

**Rapport projet microservices**

**Réalisée par: LAGRIOUID KAWTAR| 2021/2022**

# **Enoncé**

Réaliser une application basée sur les micro-services en utilisant Spring Boot et Spring Cloud composée des services suivants :

* Micro-service Customer-service
* Micro-service Inventory-Service
* Micro-service Billing-service
* Gateway service
* Eureka Discovery service
* Sécurité avec Keycloak
* Partie KAFKA
* Partie Angular avec NGRX

**Architecture du project:**

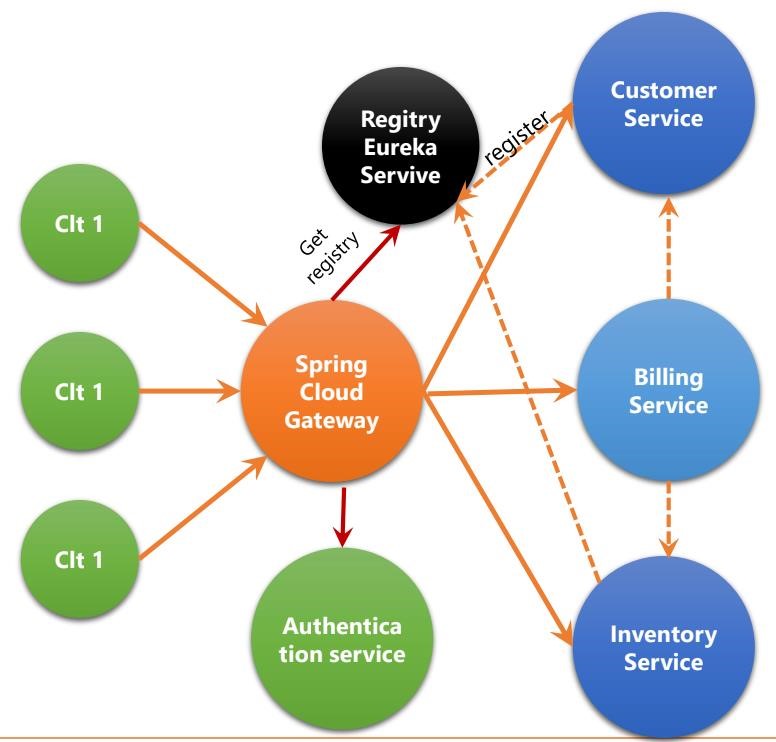
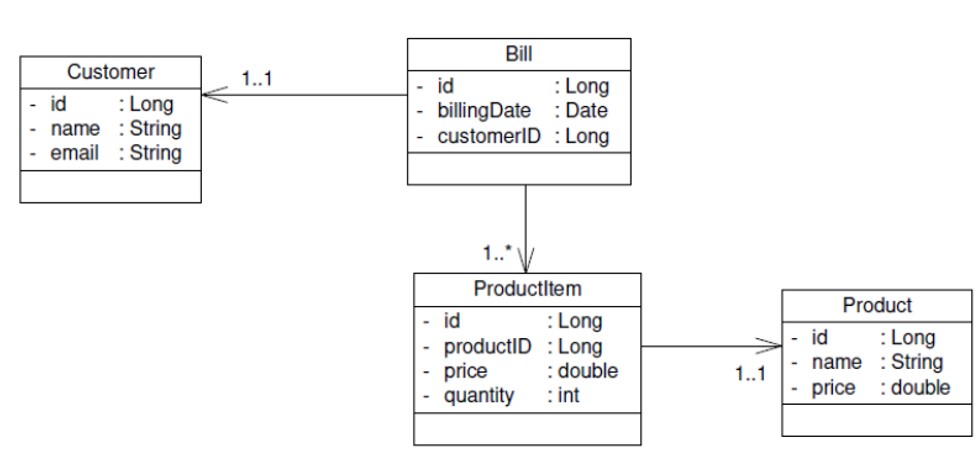
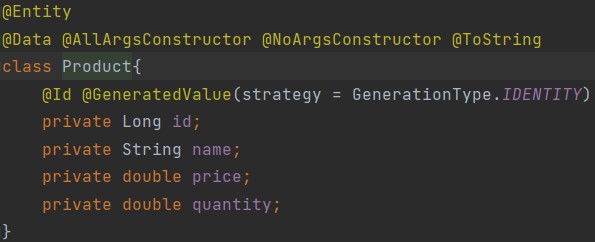


Diagramme de classe :



Inventory service:

ENTITY:

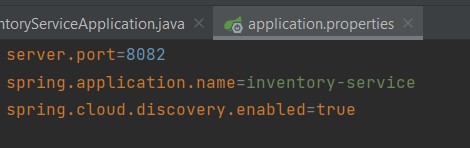


INVENTORYSERVICEAPPLICATION :

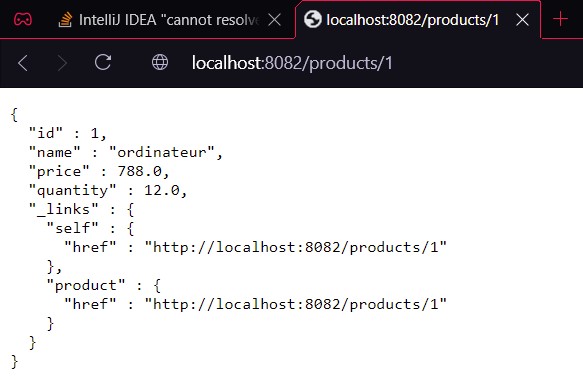
On crée quelques enregistrements dans notre base H2 grâce à un RestRepository.



