

**Systèmes Distribués**

**“Gérer les factures”**

**­**

**Nom**

**EL IDR1SSI Mohamed**

**Filière :** IAAD

**Année universitaire :** 2021-2022

**Rapport de Mini-Projet**

**Table des matières**

[1. Introduction : 3](#_Toc105867646)

[1. Les outils de développement : 4](#_Toc105867647)

[ Back-end 4](#_Toc105867648)

[ Front-end 4](#_Toc105867649)

[ Base de données 4](#_Toc105867650)

[ Autres outils 5](#_Toc105867651)

[2. Mettre en place les micro-services 6](#_Toc105867652)

[3. Mise en place du service de Sécurité avec Keycloak 7](#_Toc105867653)

[ Mettre en place le serveur d’authentification OAuth2 Keycloak version 12.0.1 7](#_Toc105867654)

[ Créer un Realm 8](#_Toc105867655)

[ Le client à sécuriser en mode public client 8](#_Toc105867656)

[ Créer les rôles (USER, ADMIN, PRODUCT\_MANAGER, CUSTOMER\_MANAGER et BILLING\_MANAGER) 9](#_Toc105867657)

[ Créer quelques utilisateurs, Affecter les rôles aux utilisateurs 9](#_Toc105867658)

[ Personnaliser le paramétrage des timeouts des tokens 10](#_Toc105867659)

[4. Sécurité l’ensemble des micro-services fonctionnels en mode Bearer-Only 11](#_Toc105867660)

[ Configuration real pour application.proprties 11](#_Toc105867661)

[ Keycloak configuration 11](#_Toc105867662)

[5. Développer une application Web Front End 12](#_Toc105867663)

[ ROLE\_CUSTOMER\_SERVICE 12](#_Toc105867664)

[ Configuration keycloak 12](#_Toc105867665)

[ Home page 13](#_Toc105867666)

[ Login page 13](#_Toc105867667)

[ Home apres sign-in par un customer utilisateur 14](#_Toc105867668)

[ Résultat de la consolation de la page produit par un customer 14](#_Toc105867669)

[ Page customer 15](#_Toc105867670)

[ Ajouter des produits au panier 15](#_Toc105867671)

[ Générer et imprimer la facture 16](#_Toc105867672)

[ ROLE\_PRODUCT\_SERVICE 16](#_Toc105867673)

[ Interface product 16](#_Toc105867674)

[ Ajouter un produit 5 17](#_Toc105867675)

[ Modifié les informations de produit 1 17](#_Toc105867676)

[ Supprimé le produit 3 17](#_Toc105867677)

[ ROLE\_ADMIN 18](#_Toc105867678)

[ Consulter toutes les factures 18](#_Toc105867679)

[6. Sécuriser l’application Front end en mode public client 19](#_Toc105867680)

[7. Personnaliser la sécurité de la partie frontend 20](#_Toc105867681)

[ Auto-inscription des utilisateurs 20](#_Toc105867682)

[ Politique des mots de passe 20](#_Toc105867683)

[ Double authentification OTP 21](#_Toc105867684)

[8. Mise en place d’une solution de messagerie asynchrone avec le Broker KAFKA 22](#_Toc105867685)

[i. Mettre en place le Broker KAFKA 22](#_Toc105867686)

[ii. Broker KAFKA permet d’envoyer à un tompic « FACTURATION » 22](#_Toc105867687)

[ Configuration application.properties de Producer 22](#_Toc105867688)

[ La Création des factures aléatoirement et de les envoyés au Broker Kafka chaque seconde 23](#_Toc105867689)

[ Run Producer 24](#_Toc105867690)

[iii. Consommer les messages du Topic « FACTURATION » 25](#_Toc105867691)

[ Consumer Deserializer 25](#_Toc105867692)

[ Lire les messages et les enregistrer dans BD et dans un fichier CSV 25](#_Toc105867693)

[ Base de données 26](#_Toc105867694)

[ Fichier .txt 27](#_Toc105867695)

[ Une API REST qui permet de consulter les factures 28](#_Toc105867696)

[9. Lien de GitHub : 29](#_Toc105867697)

# Introduction :

Ce rapport présentera notre travail de mini-projet du module de Systèmes Distribués, afin d’appliquer l’ensemble de nos connaissances acquises durant le cours.

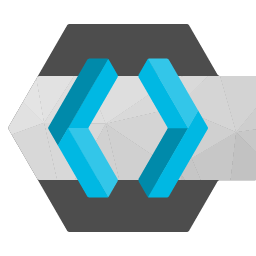
L’objectif est de créer un système distribué basé sur les micro-services permettant de gérer les factures des clients en utilisant la même architecture que vous avez déjà développé auparavant en y intégrant un système de sécurité basé sur Keycloak, Un Bus de messagerie avec KAFKA, un service de Stream processing avec Kafka Streams et un service de Batch Processing avec Spring Batch.

# Les outils de développement :

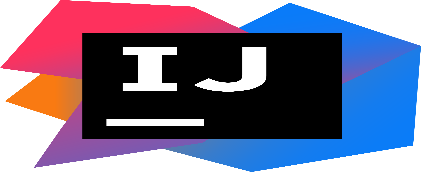
## Back-end

**Spring boot** : Spring est un [framework](https://fr.wikipedia.org/wiki/Framework" \o "Framework) open source pour construire et définir l'infrastructure d'une application [Java](https://fr.wikipedia.org/wiki/Java_(technique))[3](https://fr.wikipedia.org/wiki/Spring_(framework)#cite_note-3), dont il facilite le [développement](https://fr.wikipedia.org/wiki/D%C3%A9veloppement_de_logiciel) et les tests.

**KAFKA** : est une plateforme de streaming d'événements distribués open source utilisée par des milliers d'entreprises pour les pipelines de données hautes performances, l'analyse de streaming, l'intégration de données et les applications critiques.

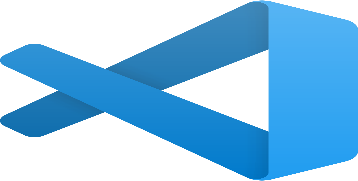


**Keycloak** : est un logiciel à code source ouvert permettant d'instaurer une méthode d'authentification unique à travers la gestion par identité et par accès.



**IntelliJ IDEA** : également appelé « IntelliJ », « IDEA » ou « IDJ » est un environnement de développement intégré destiné au développement de logiciels informatiques reposant sur la technologie Java.

## Front-end

**Angular** : est un framework côté client, open source, basé sur TypeScript, et co-dirigé par l'équipe du projet « Angular » à Google et par une communauté de particuliers et de sociétés. Angular est une réécriture complète d'AngularJS, cadriciel construit par la même équipe.

**Visual Studio** : Un éditeur de code léger développé en 2015 par Microsoft, gratuit et open-source, il contient par défaut des supports pour JavaScript, Type Script et Node.js.

## Base de données

**MySQL** : est un système de gestion de bases de données relationnelles. Il est distribué sous une double licence GPL et propriétaire

## Autres outils

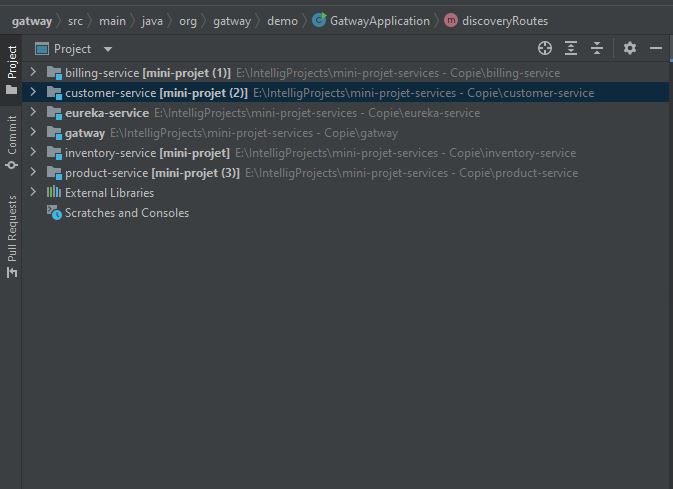
**GitHub :** Est un outil gratuit pour héberger notre code, qui utilise Git, Git est un gestionnaire de versions des fichiers qui permet d’enregistrer les modifications faites sur le même fichier. C'est le numéro 1 mondial et il héberge plus d'une dizaine de millions de repositories.



**Git Bash :** Git Bash est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. En 2016, il s’agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes.

