

Rapport du projet Architecture Microservices

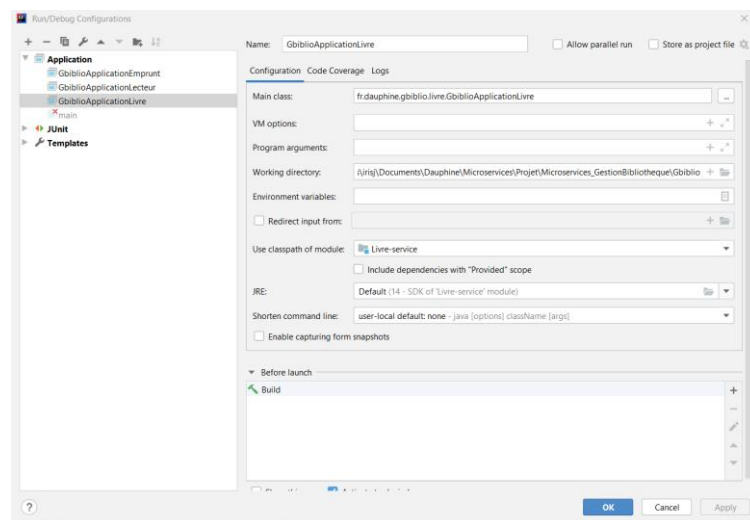
HADI Ismaïl & JOLIMAN Iris

I. Indications pour la compilation et l'utilisation

Nous avons fourni une liste des prérequis et une description globale des microservices dans le README.MD sur notre dépôt Git.

a) Compilation avec un IDE

1. Importer le projet avec Maeven et Version control grâce à l'Url suivant : https://github.com/irisjoliman/Microservices_GestionBibliotheque.git
2. Builder le projet
3. Editer les configurations de Run/Debug pour les 3 microservices.



4. Cliquer sur Run pour chacune des configurations des microservices. Bien qu'indépendants, si l'on souhaite pouvoir les utiliser en parallèle, cette étape est indispensable.
5. Ouvrez l'interface utilisateur en cliquant sur : **Front_WebService.jar** dans le dossier FrontEnd

L'application est désormais lancée et fonctionnelle si vous obtenez les indications suivantes pour les trois microservices :

```
2020-05-31 23:49:38.040 INFO 24924 --- [ restartedMain] f.d.g.livre.GbbiblioApplicationLivre : Starting GbbiblioApplicationLivre on DESKTOP-02T1G14 with P
2020-05-31 23:49:38.045 INFO 24924 --- [ restartedMain] f.d.g.livre.GbbiblioApplicationLivre : No active profile set, falling back to default profiles: d
2020-05-31 23:49:38.243 INFO 24924 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.ad
2020-05-31 23:49:38.243 INFO 24924 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'l
2020-05-31 23:49:38.045 INFO 24924 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode
2020-05-31 23:49:38.945 INFO 24924 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 865ms. Found 1
2020-05-31 23:49:37.228 INFO 24924 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-05-31 23:49:37.284 INFO 24924 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-05-31 23:49:37.284 INFO 24924 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.34]
2020-05-31 23:49:37.682 INFO 24924 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-05-31 23:49:37.683 INFO 24924 --- [ restartedMain] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 74
2020-05-31 23:49:38.032 INFO 24924 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2020-05-31 23:49:38.734 INFO 24924 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2020-05-31 23:49:38.754 INFO 24924 --- [ restartedMain] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available
2020-05-31 23:49:39.421 INFO 24924 --- [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HH0000284: Processing PersistenceUnitInfo [name: default]
2020-05-31 23:49:39.774 INFO 24924 --- [ restartedMain] org.hibernate.Version : HH0000412: Hibernate ORM core version 5.4.15.Final
2020-05-31 23:49:40.368 INFO 24924 --- [ restartedMain] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
2020-05-31 23:49:40.812 INFO 24924 --- [ restartedMain] org.hibernate.dialect.Dialect : HH0000400: Using dialect: org.hibernate.dialect.H2Dialect
Hibernate: drop table if exists livre CASCADE
2020-05-31 23:49:44.103 INFO 24924 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HH0000490: Using JtaPlatform implementation: [org.hibernate
2020-05-31 23:49:44.118 INFO 24924 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit
2020-05-31 23:49:44.148 INFO 24924 --- [ restartedMain] o.s.b.a.o.OptionalLiveReloadServer : LiveReload server is running on port 35729
2020-05-31 23:49:44.427 WARN 24924 --- [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore,
2020-05-31 23:49:44.867 INFO 24924 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-05-31 23:49:47.922 INFO 24924 --- [ restartedMain] o.s.b.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath base path '/actuator'
2020-05-31 23:49:48.107 INFO 24924 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path '/'
2020-05-31 23:49:48.113 INFO 24924 --- [ restartedMain] f.d.g.livre.GbbiblioApplicationLivre : Started GbbiblioApplicationLivre in 20.222 seconds (JVM run
```

