**ASI – Atelier 1**

**Table des matières**

[**I-** **Diagramme de classe** 2](#_Toc119860514)

[**II-** **Architecture du projet** 3](#_Toc119860515)

[2.1- BackEnd avant réalisation 3](#_Toc119860516)

[2.2- BackEnd après réalisation 4](#_Toc119860517)

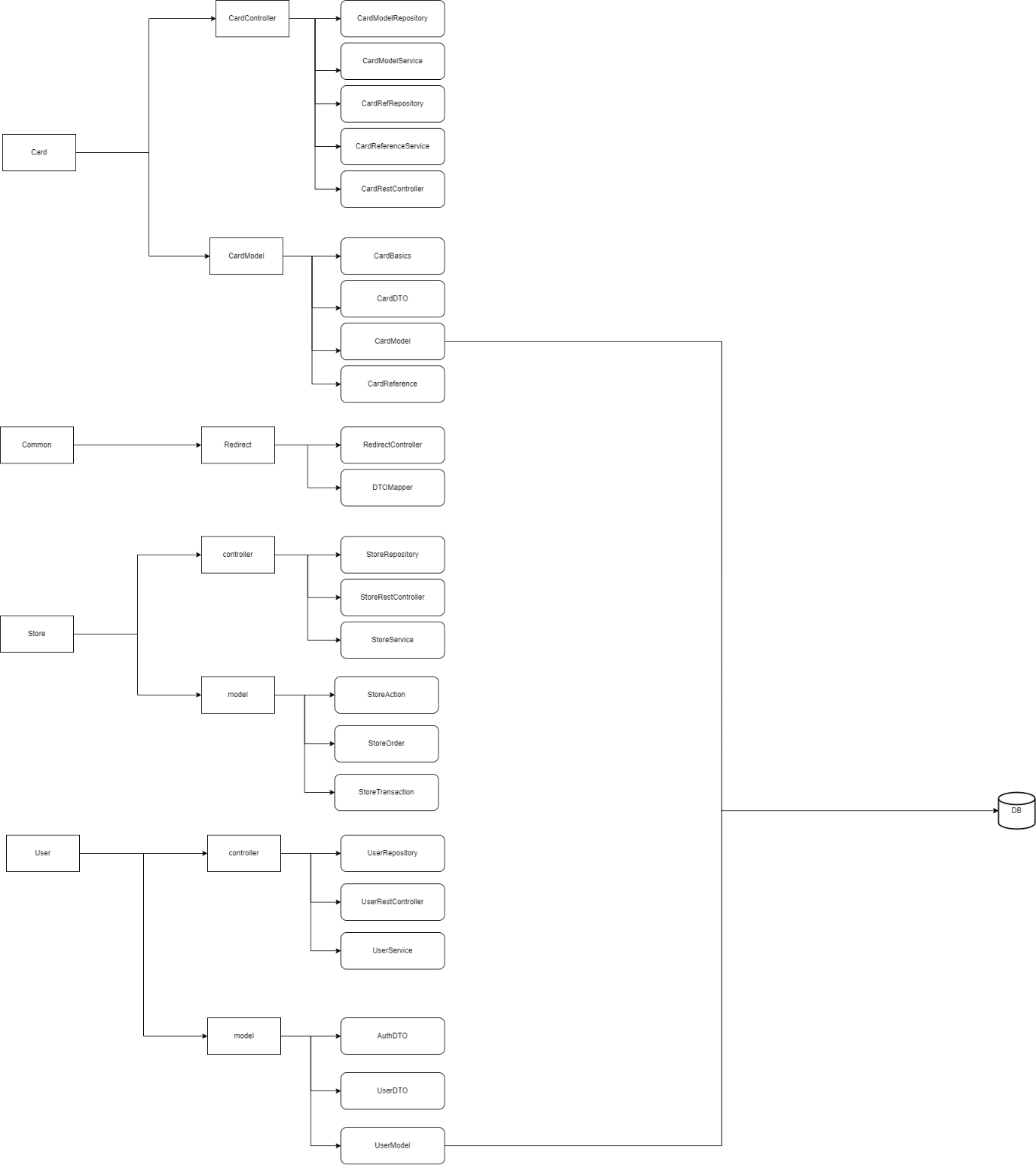
[2.3- FrontEnd 5](#_Toc119860518)