CONFIGURATION DU MICROSERVICE :

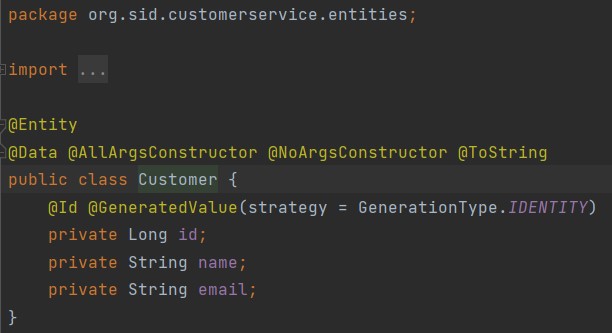


TEST DU MICROSERVICE :



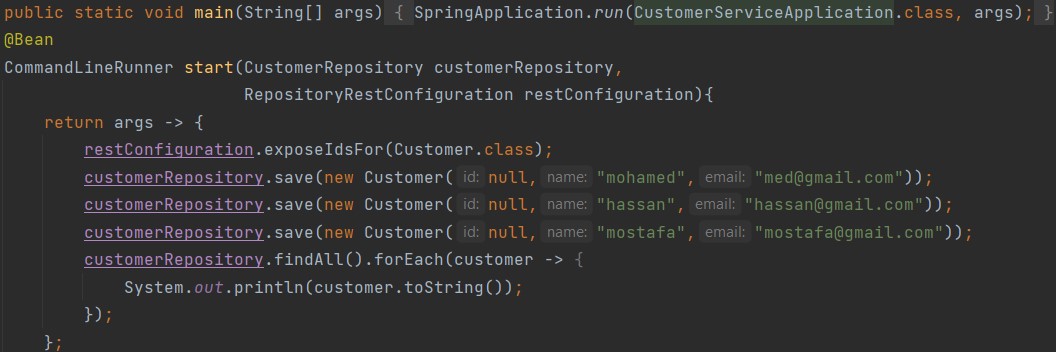
Customer service:

ENTITY:



INVENTORYSERVICEAPPLICATION :

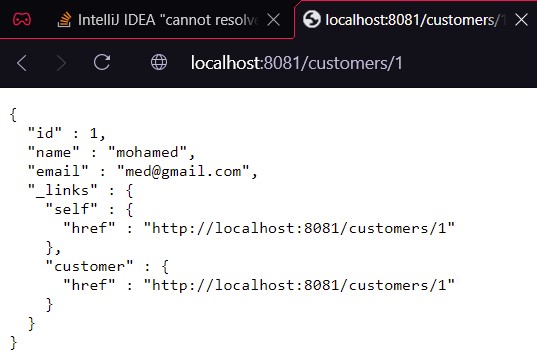
On crée quelques enregistrements dans notre base H2 grâce à un RestRepository.



CONFIGURATION DU MICROSERVICE :

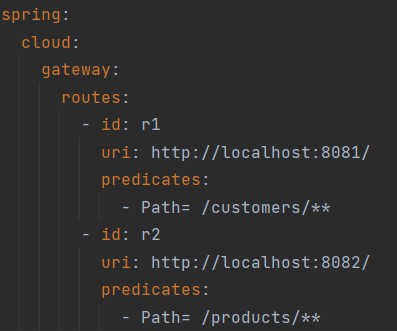


TEST DU MICROSERVICE :

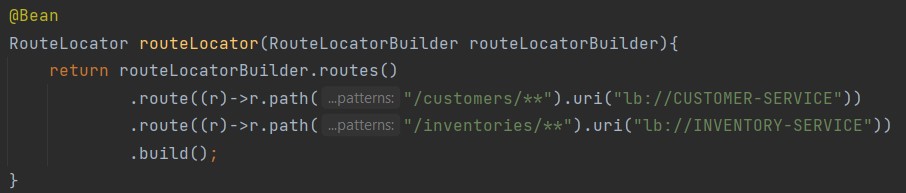


Gateway microservice (Spring Cloud Gateway):

CONFIGURATION STATIQUE (APPLICATION.YML):



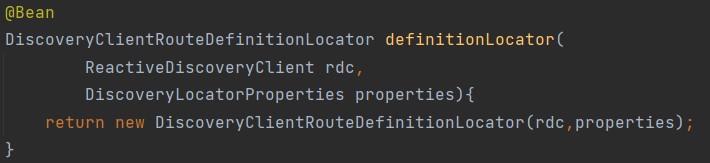
CONFIGURATION DYNAMIQUE 1 (FICHIER JAVA):



On utilise cette configuration quand on connaît le nom des microservices.

Lb= load balancer, pour la répartition d’un ensemble de tâches sur un ensemble de ressources, dans le but d’en rendre le traitement global plus efficace.

CONFIGURATION DYNAMIQUE 2 (FICHIER JAVA) :

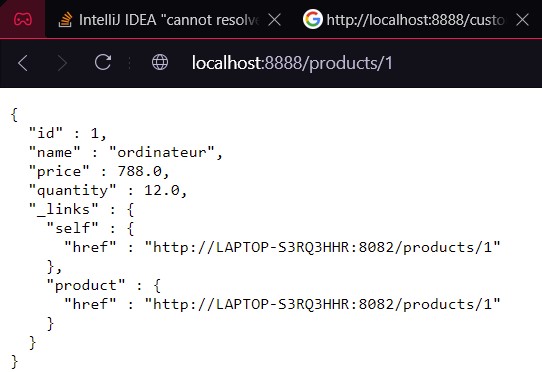


On utilise cette configuration on avec Eureka Discovery quand on ne connaît pas le nom des microservices.