a) Compilation en ligne de commande avec Docker

1. Checkout du code avec git clone
https://github.com/irisjoliman/Microservices_GestionBibliotheque.git
2. En ligne de commande à la racine de chaque microservice : `mvn clean install package`
3. Ouverture de Docker Quick Start Terminal et à la racine de chaque microservice : `docker build -t livre-service .`


Livre est à titre d'exemple et à remplacer par emprunt ou lecteur.

4. Toujours dans le docker Quick Start Terminal Docker run -d -p 9090 :8000 livre-service

```
Start interactive shell

irisj@DESKTOP-02T1G14 MINGW64 /c/Program Files/Docker Toolbox
$ docker-machine ip default
192.168.99.100

irisj@DESKTOP-02T1G14 MINGW64 /c/Program Files/Docker Toolbox
$ docker run -p 9090:8000 livre-service
```



```
:: Spring Boot ::                (v2.2.7.RELEASE)

2020-05-27 15:40:13.415 INFO 1 --- [           main] f.d.g.livre.GbiblioApplicationLivre : Start
2020-05-27 15:40:13.431 INFO 1 --- [           main] f.d.g.livre.GbiblioApplicationLivre : No ac
2020-05-27 15:40:22.159 INFO 1 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Boots
```

L'application est fonctionnelle et les microservices tournent en parallèle

Liste des ports :

- Microservice Livre 9090 : 8000
- Microservice Lecteur 9091 : 8001
- Microservice Emprunt 9092 : 8002

La liste des fonctions disponibles est également indiquée dans le README.MD

Vous pouvez désormais ouvrir l'application front end, faire le choix de fonctionnalité et rentrer les requêtes dans POSTMAN ou simplement accéder aux requêtes dans le fichier indiqué précédemment.

