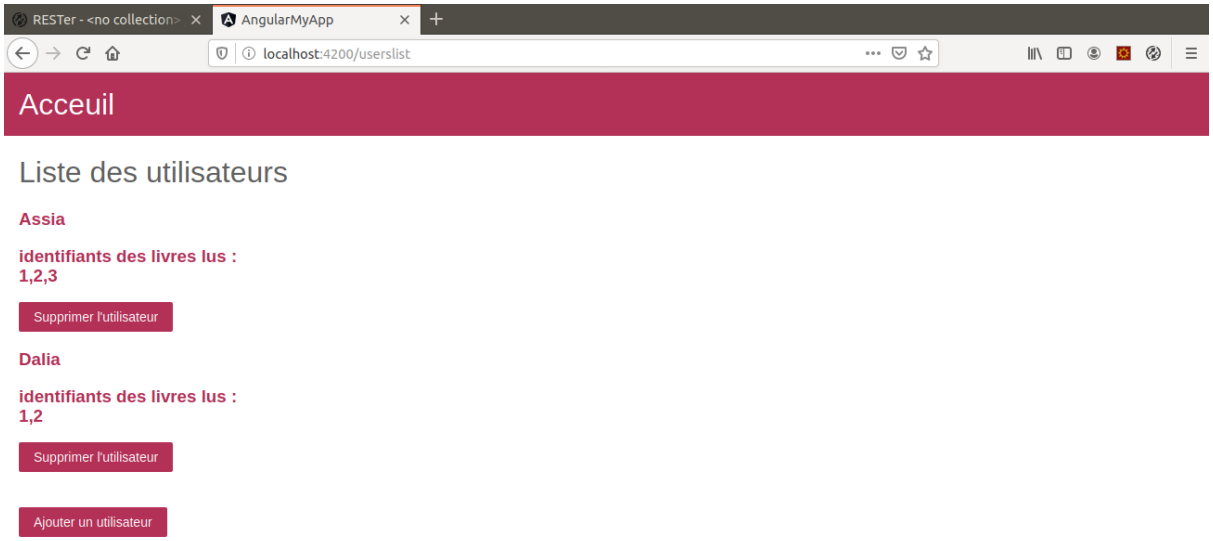
L’affichage de la liste des utilisateurs /livres fonctionne bien.
Pour pouvoir afficher cette liste j’ai dû faire un POST via le plugin RESTer, car malgré tous mes efforts je n’ai pas réussi à faire fonctionner l’ajout d’un utilisateur/livre .
La suppression d’un utilisateur/livre quant à elle fonctionne bien
La vue concernant l’affichage du catalogue de livres de chaque utilisateur (micro-service BooksCatalogService) n’a pas pu être implémentée.

Les différentes interfaces de l’application sont représentés dans les captures d’écran qui suivent.