[**III-** **Tableau récapitulatif des bus de communications** 6](#_Toc119860519)

[**IV-** **Tableau récapitulatif des Framework Front-End** 6](#_Toc119860520)

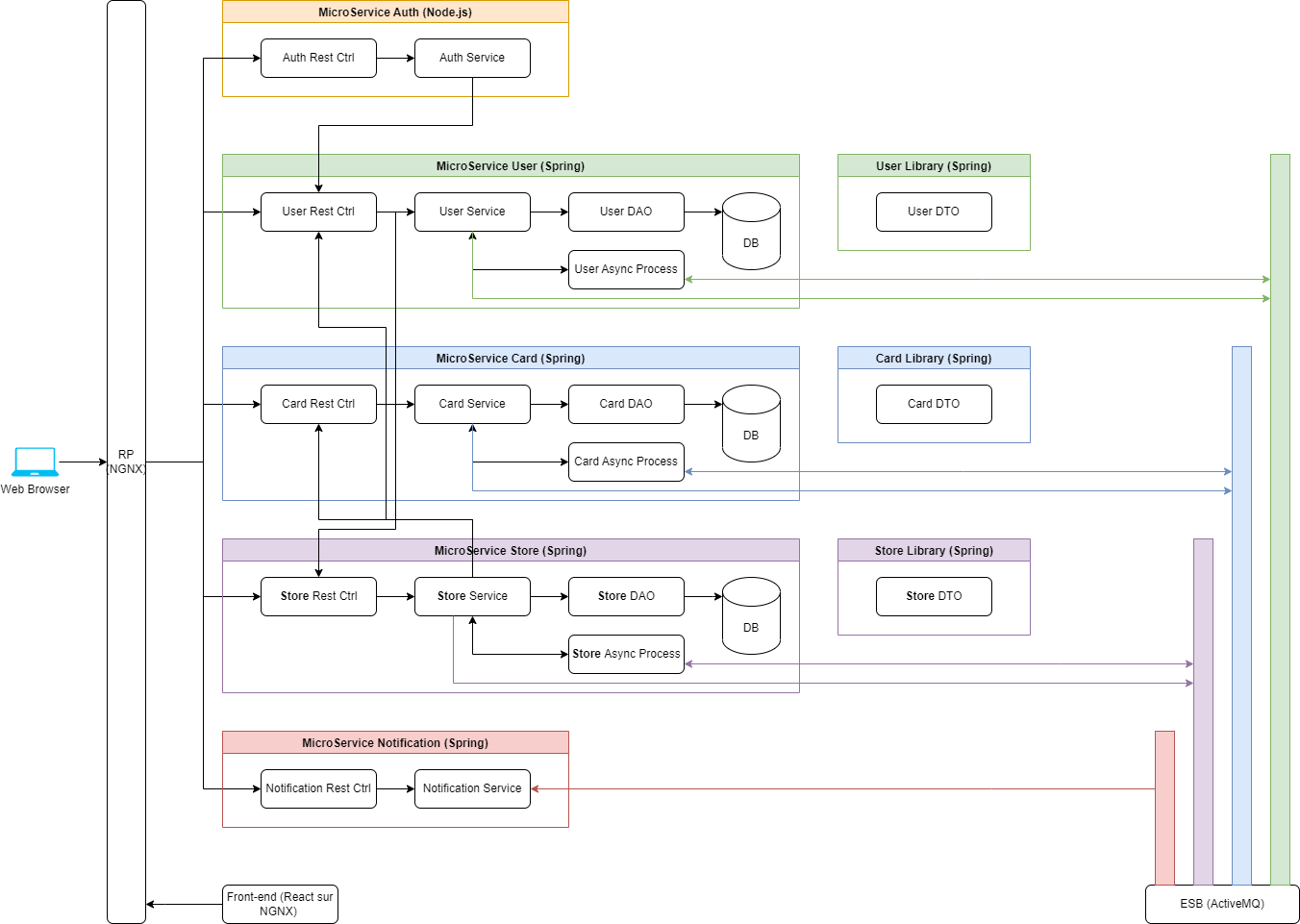
[**V-** **Questions** 7](#_Toc119860521)