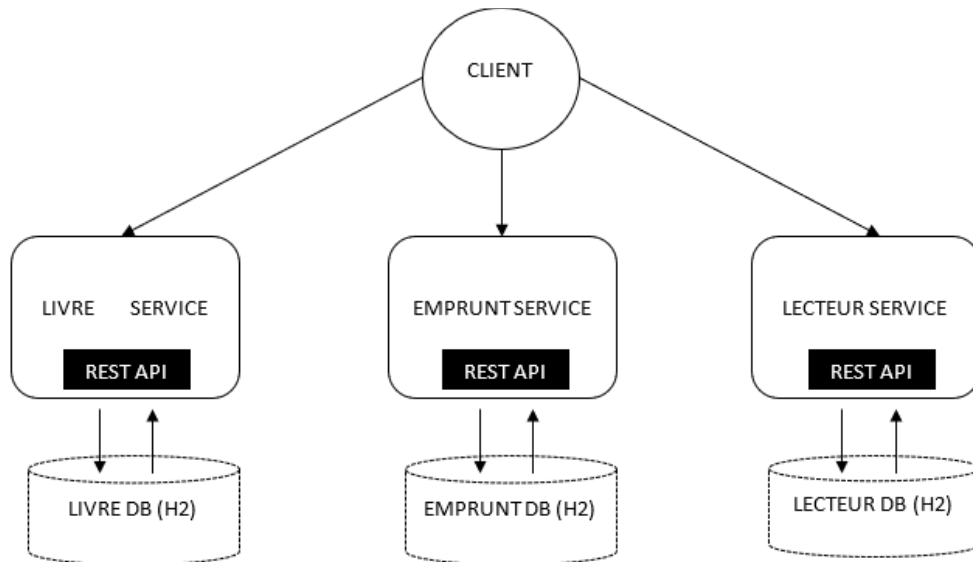
II. Documentation technique

a) *Choix d'architecture*

Nous nous sommes inspirés de nos recherches, notamment celles effectuées sur GitHub pour choisir la structure de notre projet. Nous avons dû nous éloigner de nos habitudes concernant les architectures monolithiques. Finalement, nous avons compris l'intérêt de l'indépendance des microservices et avons découpé les spécifications en trois parties afin de réaliser trois microservices complètement indépendants.

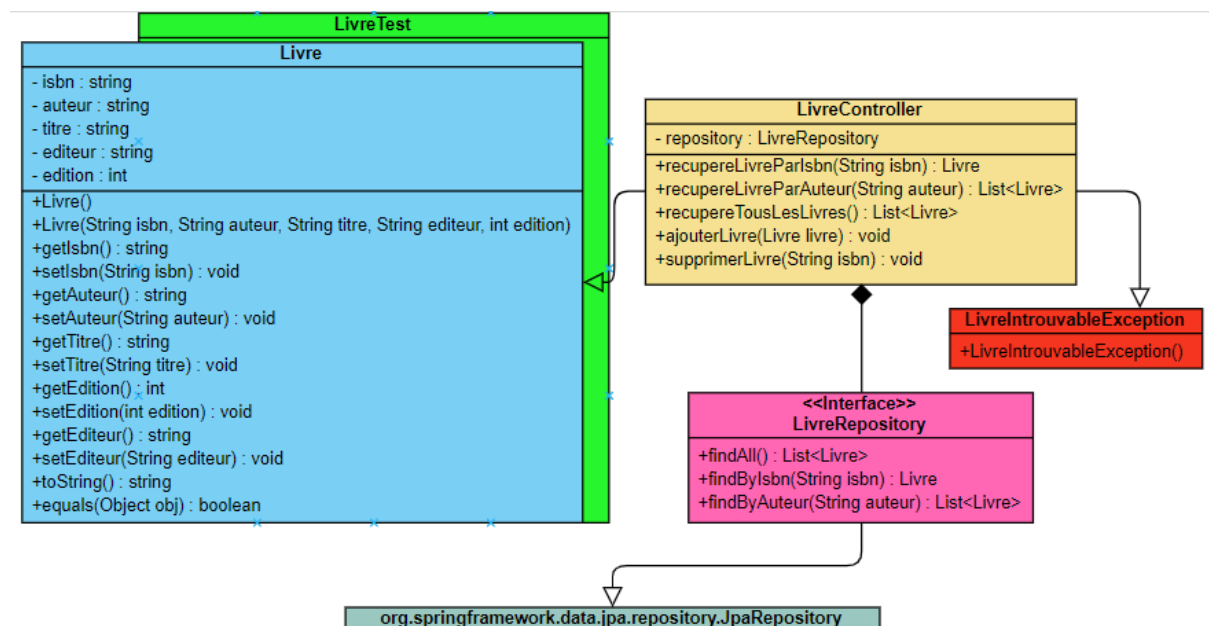
Ces microservices ne communiquent pas entre eux. Nous avons étudié cet aspect de communication (synchrone & asynchrone) mais cela complexifierait le projet et sortait selon notre point de vue des spécifications de ce dernier. C'est pourquoi nous ne gérons pas la fiabilité des données et partons du principe que l'utilisateur ne rentre de données incohérentes.