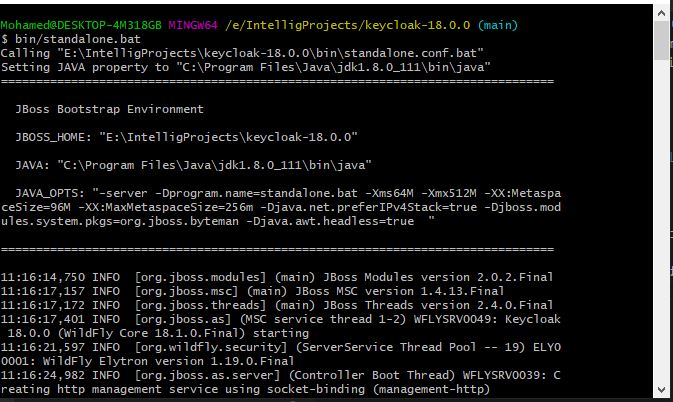
# Mettre en place les micro-services

* a. Customer-Service
* b. Inventory-Service
* c. Billing-Service
* d. Eureka Discovery Service
* e. Spring Cloud Gateway
* f. Product-Service

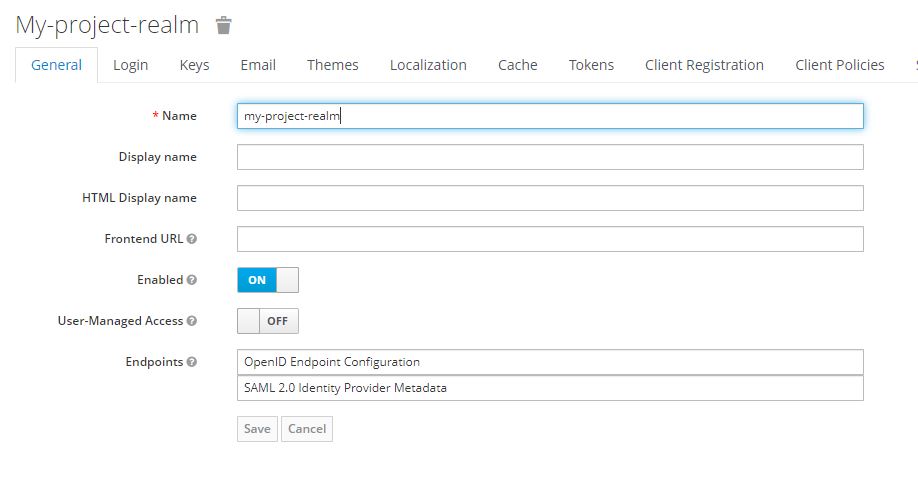


# Mise en place du service de Sécurité avec Keycloak

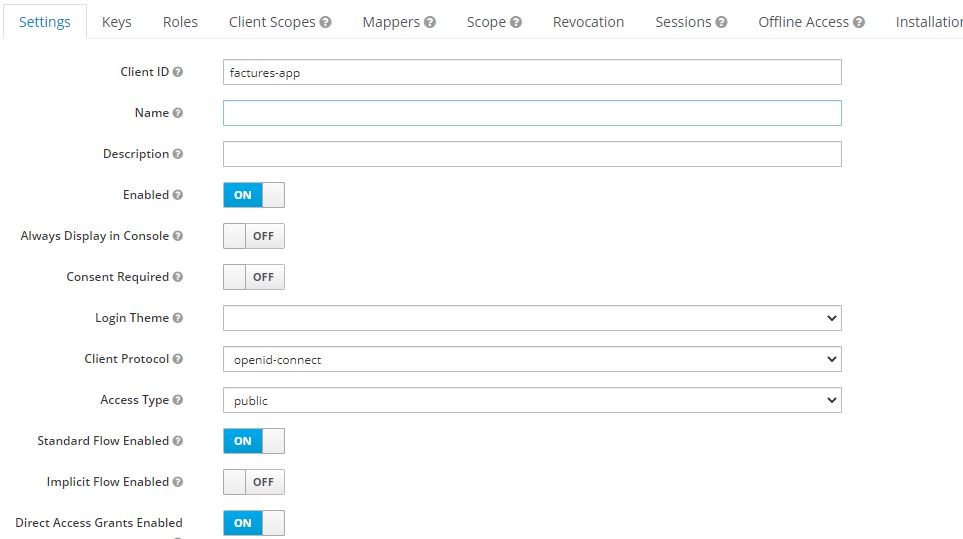
## Mettre en place le serveur d’authentification OAuth2 Keycloak version 12.0.1



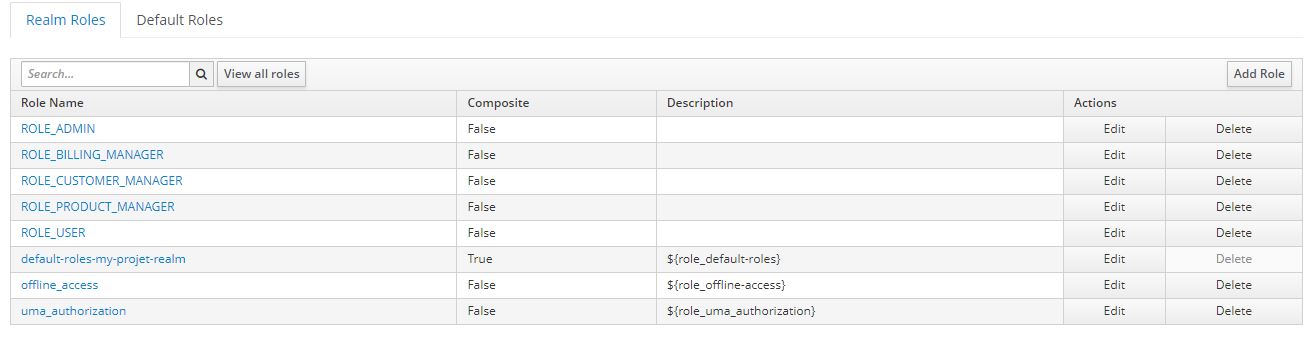
## Créer un Realm



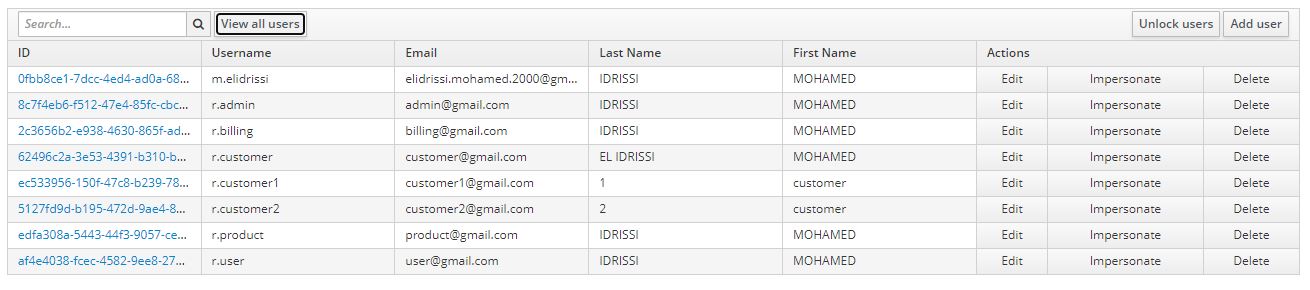
## Le client à sécuriser en mode public client



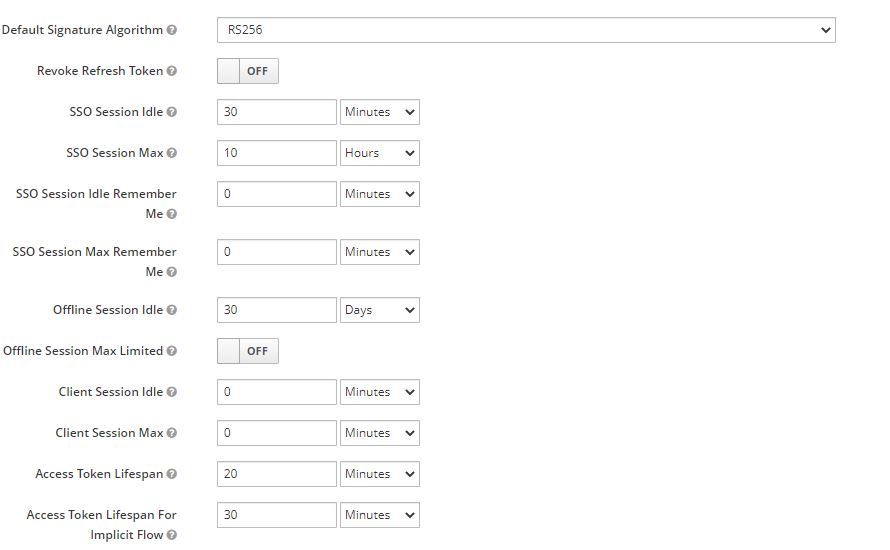
## Créer les rôles (USER, ADMIN, PRODUCT\_MANAGER, CUSTOMER\_MANAGER et BILLING\_MANAGER)



## Créer quelques utilisateurs, Affecter les rôles aux utilisateurs



## Personnaliser le paramétrage des timeouts des tokens

  
  