# **Diagramme de classe**

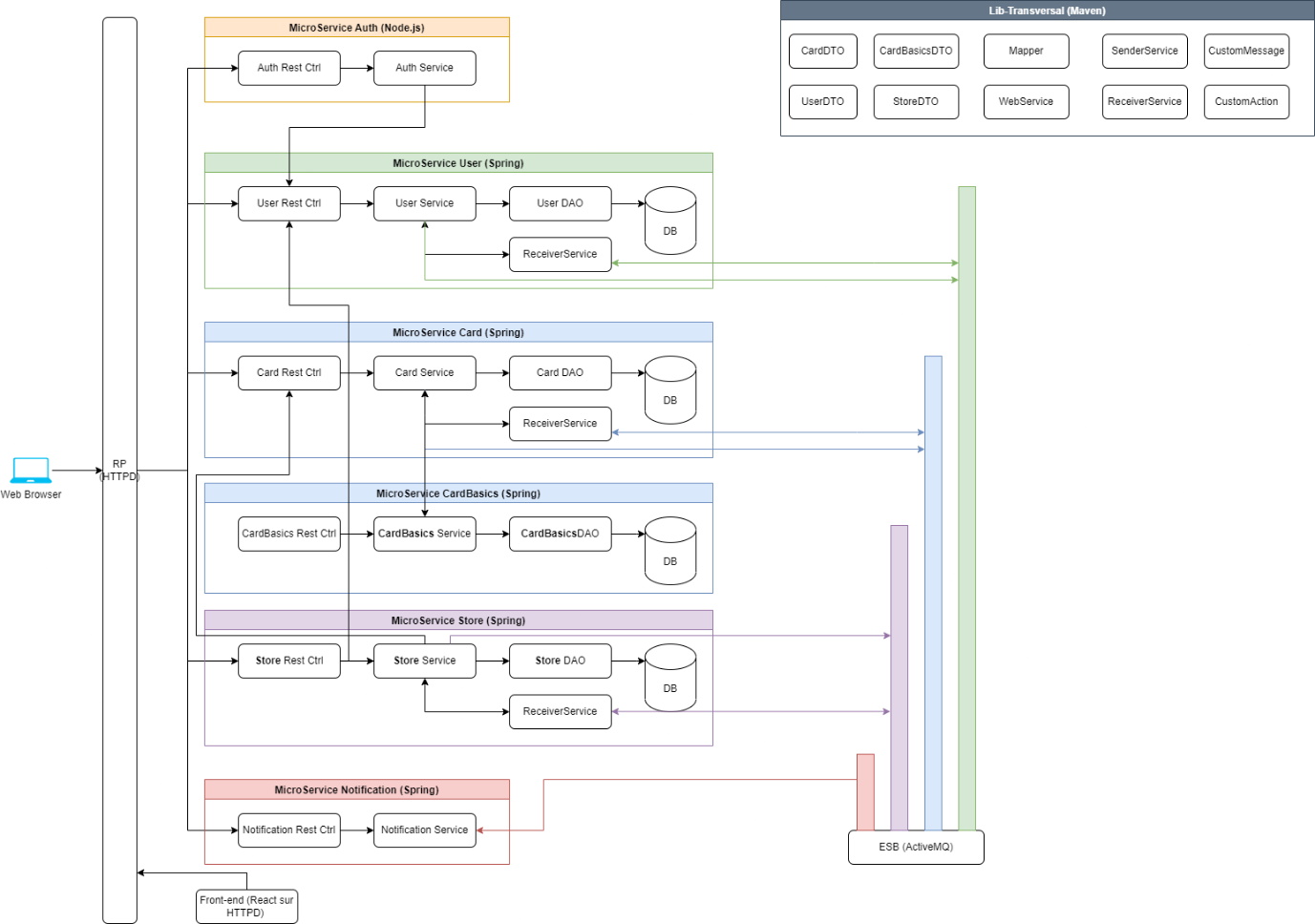


# **Architecture du projet**

## 2.1- BackEnd avant réalisation



## 2.2- BackEnd après réalisation



Pendant la réalisation de l’atelier 1, nous avons dû effectuer des changements au niveau de l’architecture.