Voici notre schéma d'architecture :

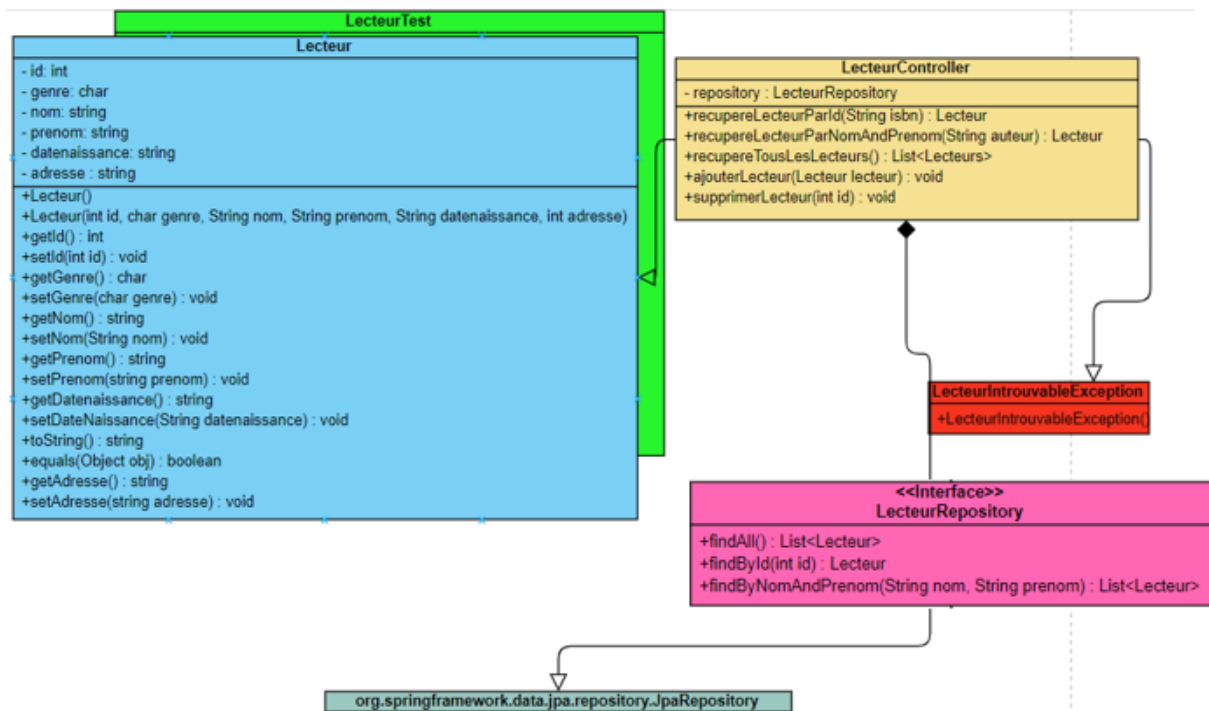


L'interface utilisateur a été développée en Java Swing et permet d'indiquer à l'utilisateur les requêtes http à effectuer dans Postman (notamment pour les POST et DELETE). Pour les GET, elle ouvre automatiquement le navigateur.