🡺 Refresh token: 30min

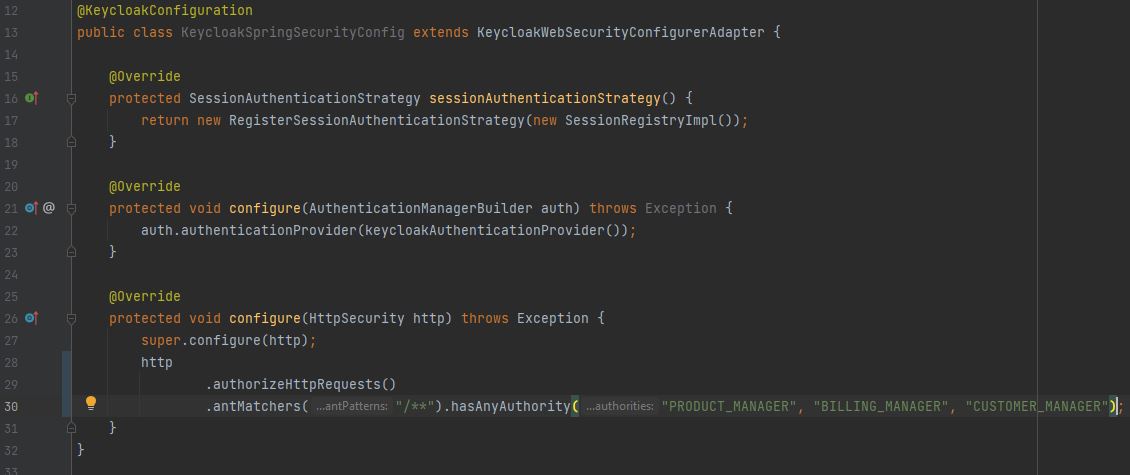
🡺 Access token: 20min

# Sécurité l’ensemble des micro-services fonctionnels en mode Bearer-Only

## Configuration real pour application.proprties



## Keycloak configuration

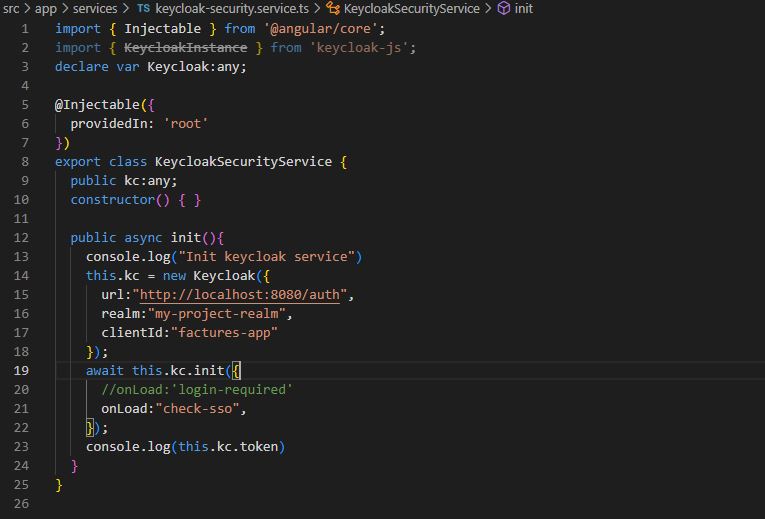


# Développer une application Web Front End

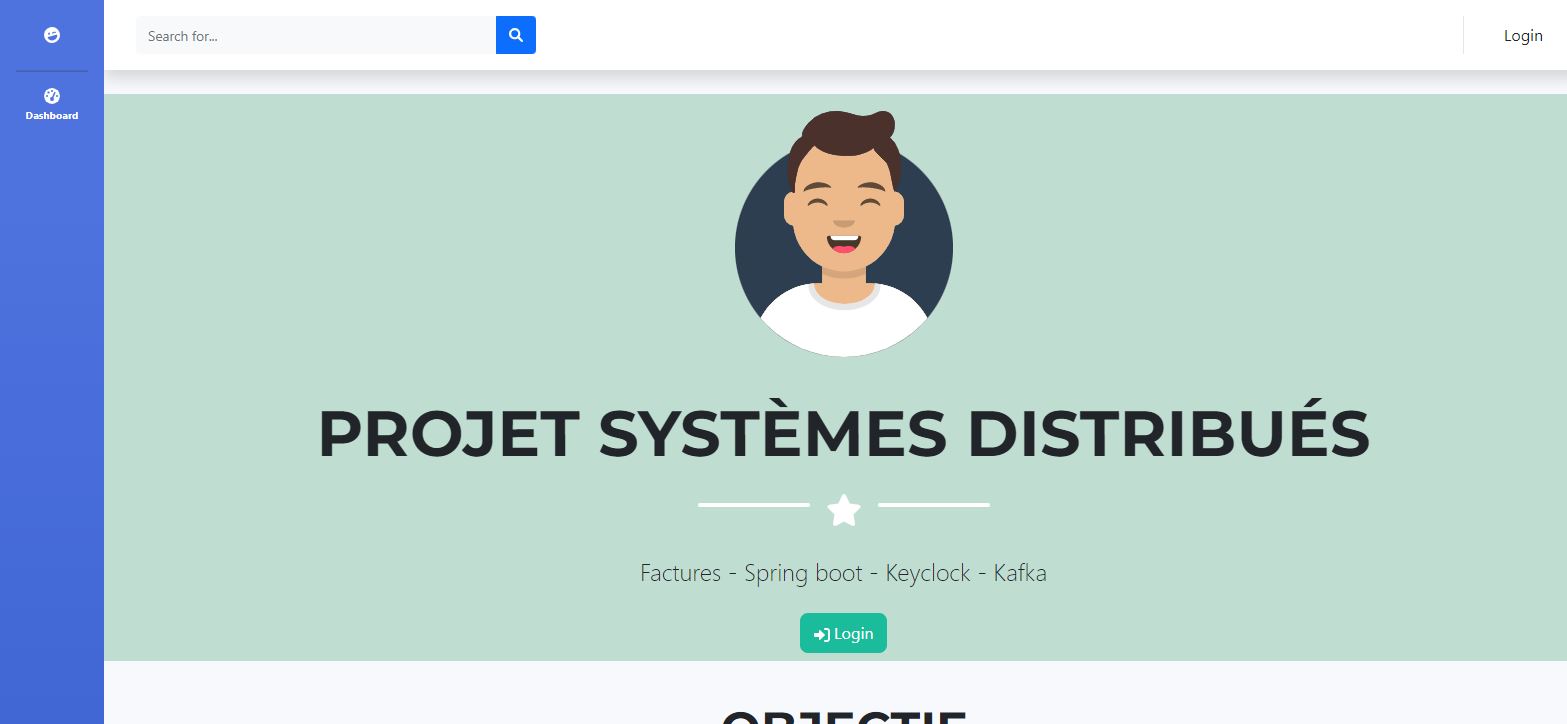
**Pour l'application web j'ai permis d’utiliser le framawork de javascript Angular.**