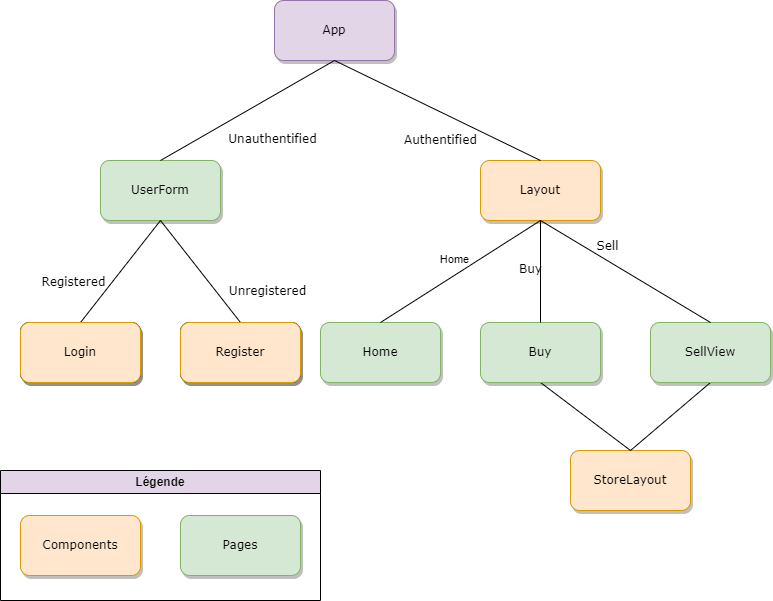
En effet, nous avons décider de mettre dans une librairie Maven :

* L’ensemble des DTO
* Mapper : Une classe générique permettant de mapper les DTO en Entity et inversement.
* WebServices : Une classe générique permettant de réaliser des appels GET, POST et PUT
* Les services permettant le traitement d’un message asynchrone
* CustomMessage : L’objet générique qui sera envoyé dans les différentes Queue

Bien que la mise en place des classes génériques fût décidée très tôt, ces modifications ont été réalisées car notre compréhension sur la partie ESB s’est affinée au fur et à mesure.

## 2.3- FrontEnd

Ce schéma indique tous les modules ReactJS qui ont été créer.



# **Tableau récapitulatif des bus de communications**

Source :

* <https://www.educba.com/activemq-vs-rabbitmq/>
* <https://www.educba.com/activemq-vs-kafka/>

# **Tableau récapitulatif des Framework Front-End**

# **Questions**

Qu’est ce que le CROSS ORIGIN ? En quoi cela est-il dangereux ?

- Le partage des ressources d'origine croisée (CORS) est un mécanisme basé sur l'en-tête HTTP qui permet à un serveur d'indiquer toute origine (domaine, schéma ou port) autre que la sienne à partir de laquelle un navigateur doit autoriser le chargement de ressources. CORS repose également sur un mécanisme par lequel les navigateurs effectuent une requête "preflight" au serveur hébergeant la ressource d'origine croisée, afin de vérifier que le serveur autorisera la requête réelle. Lors de ce contrôle préalable, le navigateur envoie des en-têtes qui indiquent la méthode HTTP et les en-têtes qui seront utilisés dans la demande réelle.

Comment REACTJS fait-il pour afficher rapidement les modifications sur les composants ?

- Les composants ReactJS fonctionne grâce au JSX, c’est une version de l’HTML qui est directement écrite à l’intérieur du composant et qui permets d’implémenter de la logique.En JSX, on peux directement dire à un composant de lancer une fonction si on clique dessus, tout est simplifié.

- https://leblogducodeur.fr/react-js-les-composants/

Quelle est la fonction essentielle de REACTJS ?

- Affichage conditionnel

Quelle est la fonction essentielle de FLUX ?

- Flux est une architecture d'application mais n'est ni une bibliothèque ni un framework.