CONFIGURATION DE LA GATEWAY :



TEST DE LA GATEWAY :



Eureka-Discovery :

CONFIGURATION DU MICROSERVICE :

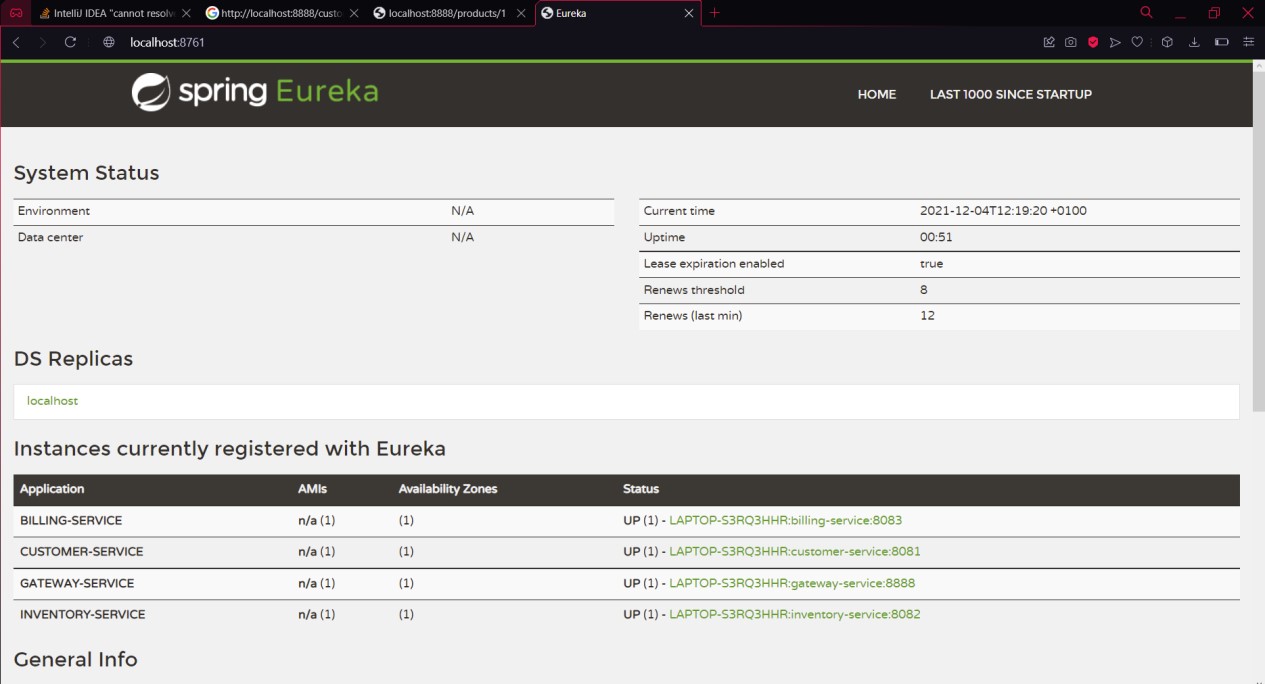


Ligne 2 : pour dire à Eureka de ne pas exposer ses services dans l’annuaire.

Ligne 3 : pour dire à Eureka de ne pas s’autoenregistrer dans l’annuaire.

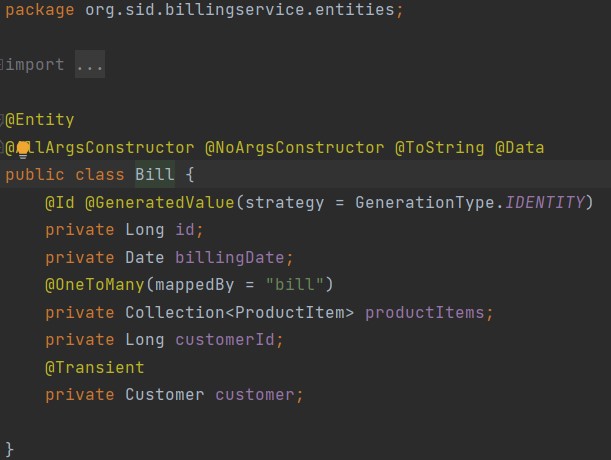
Il faut également ajouter  dans EurekaDiscoveryApplication.