## ROLE\_CUSTOMER\_SERVICE

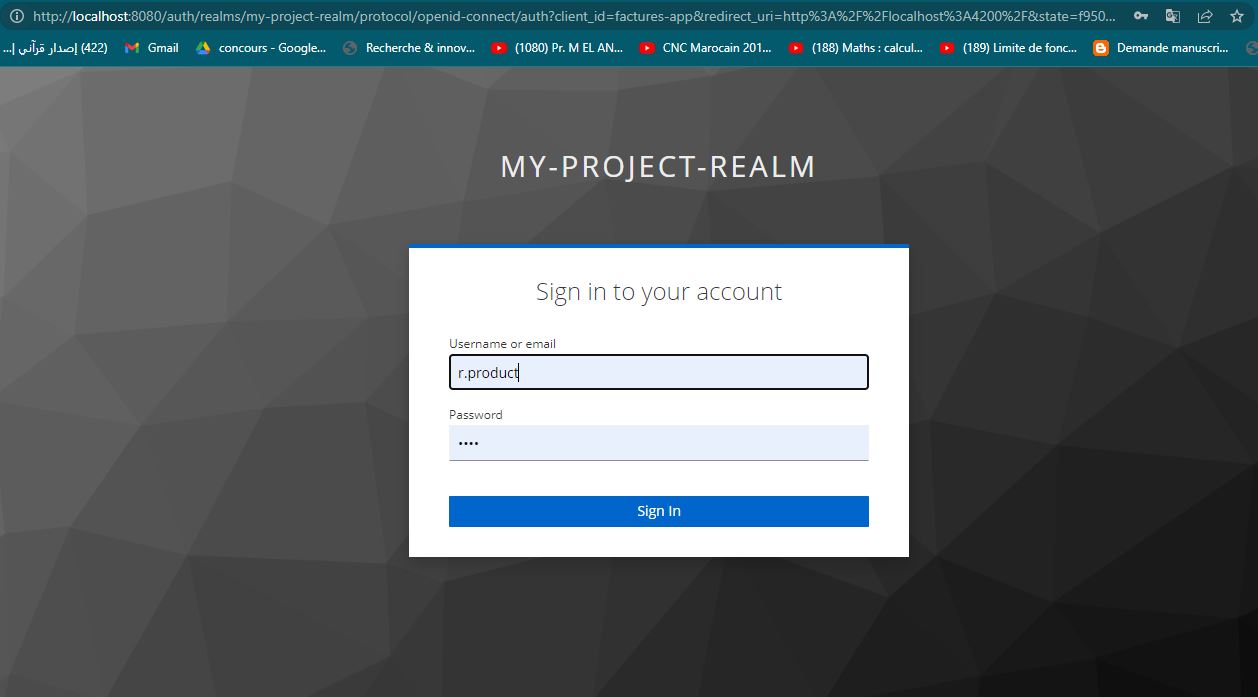
### Configuration keycloak



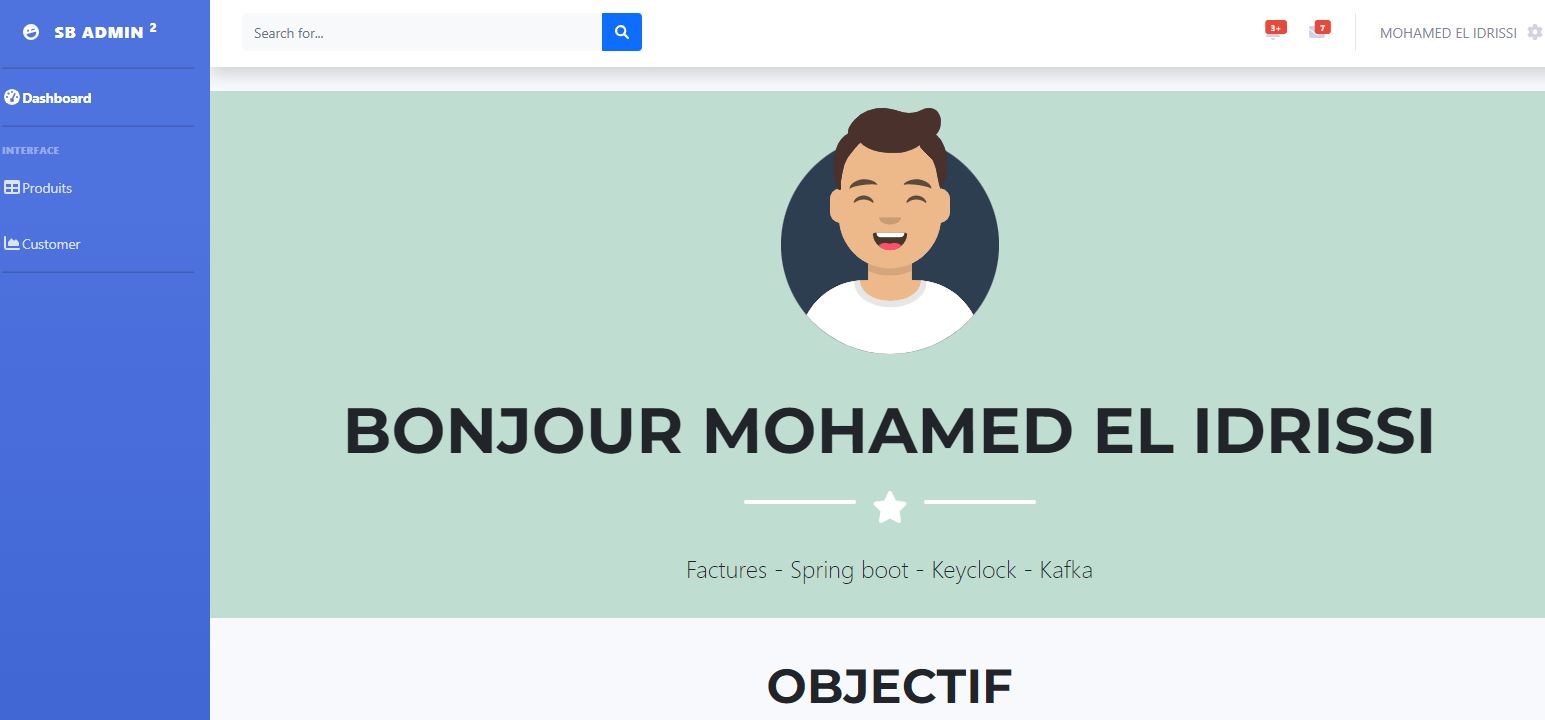
### Home page



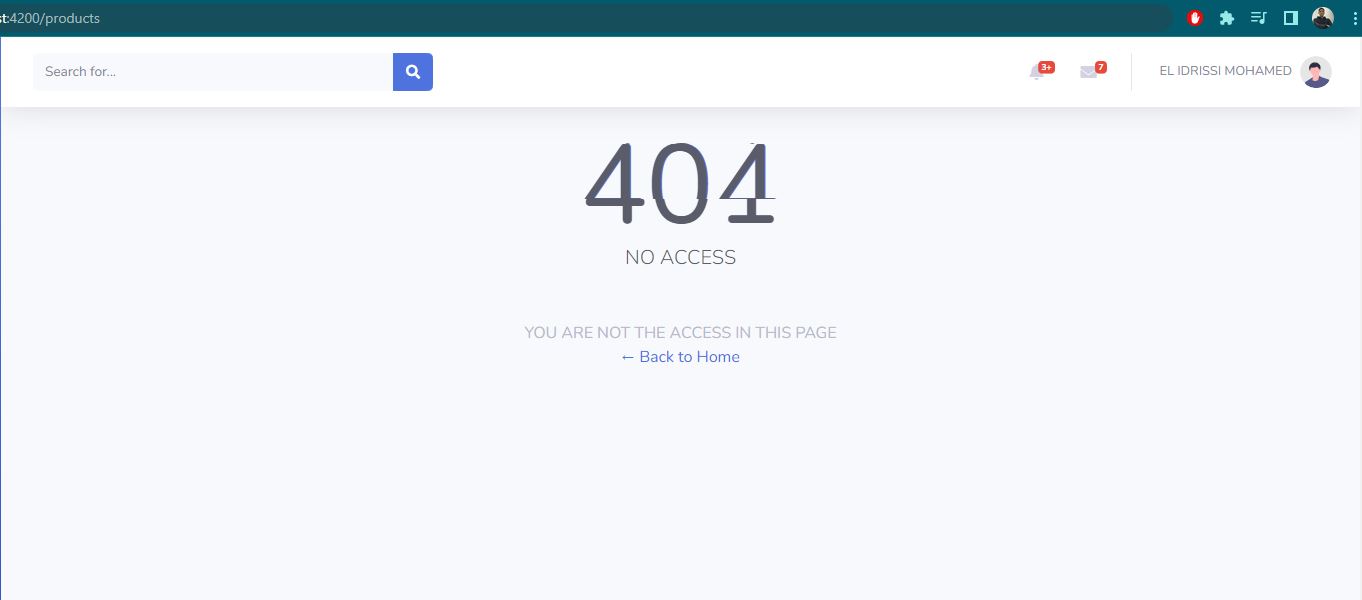
### Login page



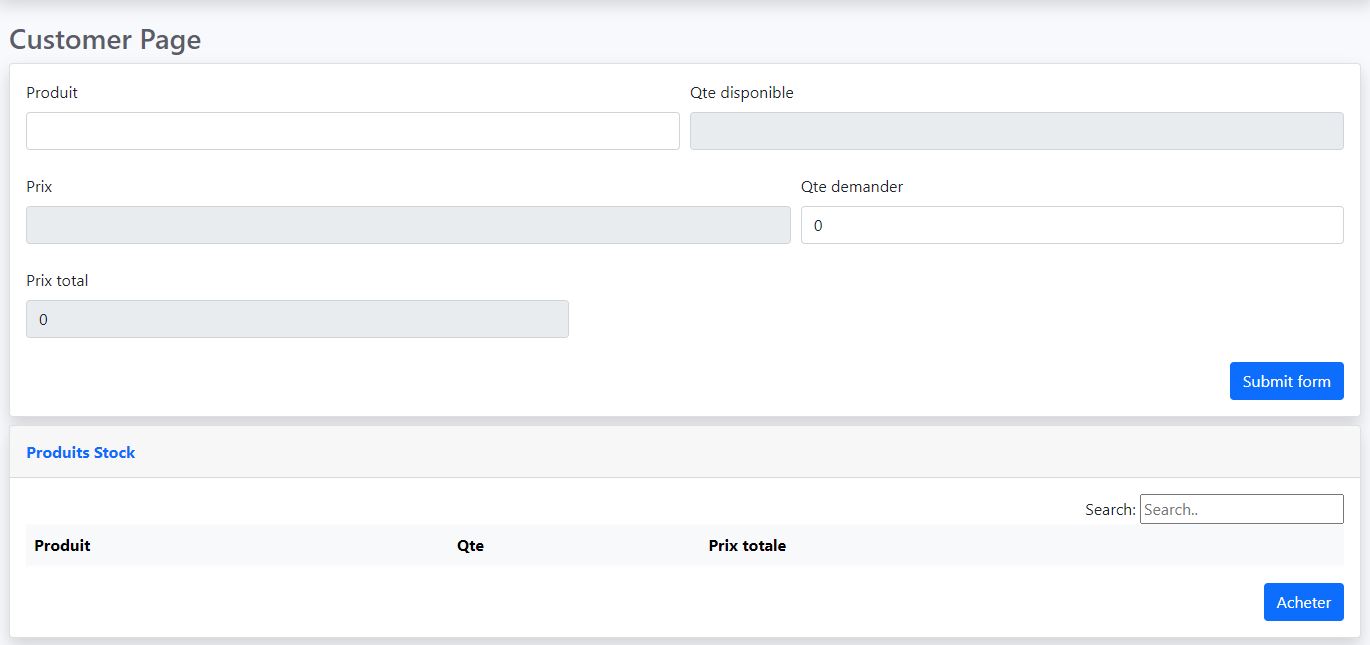
### Home apres sign-in par un customer utilisateur