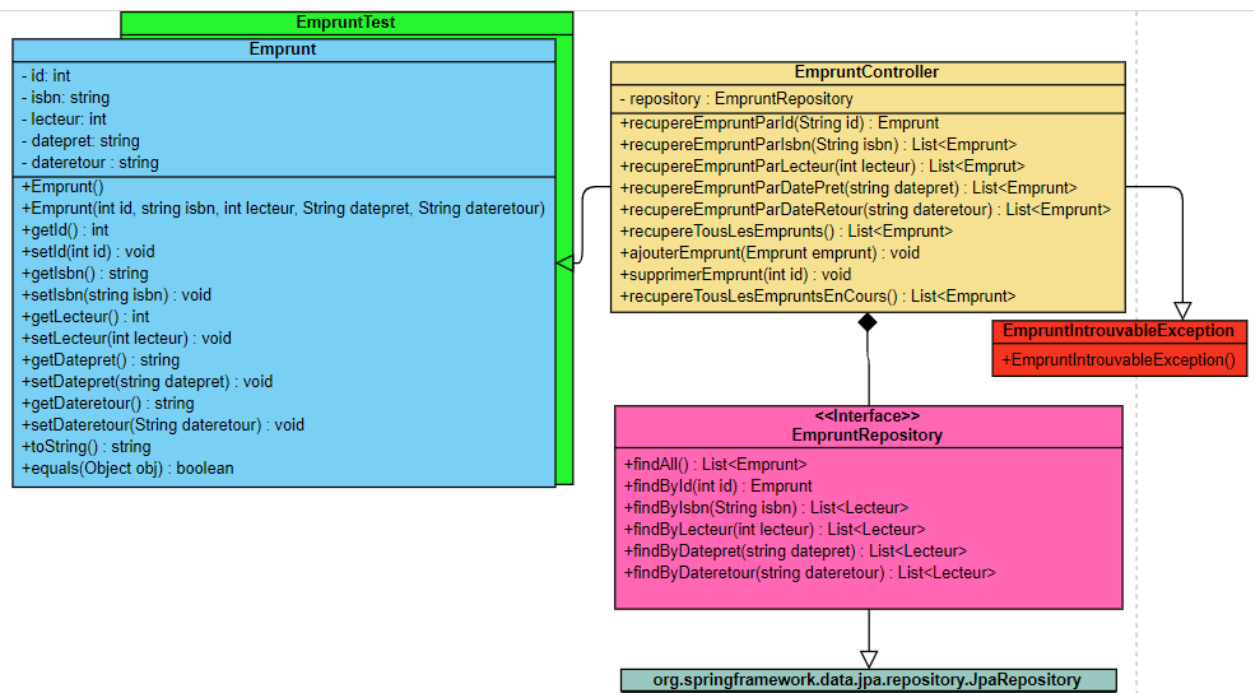
b) Diagrammes de classes



1 Diagramme de classe du microservice Livres



2 Diagramme de classe du microservice Lecteur



3 Diagramme du microservice Emprunt

III. Bilan du projet

Ce projet s'est avéré être intéressant pour mieux comprendre le cours théorique. En effet, en ayant le cours théorique, plus la virtualisation de ce dernier, nous avons eu dans un premier temps un peu de mal à voir le concept de l'architecture microservice. Dans le contexte professionnel (stage ou alternance), nous avons tous les deux rencontrés les microservices mais pas dans le détail comme nous avons pu les voir grâce à ce projet.

De plus, il y a eu un vrai travail de recherche tout au long du projet où nous avons pu découvrir d'autres technologies (Spring Boot), d'autres erreurs, et d'autres concepts (comme la Conteneurisation). Nous avons pu découvrir les facilités d'implémentation grâce au Spring Framework. Nous avons eu une nouvelle manière de voir désormais l'architecture de nos applications et cela a été également bénéfique pour nos tâches quotidiennes en alternance.

En revanche nous avons beaucoup moins apprécié l'utilisation de Docker et Minikube. Pour notre part, cela s'est révélé être une perte de temps énorme parce que même si nous avons réussi à installer et utiliser Docker, nous n'avons pas pu appréhender l'utilité de ce dernier. De plus, nous avons vraiment eu du mal à utiliser Minikube, même en ayant fait toutes les requêtes des tutoriels correctement, nous étions confrontés à des erreurs que nous ne comprenions pas et nous avons vraiment subi cette partie du projet. A force de tenter sans résultat, cela nous a quelque peu démotivés. Nous avons plus suivi des tutoriels qu'expérimenté les bienfaits de ces deux technologies et nous trouvons cela regrettable. Le fait de travailler sur des ordinateurs Windows n'ont pas vraiment arrangé cet aspect d'autre part.