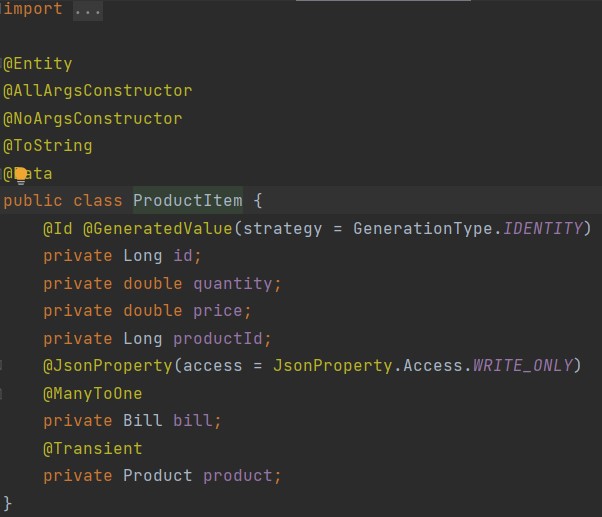
TEST DU MICROSERVICE :



Billing service:

ENTITY:

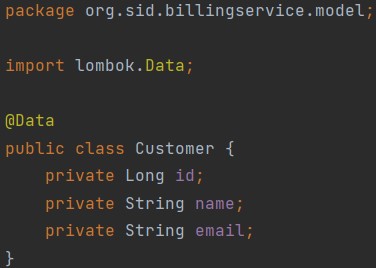
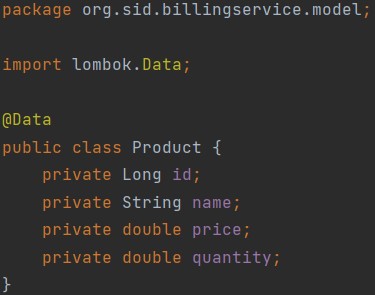




L’annotation @Transient sert à indiquer que le champ ne doit pas être persisté dans la base de données.

@JsonProperty sert a sortir d’une boucle infini lors de la lecture d’un productItem.

MODEL DE DONNEES:



OPENFEIGN:



Permet de faire des requettes http entre les microservices grâce a un Rest Client.

BILLINGRESTCONTROLLER :



CONFIGURATION DU MICROSERVICE :

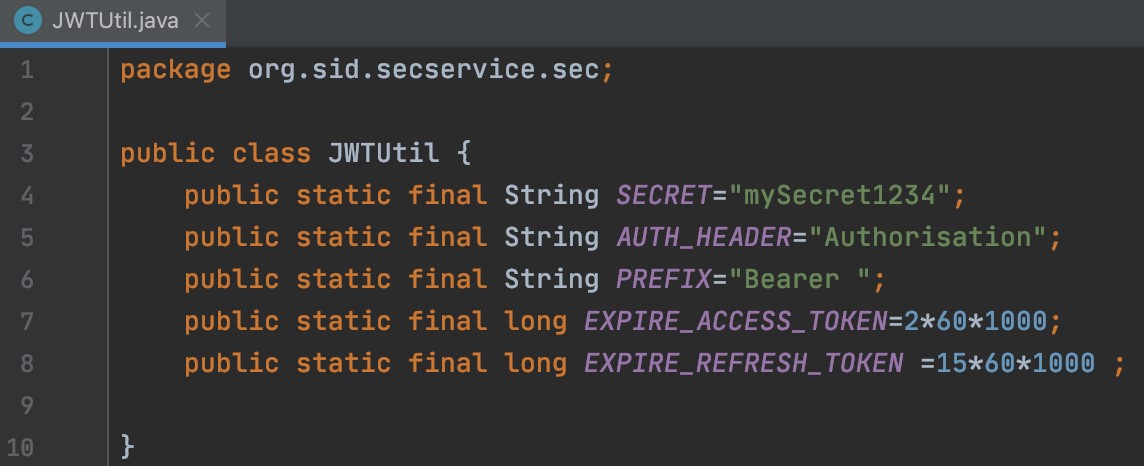


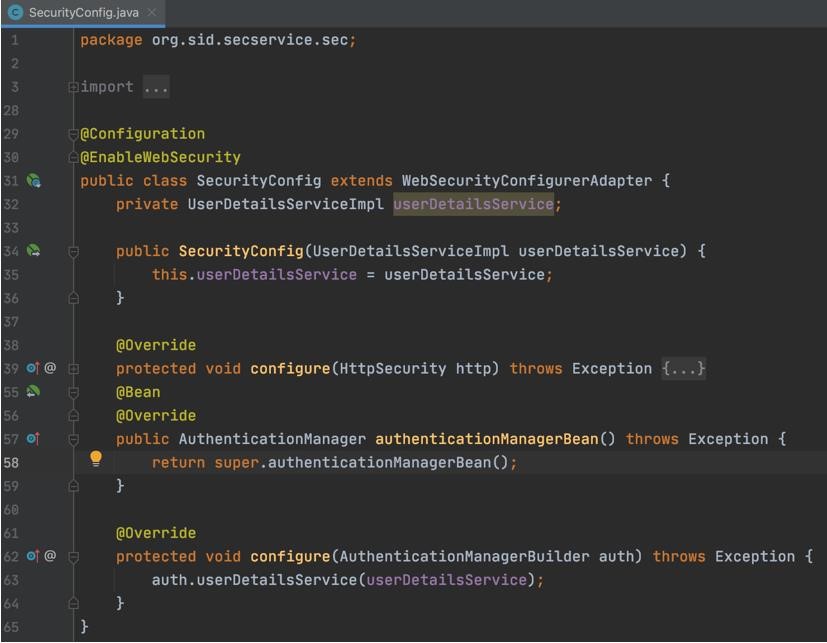
TEST DU MICROSERVICE :