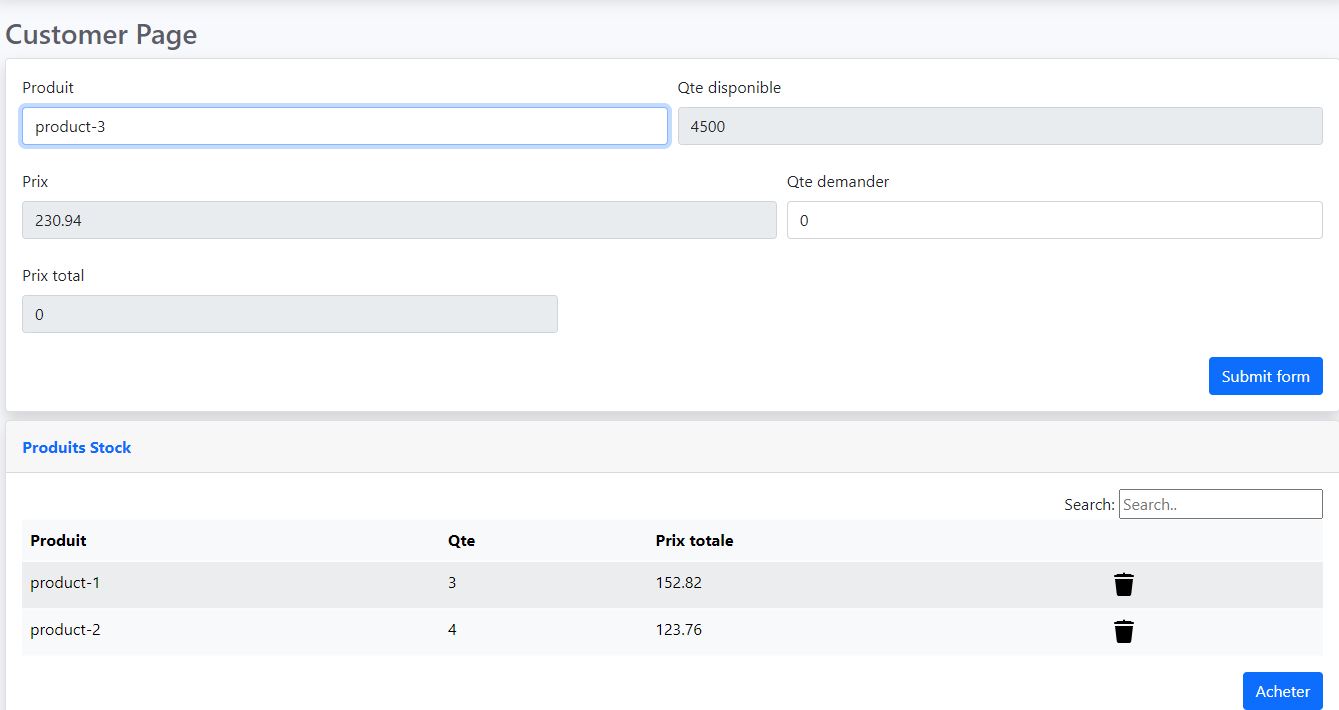
### Résultat de la consolation de la page produit par un utilisateur de rôle customer-manager