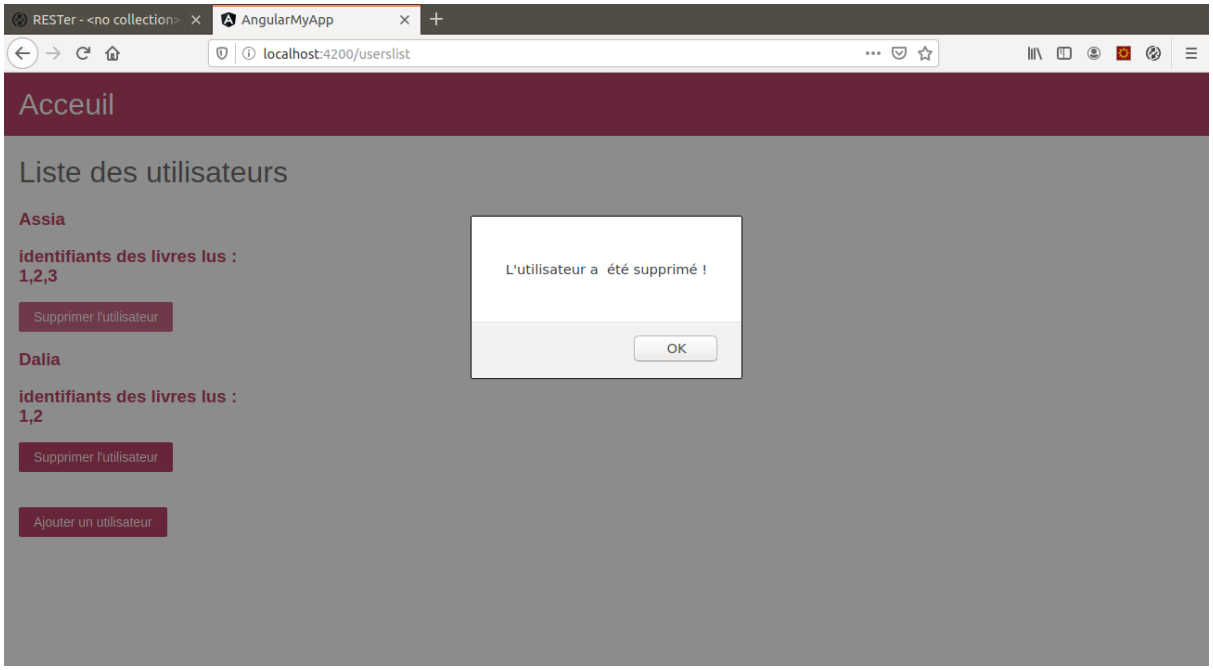
- **Accueil**

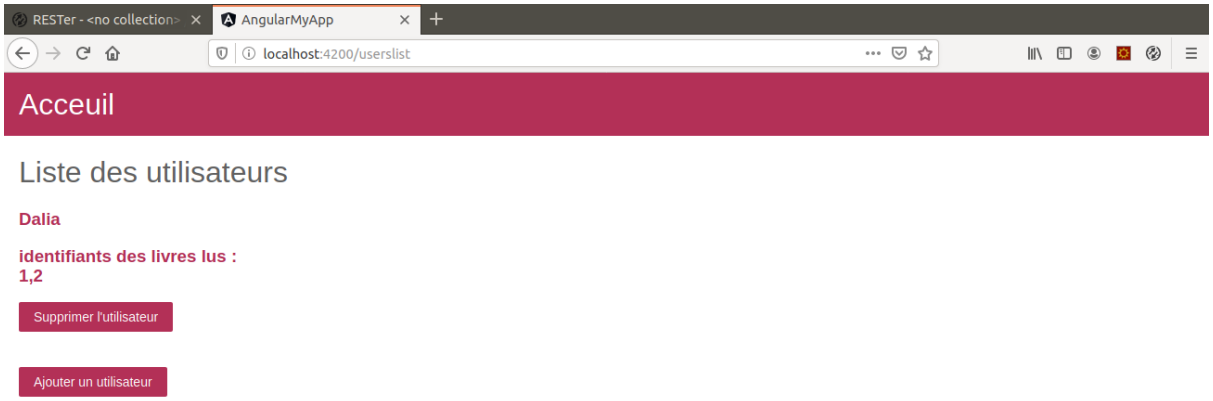


- **Liste des utilisateurs**

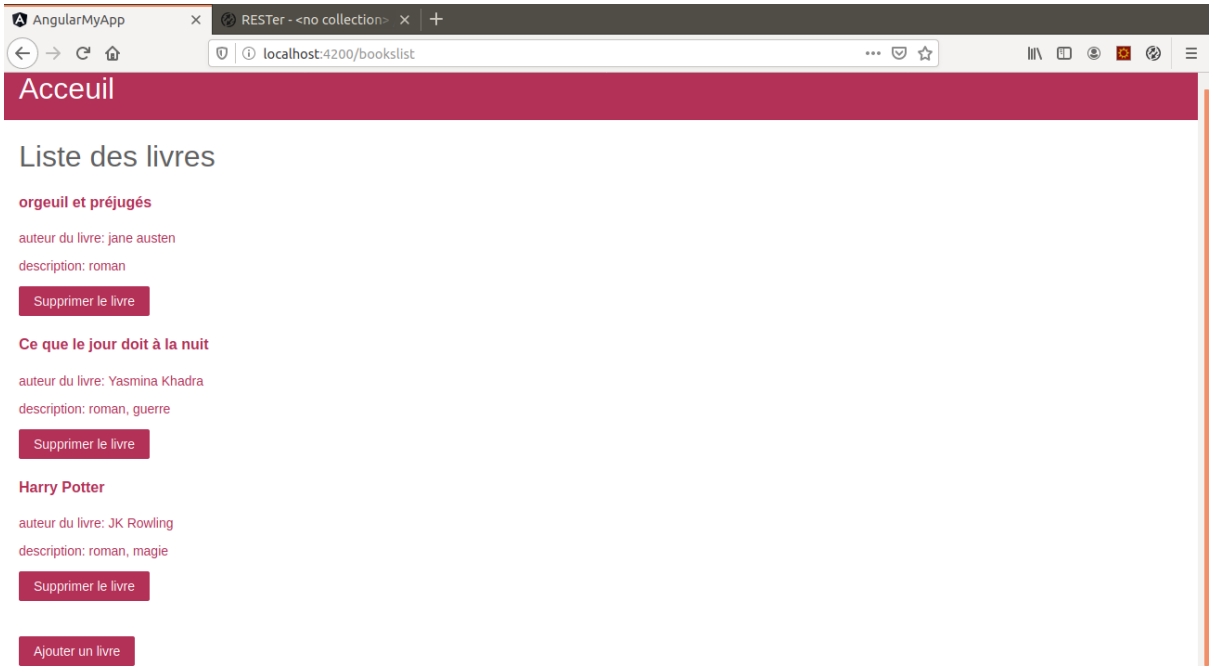


- **Suppression des utilisateurs**

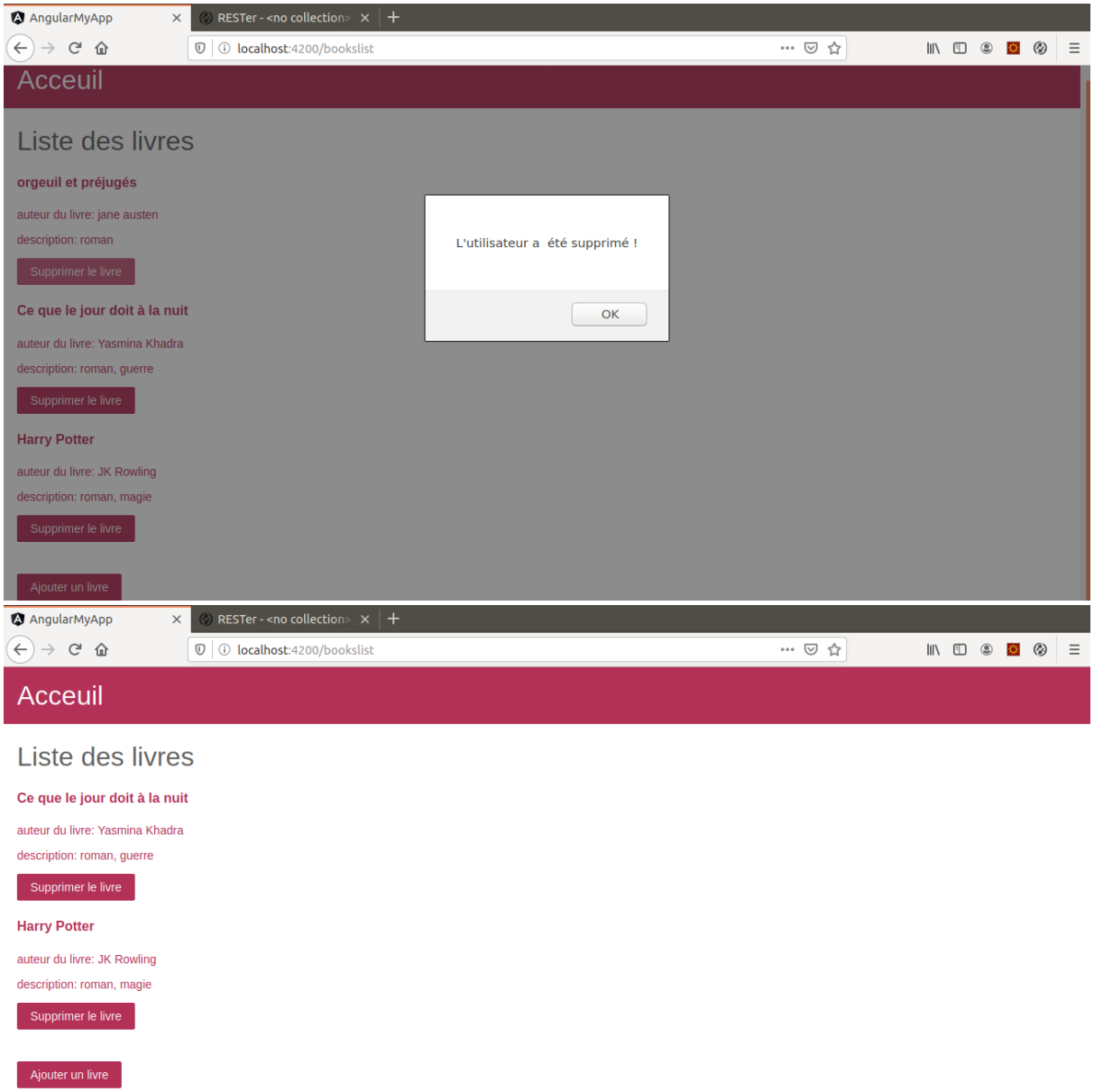