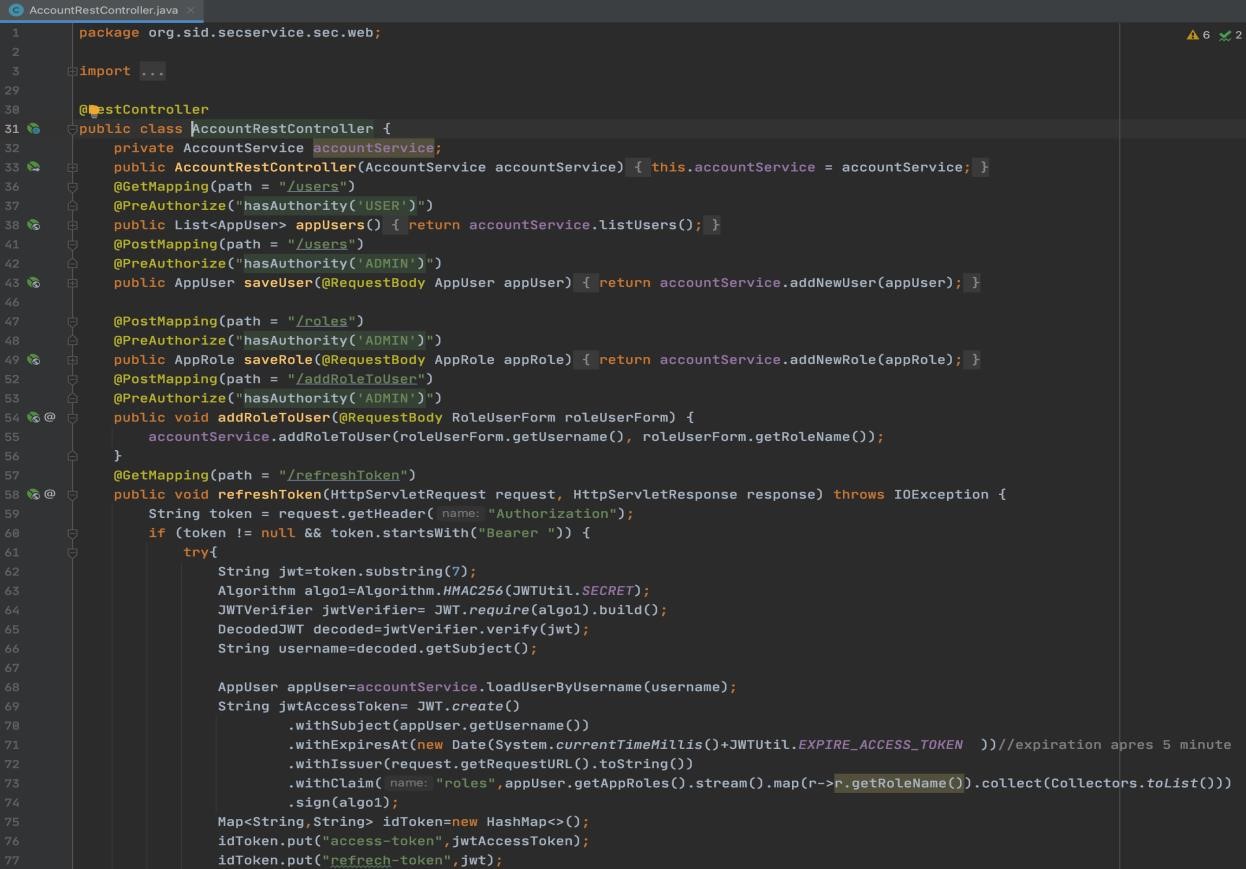
PAGE

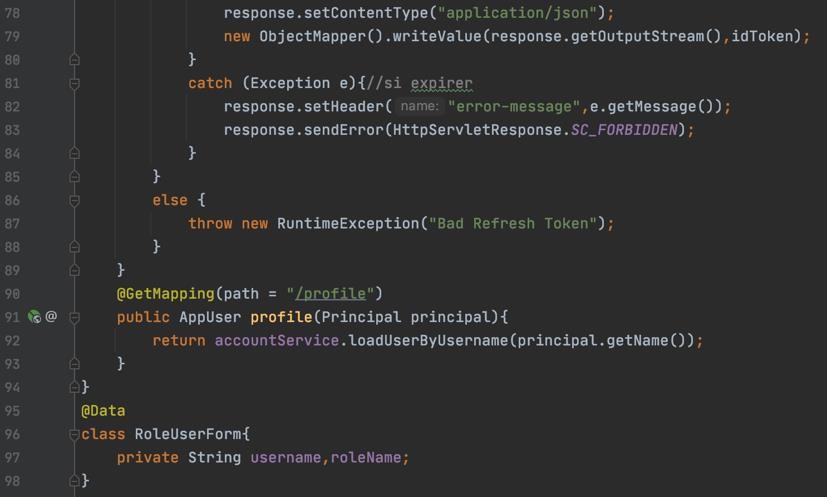
org.sid.secservice.sec:



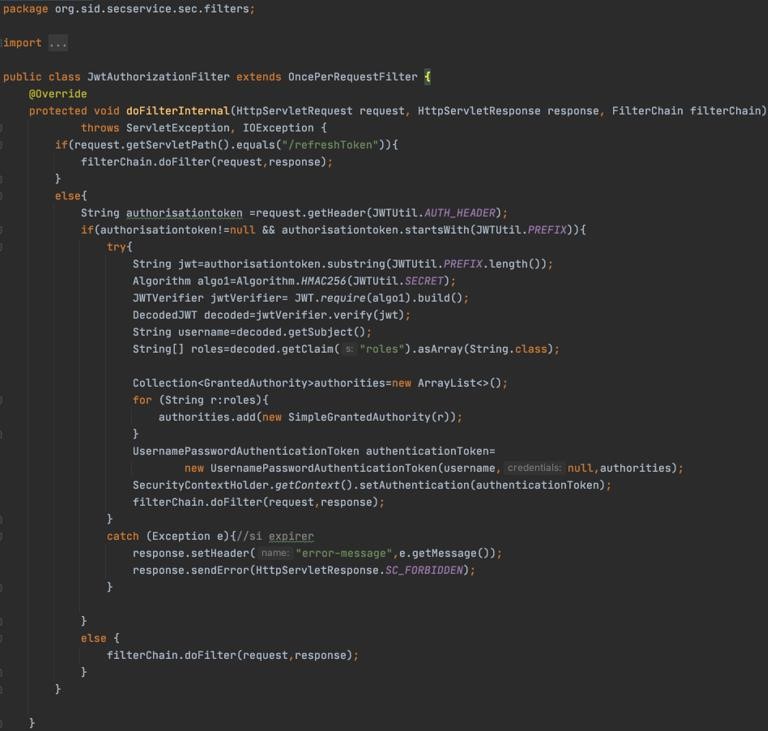


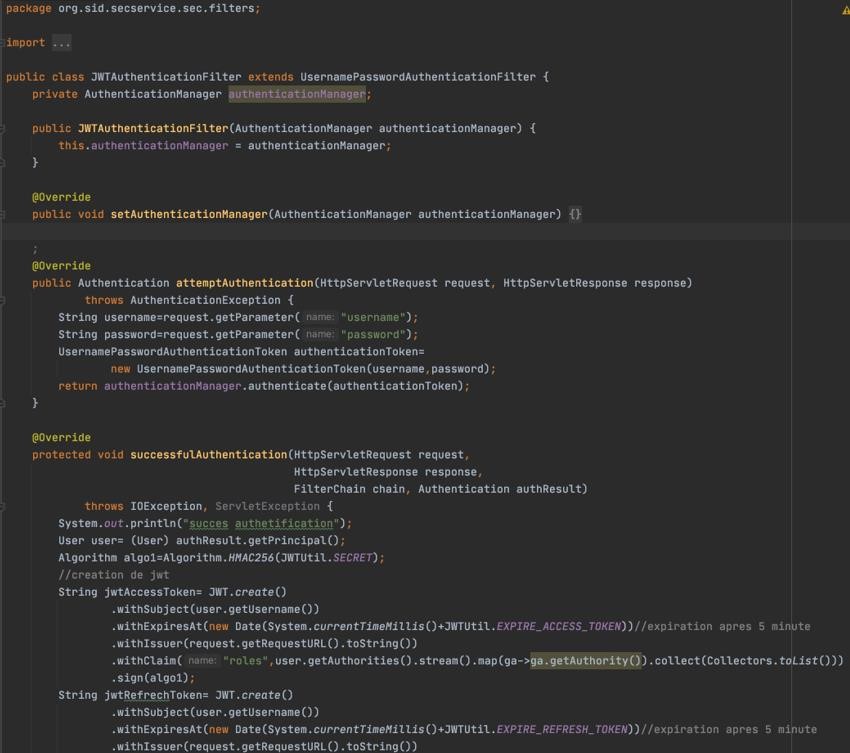
WEB:





FILTER:

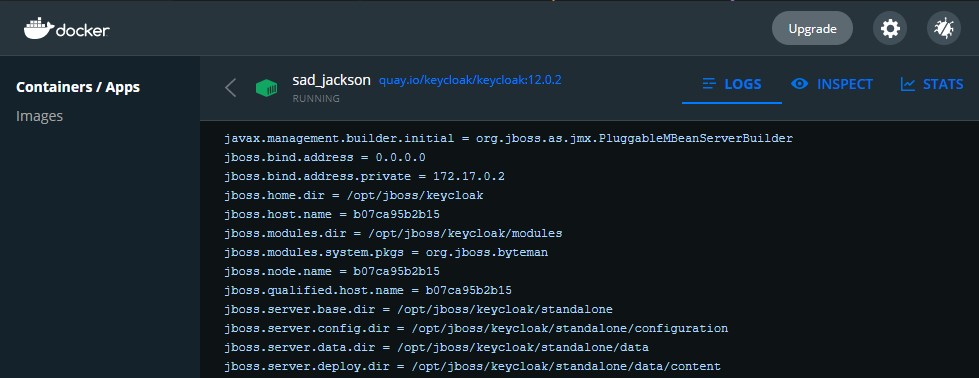




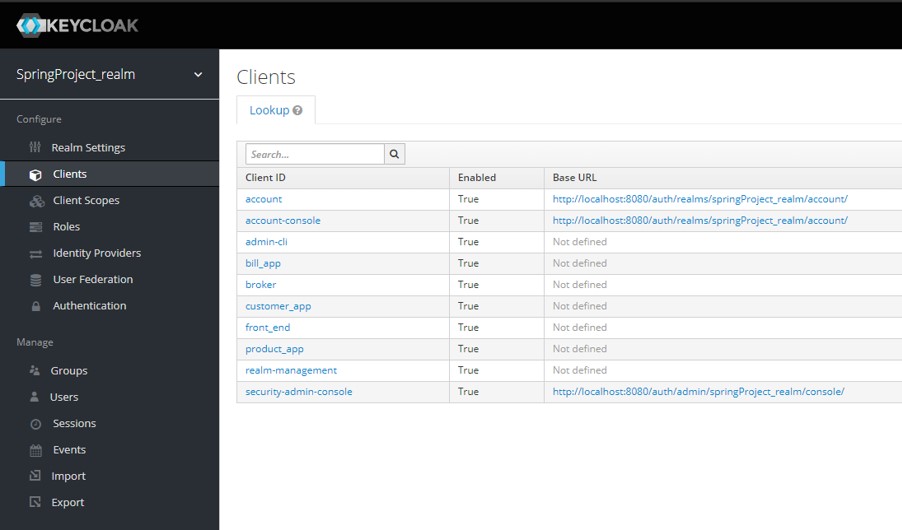
# Keycloak

Conteneur Keycloak Docker

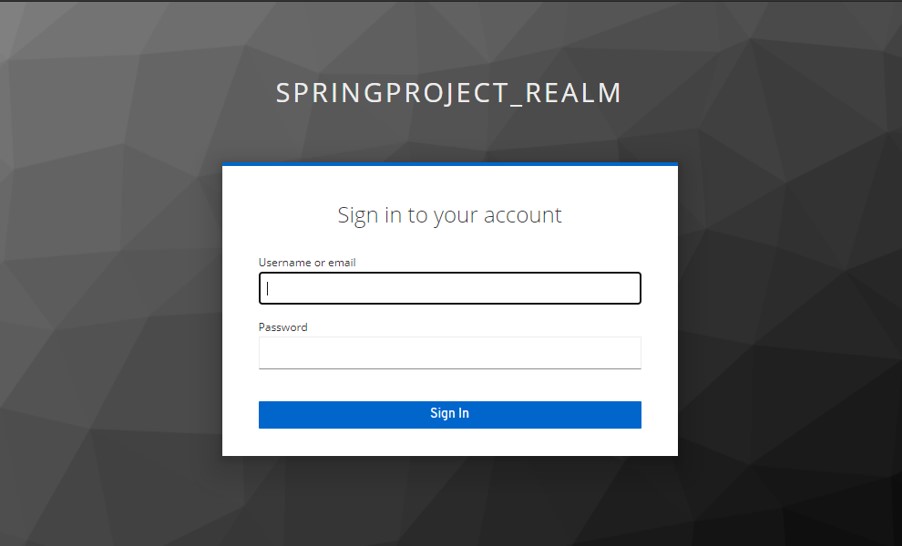
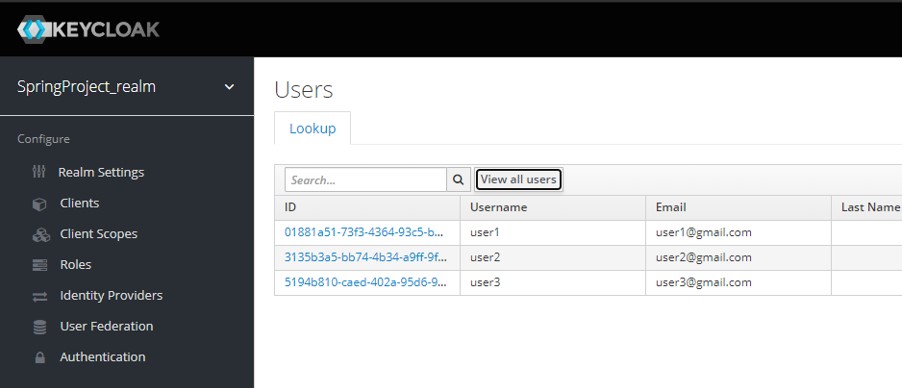
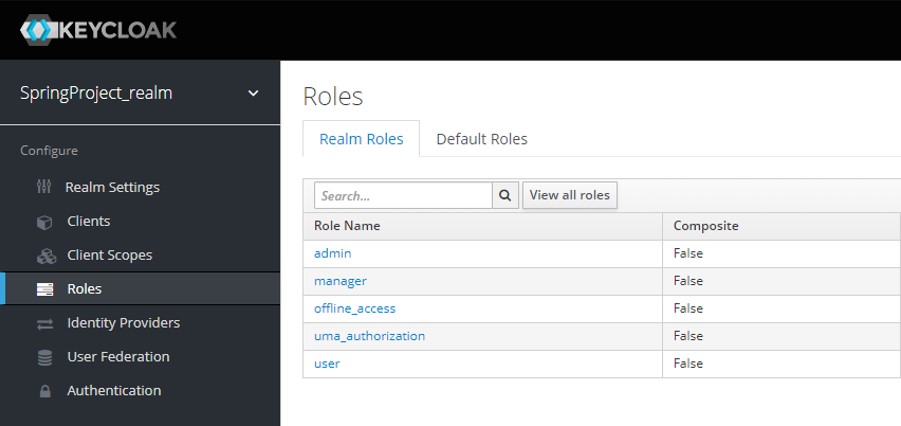
* docker run -p 8080:8080 -e KEYCLOAK\_USER=admin -e
* KEYCLOAK\_PASSWORD=admin quay.io/keycloak/keycloak:12.0.3