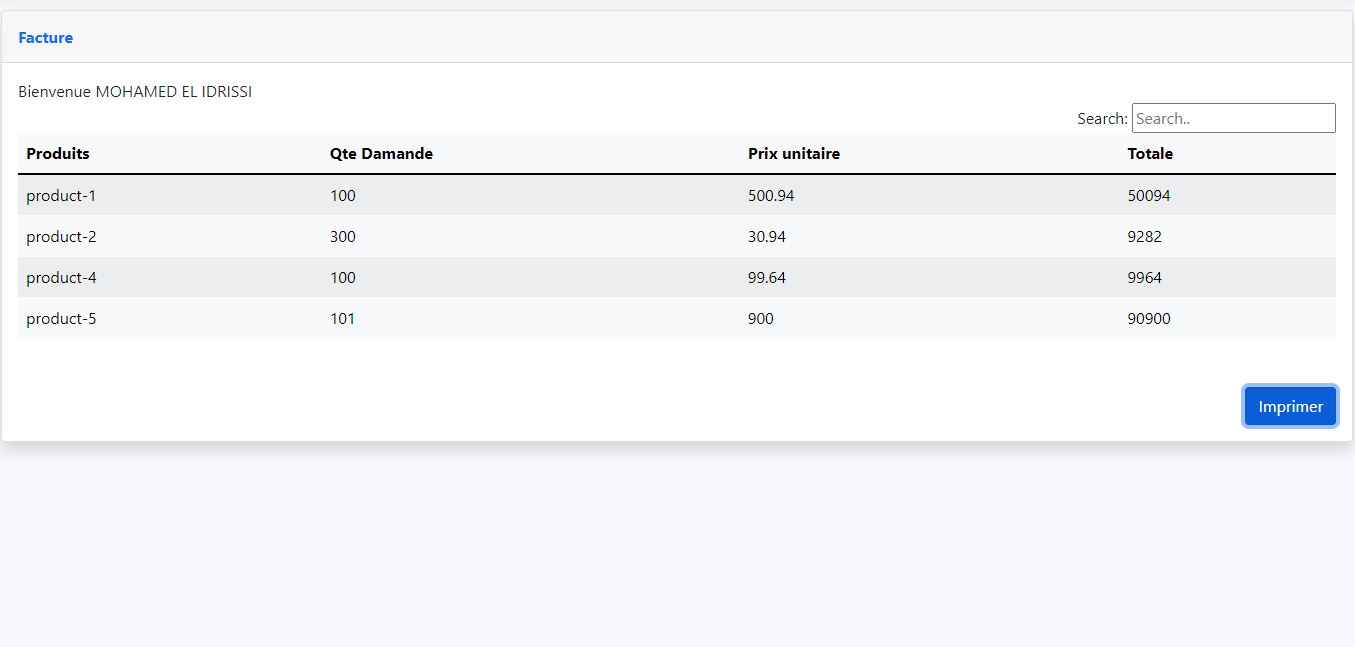
### Page customer



### Ajouter des produits au panier

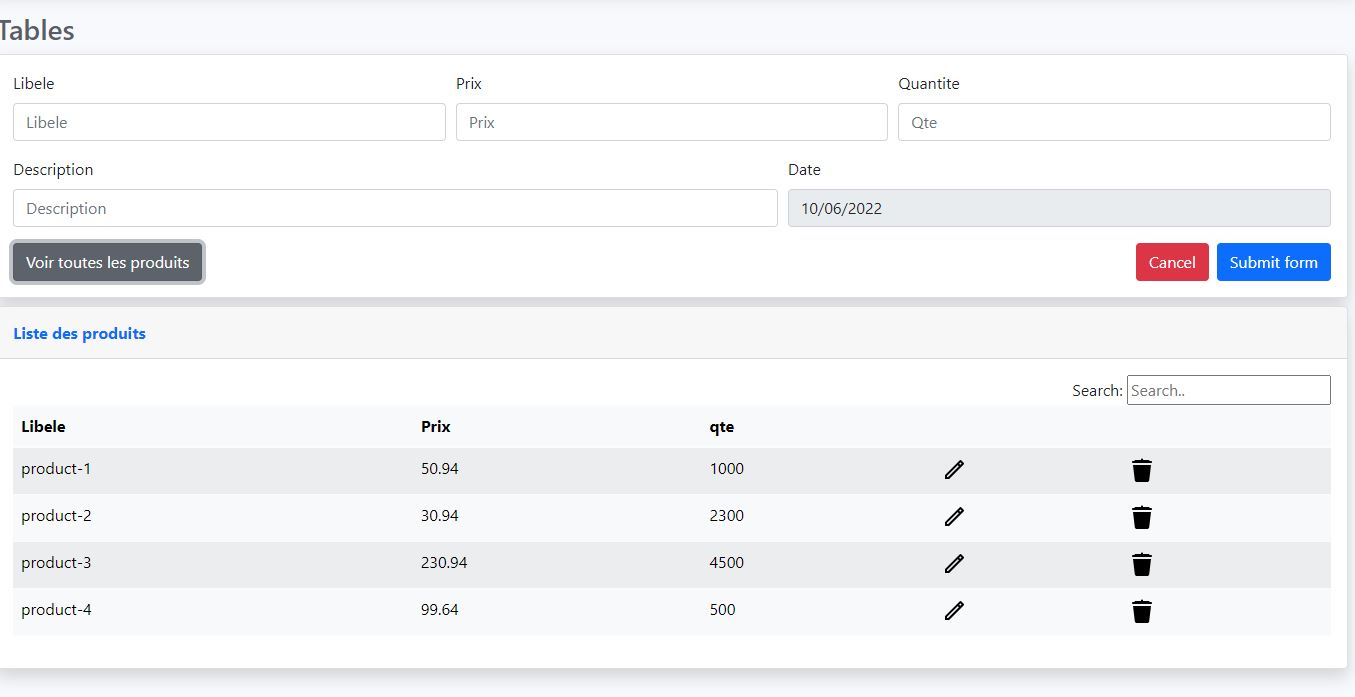
  
  
🡺 Après l'achat la quantité de chaque produit a été modifie

### Générer et imprimer la facture

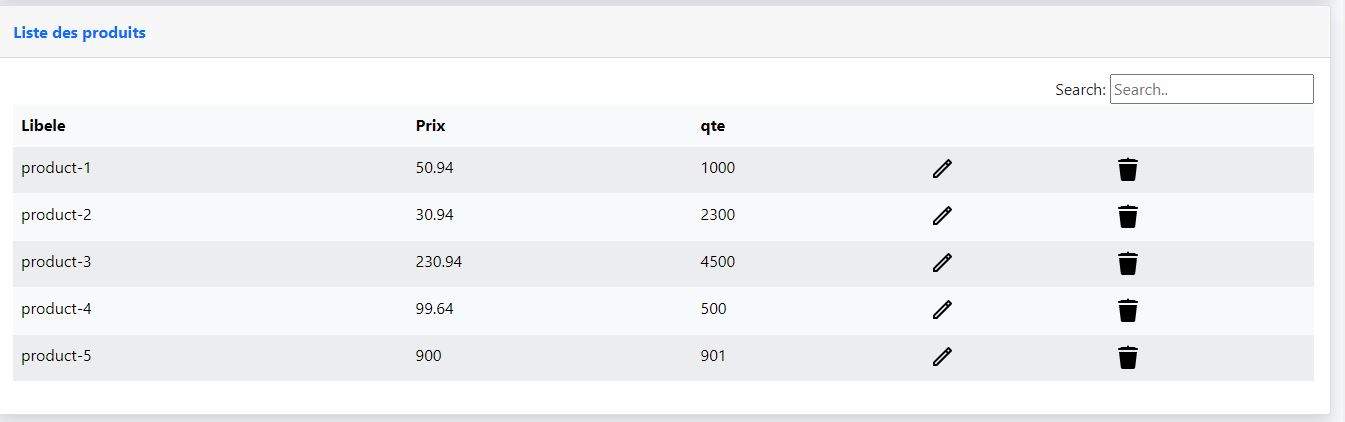


## ROLE\_PRODUCT\_SERVICE

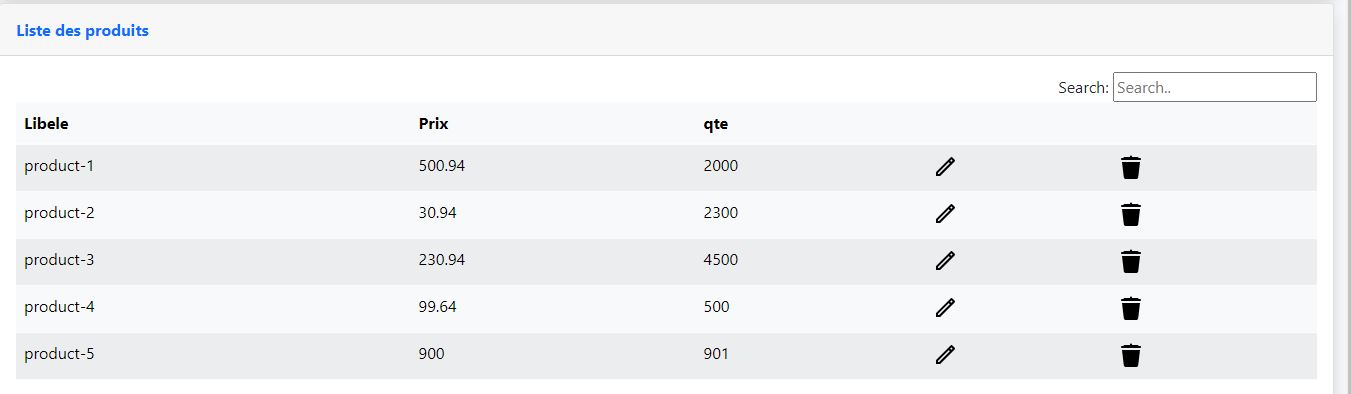
### Interface product



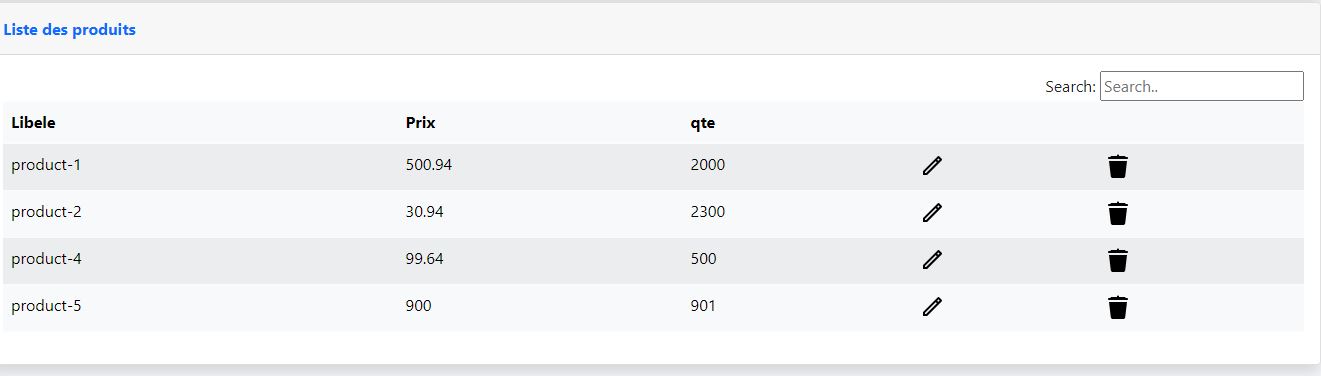
### Ajouter un produit 5



### Modifié les informations de produit 1

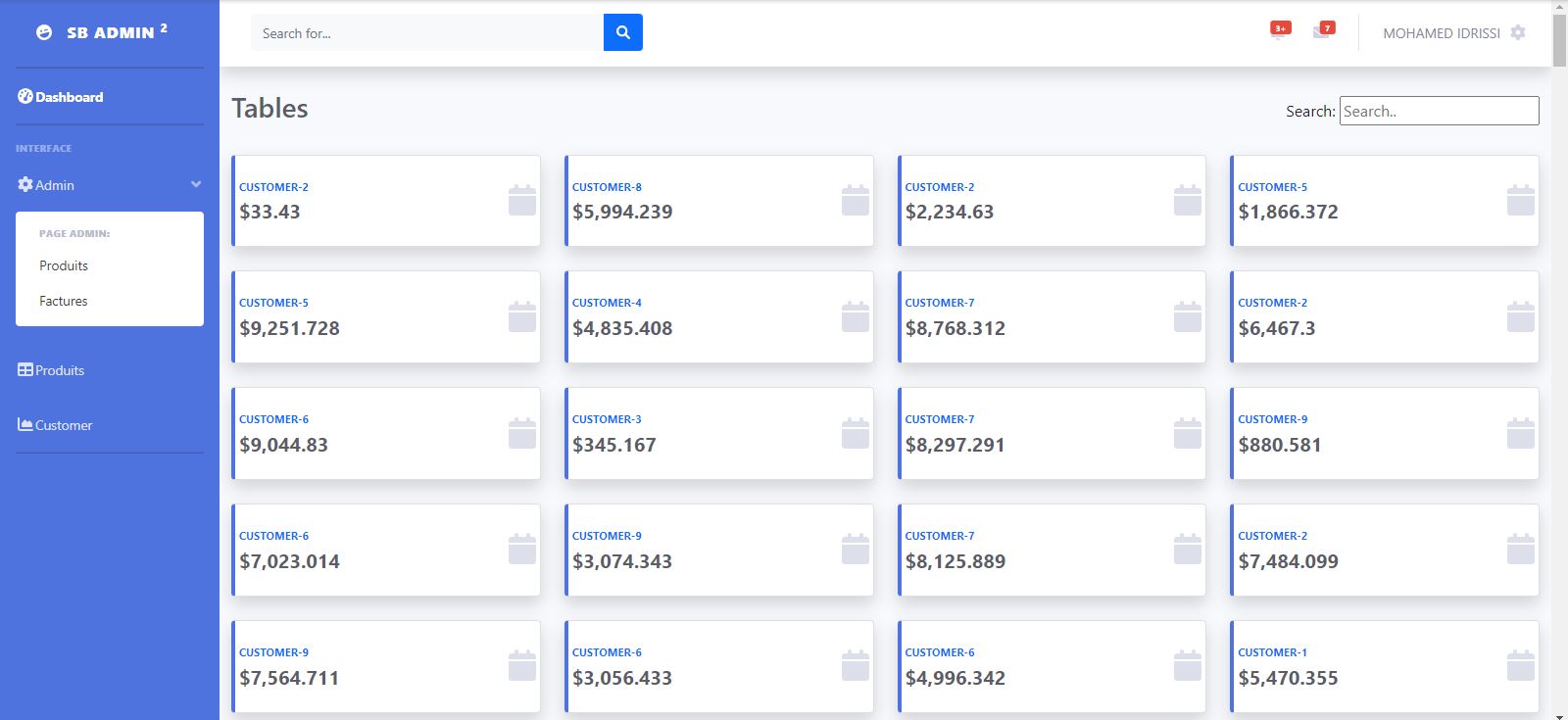