- **Liste des livres**



- **Suppression d'un livre**



6.Construction des images Docker

Une image Docker a été créée pour chaque micro-service cité précédemment. Ces images Docker ont ensuite été déployées sur mon compte Docker Hub (assiabourai).

La communication entre containers Docker se fait bien entre mes micro-services, j'ai pu m'assurer de cela en lançant la commande Docker run avec l'argument --net=host.

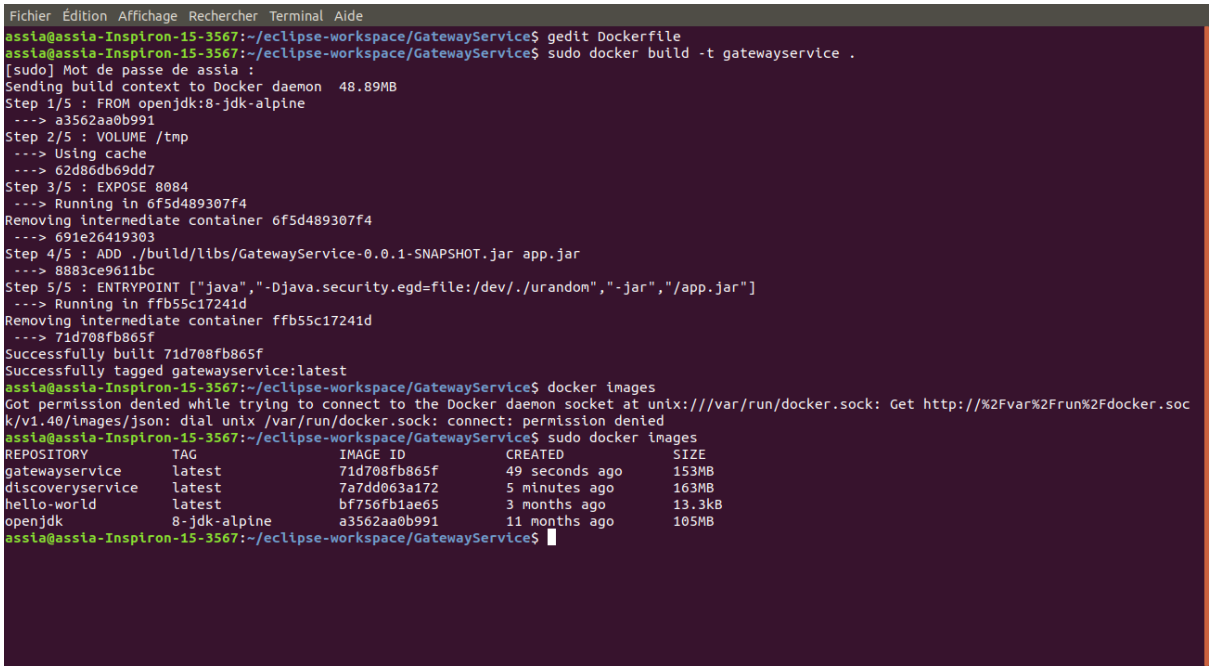
Les figures suivantes montrent l'exécution des commandes docker.