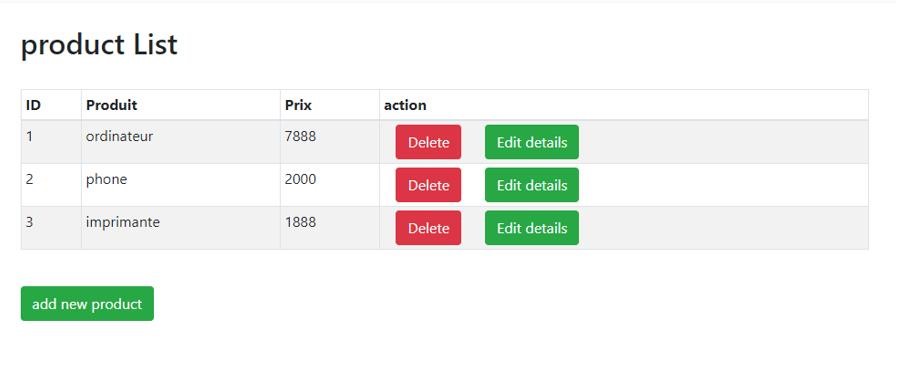
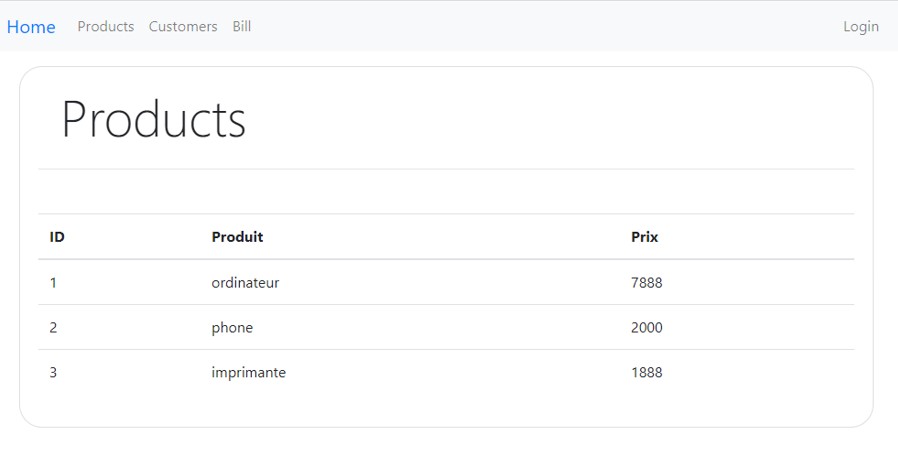
Mise en place d’un Realm et les clients :



Création des user et les rôles :

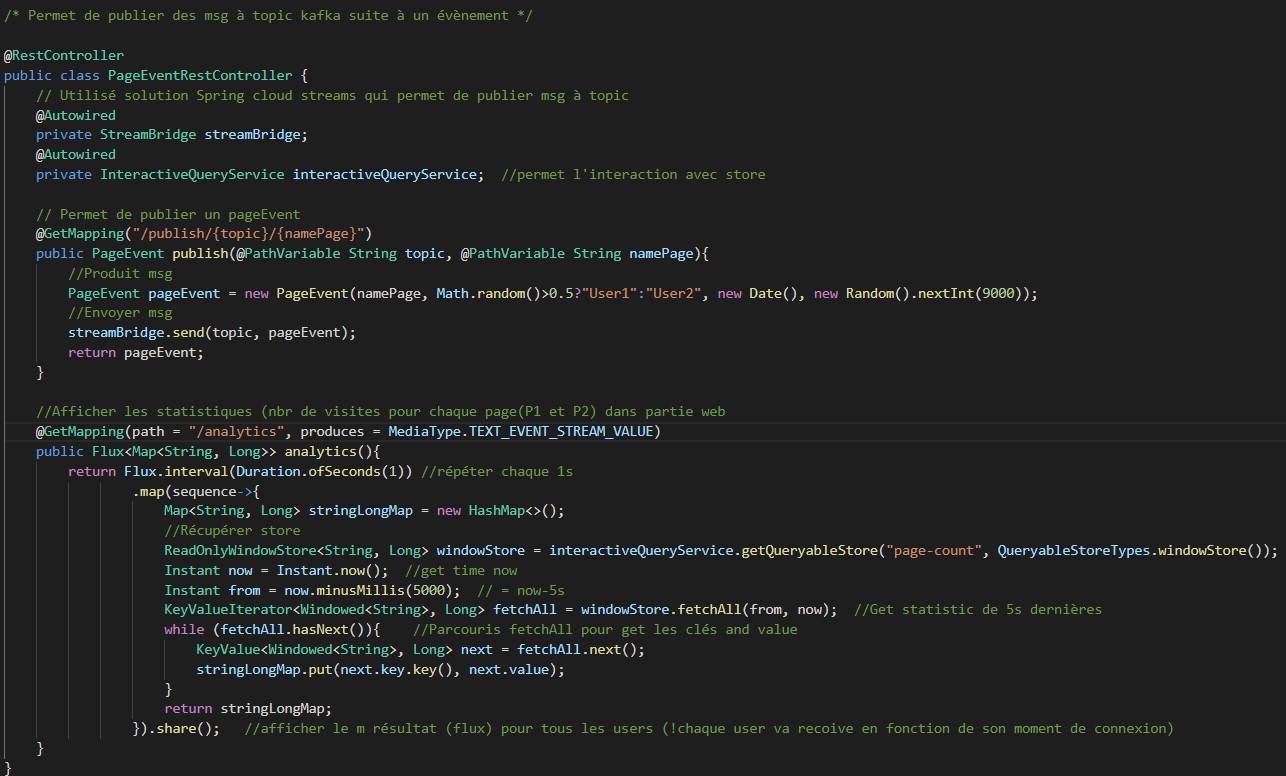


# Front-End



# Kafka

PageEventController



PageEnventService