### Supprimé le produit 3

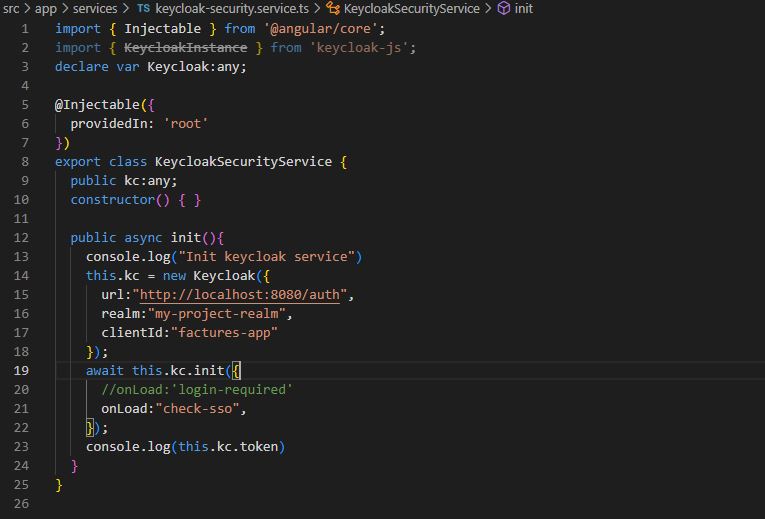


## ROLE\_ADMIN

### Consulter toutes les factures

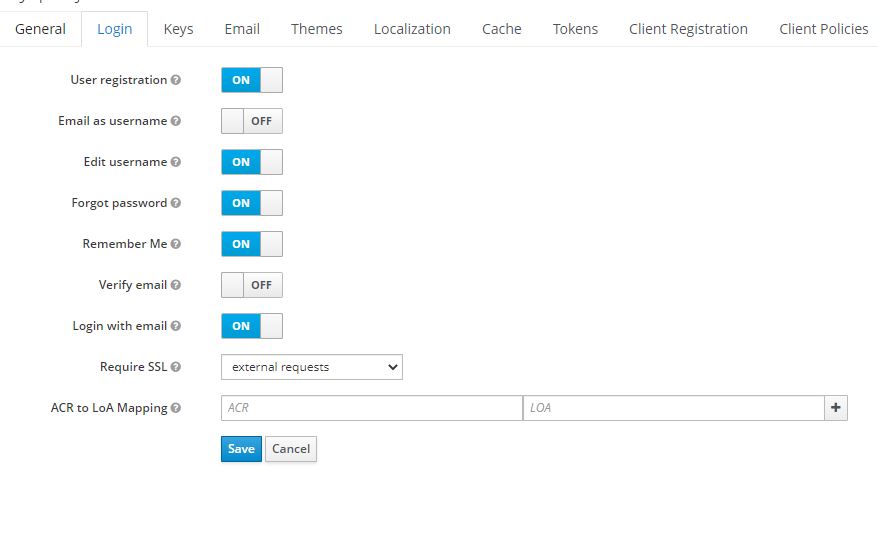


# Sécuriser l’application Front end en mode public client



# Personnaliser la sécurité de la partie frontend

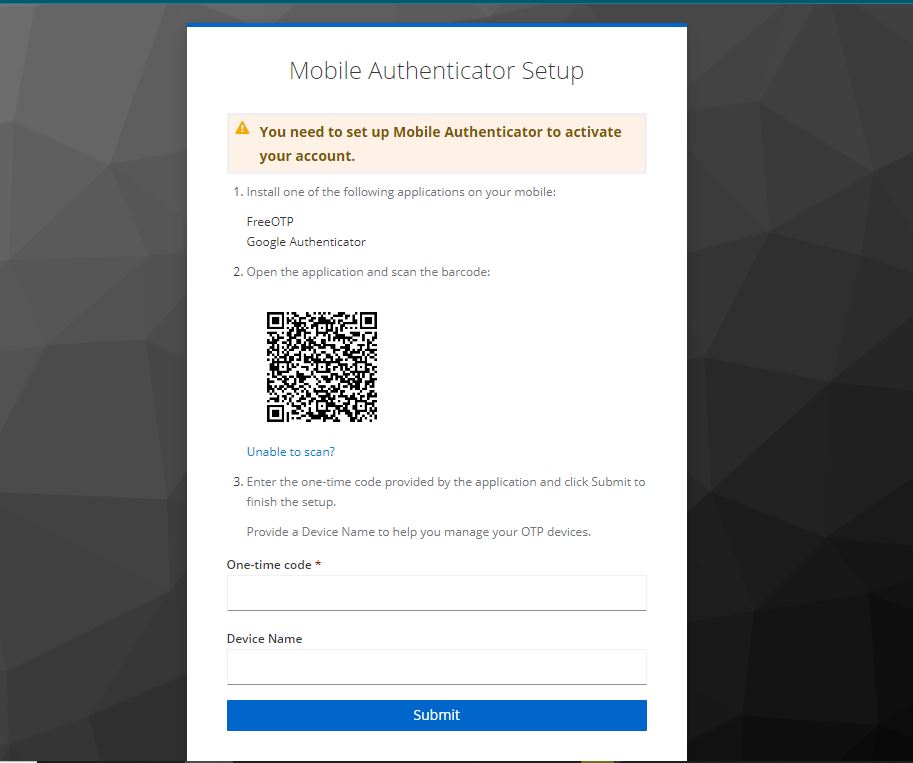
## Auto-inscription des utilisateurs



## Politique des mots de passe

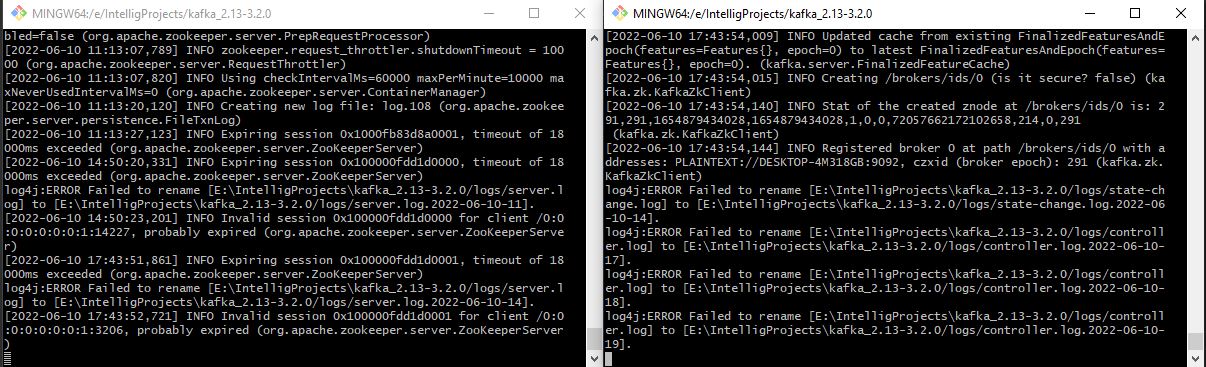


## Double authentification OTP



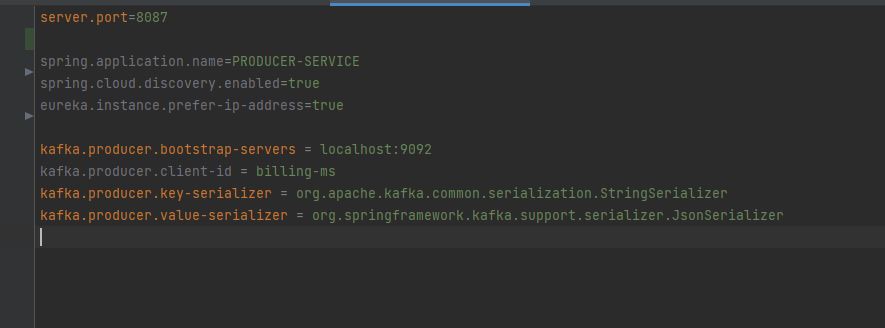
# Mise en place d’une solution de messagerie asynchrone avec le Broker KAFKA