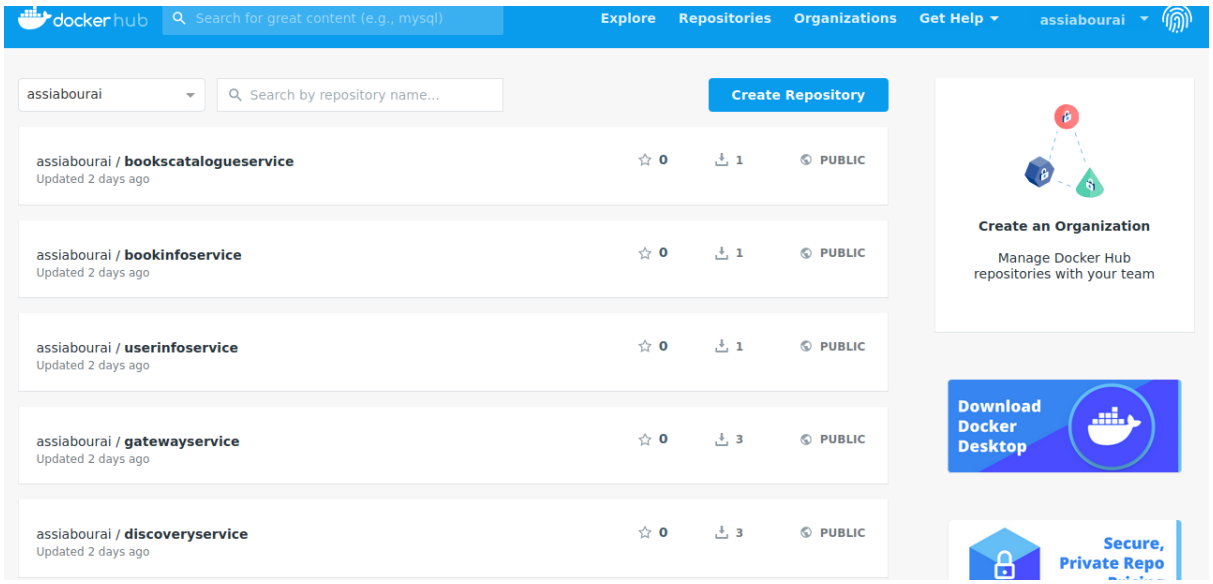
```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
assia@assia-Inspiron-15-3567:~/eclipse-workspace/UserInfoService$ gedit Dockerfile
assia@assia-Inspiron-15-3567:~/eclipse-workspace/UserInfoService$ sudo docker build -t userinfoservice .
[sudo] Mot de passe de assia :
Désolé, essayez de nouveau.
[sudo] Mot de passe de assia :
Sending build context to Docker daemon  63.82MB
Step 1/5 : FROM openjdk:8-jdk-alpine
 ---> a3562aa0b991
Step 2/5 : VOLUME /tmp
 ---> Using cache
 ---> 62d86db69dd7
Step 3/5 : EXPOSE 8081
 ---> Running in 62265ceb0396
Removing intermediate container 62265ceb0396
 ---> ee57e18db8ca
Step 4/5 : ADD ./build/libs/UserInfoService-0.0.1-SNAPSHOT.jar app.jar
 ---> b8cc05b7a685
Step 5/5 : ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
 ---> Running in 6c51754a5556
Removing intermediate container 6c51754a5556
 ---> 6f4f38fa0ae6
Successfully built 6f4f38fa0ae6
Successfully tagged userinfoservice:latest
assia@assia-Inspiron-15-3567:~/eclipse-workspace/UserInfoService$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
userinfoservice     latest              6f4f38fa0ae6        About a minute ago   168MB
gatewayservice      latest              71d708fb865f        4 minutes ago        153MB
discoveryservice    latest              7a7dd063a172        9 minutes ago        163MB
hello-world         latest              bf756fb1ae65        3 months ago         13.3kB
openjdk             8-jdk-alpine        a3562aa0b991        11 months ago        105MB
assia@assia-Inspiron-15-3567:~/eclipse-workspace/UserInfoService$
```