- Elle est utilisée en interne par Facebook pour construire des applications web côté client avec React. Les applications Flux ont trois rôles principaux lorsqu'elles traitent des données : Dispatcher, Stores et Views (composants React).

- https://www.w3schools.blog/flux-reactjs

Qu’est ce que REDUX ?

- FLUX is architecture, and REDUX is a library.

- Redux est une bibliothèque JavaScript open-source permettant de gérer l'état des applications. Elle est le plus souvent utilisée avec des bibliothèques telles que React ou Angular pour construire des interfaces utilisateur. Redux permet à ses utilisateurs d'écrire des applications qui peuvent fonctionner dans un environnement différent (qu'il s'agisse d'un client, d'un serveur ou d'une application native), d'avoir un comportement cohérent et d'effectuer des tests orientés vers l'est. En dehors de cela, il offre une expérience de développement étonnante, comme l'édition en direct du code avec un débogueur qui voyage dans le temps.

- https://www.educba.com/redux-vs-flux/

Qu’est ce que JMS ? Est-ce spécifique à Springboot ?

- Java Message Service

- API qui prend en charge la communication formelle appelée messagerie entre ordinateurs sur un réseau. JMS fournit une interface commune pour les protocoles de messages standard et les services de messages en soutien aux programmes Java.

- Non c'est du Java

Quelles sont les différents modes de transmissions de JMS ?

- Point-to-point

- Publish-and-subscribe

Qu’est ce que ActiveMQ ?

- Apache ActiveMQ est un broker open source écrit en Java avec un client Java Message Service (JMS) complet. Il offre des "fonctionnalités d'entreprise", ce qui signifie dans ce cas favoriser la communication à partir de plus d'un client ou serveur. Les clients pris en charge incluent Java via JMS 1.1 ainsi que plusieurs autres clients "cross language". La communication est gérée par des fonctionnalités telles que la mise en grappe d'ordinateurs et la possibilité d'utiliser n'importe quelle base de données comme fournisseur de persistance JMS en plus de la mémoire virtuelle, du cache et de la persistance du jour

- https://en.wikipedia.org/wiki/Apache\_ActiveMQ

Quel est le mode de transmission activé par défaut dans activeMq ?

-

Quels avantages proposent un bus de communication vis-à-vis de requêtes http classiques ?

- Facilite la communication en normalisant les données dans un même format

- Garantit la sécurité

Comment faire pour partager des modules Maven avec un partenaire extérieur ?

- Faire un maven plugin / jar

Comment faire pour exporter un composant REACTJS ?

- "export default App"

- "export {Greet as Greeting}"

Quel est le pré-requis pour Springboot afin de pouvoir convertir automatiquement le message reçu dans un bus de communication en objet ?

- Il faut injecter la dépendance jmsTemplate

- "jmsTemplate.convertAndSend("RESULT\_BUS\_MNG",msg);"

Comment est réalisée la traduction des messages reçus (bus de communication ou request http) en objet Java ? Quelles sont les prérequis ? Est-ce toujours possible ?

- On utilise un DTO : "@RequestBody @Validated UserDTO userDto"

Quelles sont les fonctionnalités des différentes annotations en Springboot ?:

- @EnableJpaRepositories

- Permet d'activer les JPA Repositories qui contiennent les API pour les opérations CRUD de base, les API pour la pagination et les API pour le tri.

- https://www.simplilearn.com/tutorials/jpa-tutorial/spring-boot-jpa

- @EntityScan

- Configure les paquets de base utilisés par l'autoconfiguration lors de la recherche de classes d'entités.

L'utilisation de @EntityScan entraînera l'auto-configuration à :

- Définit les paquets analysés pour les entités JPA.

- Définir le jeu d'entités initial utilisé avec les contextes de mapping Spring Data MongoDB, Neo4j, Cassandra et Couchbase.

- https://docs.spring.io/spring-boot/docs/current/api/org/springframework/boot/autoconfigure/domain/EntityScan.html

- @ComponentScan

- Indique à Spring d'analyser le paquet actuel et tous ses sous-paquets

- https://www.geeksforgeeks.org/spring-componentscan-annotation-with-example/