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
assia@assia-Inspiron-15-3567:~/eclipse-workspace/BookInfoService$ gedit Dockerfile
assia@assia-Inspiron-15-3567:~/eclipse-workspace/BookInfoService$ sudo docker build -t bookinfoservice .
[sudo] Mot de passe de assia :
Sending build context to Docker daemon  67.39MB
Step 1/5 : FROM openjdk:8-jdk-alpine
 ---> a3562aa0b991
Step 2/5 : VOLUME /tmp
 ---> Using cache
 ---> 62d86db69dd7
Step 3/5 : EXPOSE 8082
 ---> Running in 8d0a58e5203b
Removing intermediate container 8d0a58e5203b
 ---> 71a69817ebe9
Step 4/5 : ADD ./build/libs/BookInfoService-0.0.1-SNAPSHOT.jar app.jar
 ---> 96c2eb1e1363
Step 5/5 : ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
 ---> Running in ee70de9751ea
Removing intermediate container ee70de9751ea
 ---> 37e67d1e6ee9
Successfully built 37e67d1e6ee9
Successfully tagged bookinfoservice:latest
assia@assia-Inspiron-15-3567:~/eclipse-workspace/BookInfoService$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
bookinfoservice     latest              37e67d1e6ee9        7 minutes ago       168MB
userinfoservice     latest              6f4f38fa0ae6        13 minutes ago      168MB
gatewayservice      latest              71d708fb865f        16 minutes ago      153MB
discoveryservice    latest              7a7dd063a172        21 minutes ago      163MB
hello-world         latest              bf756fb1ae65        3 months ago        13.3kB
openjdk             8-jdk-alpine        a3562aa0b991        11 months ago       105MB
assia@assia-Inspiron-15-3567:~/eclipse-workspace/BookInfoService$
```

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
assia@assia-Inspiron-15-3567:~/eclipse-workspace/BooksCatalogueService$ gedit Dockerfile
assia@assia-Inspiron-15-3567:~/eclipse-workspace/BooksCatalogueService$ sudo docker build -t bookscatalogueservice .
[sudo] Mot de passe de assia :
Sending build context to Docker daemon  62.53MB
Step 1/5 : FROM openjdk:8-jdk-alpine
 ---> a3562aa0b991
Step 2/5 : VOLUME /tmp
 ---> Using cache
 ---> 62d86db69dd7
Step 3/5 : EXPOSE 8083
 ---> Running in ab03c135471e
Removing intermediate container ab03c135471e
 ---> b050cb1be508
Step 4/5 : ADD ./build/libs/BooksCatalogueService-0.0.1-SNAPSHOT.jar app.jar
 ---> b2cb32e98f16
Step 5/5 : ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
 ---> Running in f0f81ba80f63
Removing intermediate container f0f81ba80f63
 ---> a4525efff784
Successfully built a4525efff784
Successfully tagged bookscatalogueservice:latest
assia@assia-Inspiron-15-3567:~/eclipse-workspace/BooksCatalogueService$ sudo docker images
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
bookscatalogueservice   latest              a4525efff784        8 seconds ago       167MB
bookinfoservice         latest              37e67d1e6ee9        9 minutes ago       168MB
userinfoservice         latest              6f4f38fa0ae6        16 minutes ago      168MB
gatewayservice          latest              71d708fb865f        18 minutes ago      153MB
discoveryservice        latest              7a7dd063a172        23 minutes ago      163MB
hello-world             latest              bf756fb1ae65        3 months ago        13.3kB
openjdk                 8-jdk-alpine        a3562aa0b991        11 months ago       105MB
assia@assia-Inspiron-15-3567:~/eclipse-workspace/BooksCatalogueService$
```

```
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
assia@assia-Inspiron-15-3567:~/eclipse-workspace/DiscoveryService$ sudo docker build -t discoveryservice .
[sudo] Mot de passe de assia :
Sending build context to Docker daemon  60.28MB
Step 1/5 : FROM openjdk:8-jdk-alpine
8-jdk-alpine: Pulling from library/openjdk
e7c96db7181b: Pull complete
f910a506b6cb: Pull complete
c2274a1a0e27: Pull complete
Digest: sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a0f2283343547b3
Status: Downloaded newer image for openjdk:8-jdk-alpine
 ---> a3562aa0b991
Step 2/5 : VOLUME /tmp
 ---> Running in b4e5d278180d
Removing intermediate container b4e5d278180d
 ---> 62d86db69dd7
Step 3/5 : EXPOSE 8761
 ---> Running in 49a936ecd126
Removing intermediate container 49a936ecd126
 ---> 40cfabd88905
Step 4/5 : ADD ./build/libs/DiscoveryService-0.0.1-SNAPSHOT.jar app.jar
 ---> bf336e23b85b
Step 5/5 : ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
 ---> Running in dcca4ab5f2ab
Removing intermediate container dcca4ab5f2ab
 ---> 7a7dd063a172
Successfully built 7a7dd063a172
Successfully tagged discoveryservice:latest
assia@assia-Inspiron-15-3567:~/eclipse-workspace/DiscoveryService$
```





Conclusion

Ce projet a été pour moi une opportunité d'apprendre d'avantage sur le développement web et ops, il m'a permis d'acquérir une certaine autonomie dans mon travail, ainsi, j'ai pu me forger une certaine rigueur de travail qui me permettra d'évoluer plus rapidement dans les projets à venir

Le code source du projet ainsi que le rapport en PDF ont été envoyés par mail à votre adresse et déployés sur mon compte github à l'adresse suivante:

https://github.com/assia91/Projet-Prog-Web-Dist