## Mettre en place le Broker KAFKA



## Broker KAFKA permet d’envoyer à un tompic « FACTURATION »

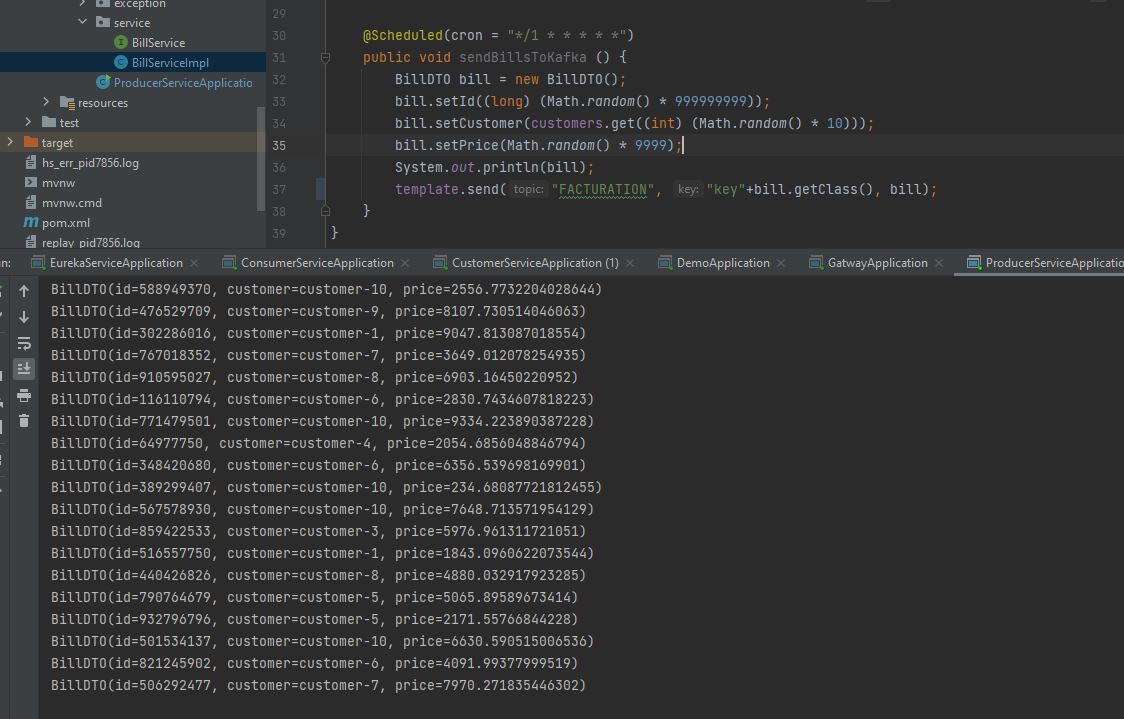
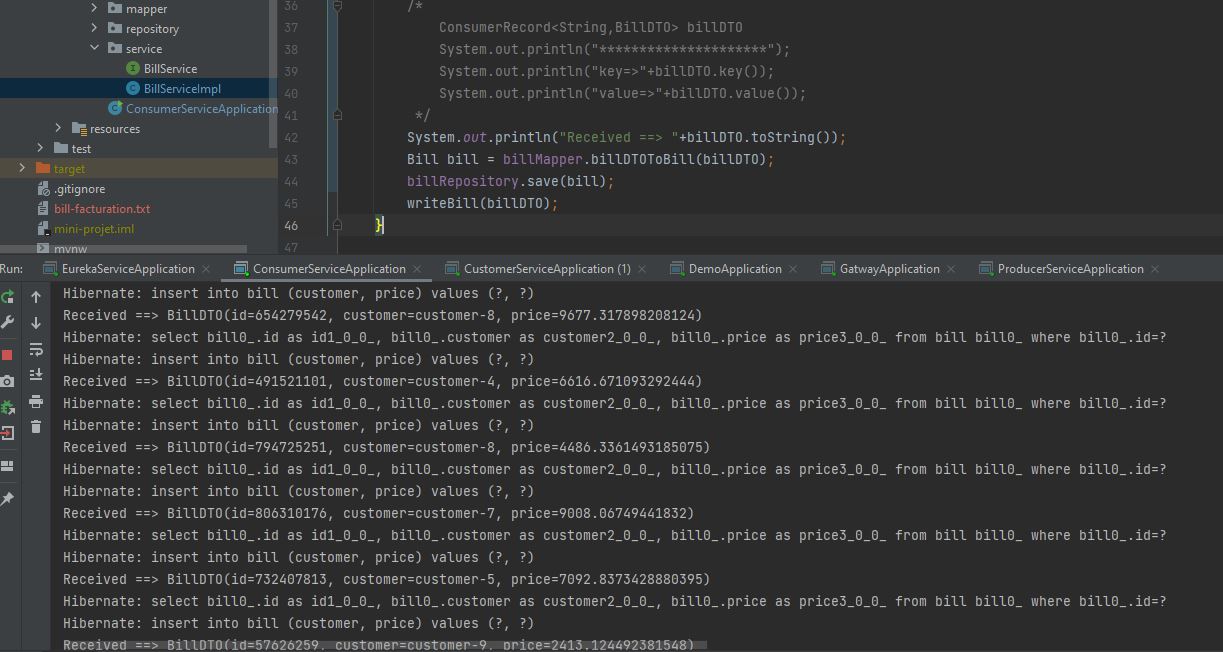
### Configuration application.properties de Producer



### La Création des factures aléatoirement et de les envoyés au Broker Kafka chaque seconde

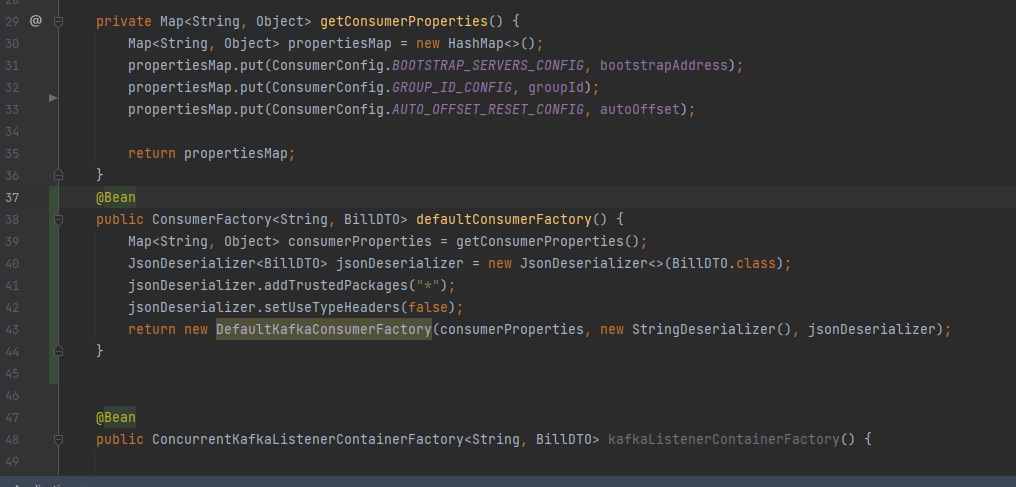


### Run Producer

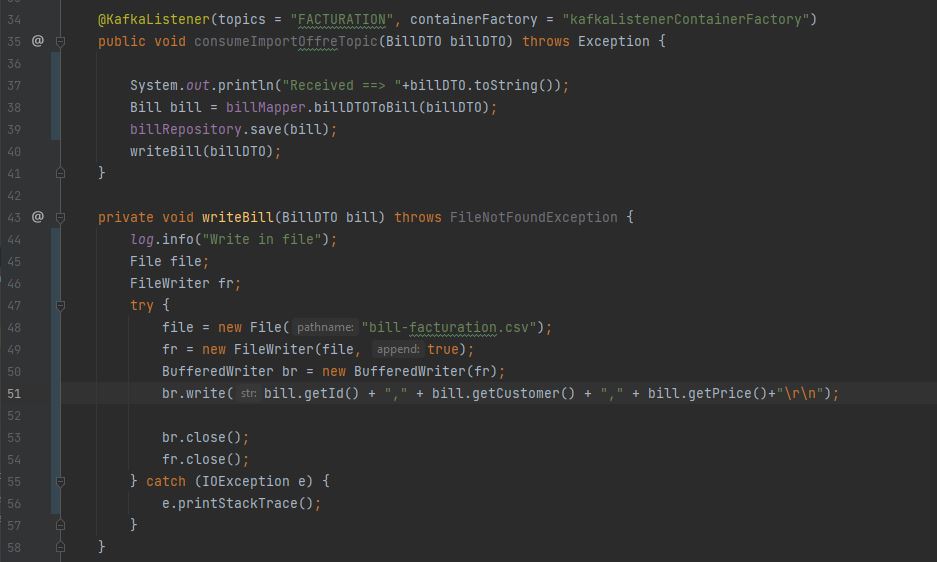
  
  


## Consommer les messages du Topic « FACTURATION »

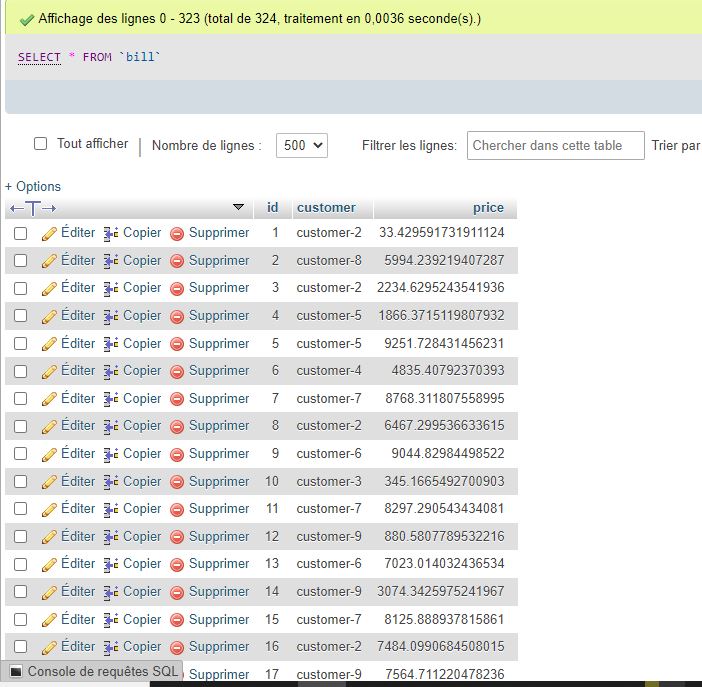
### Consumer Deserializer



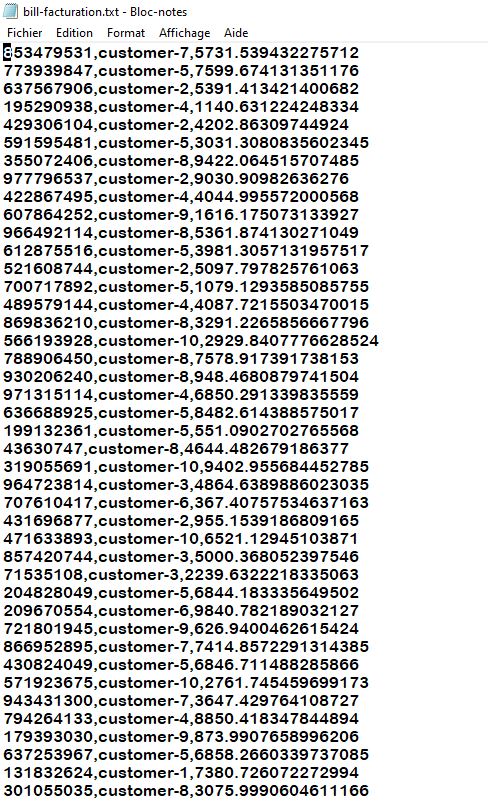
### Lire les messages et les enregistrer dans BD et dans un fichier CSV



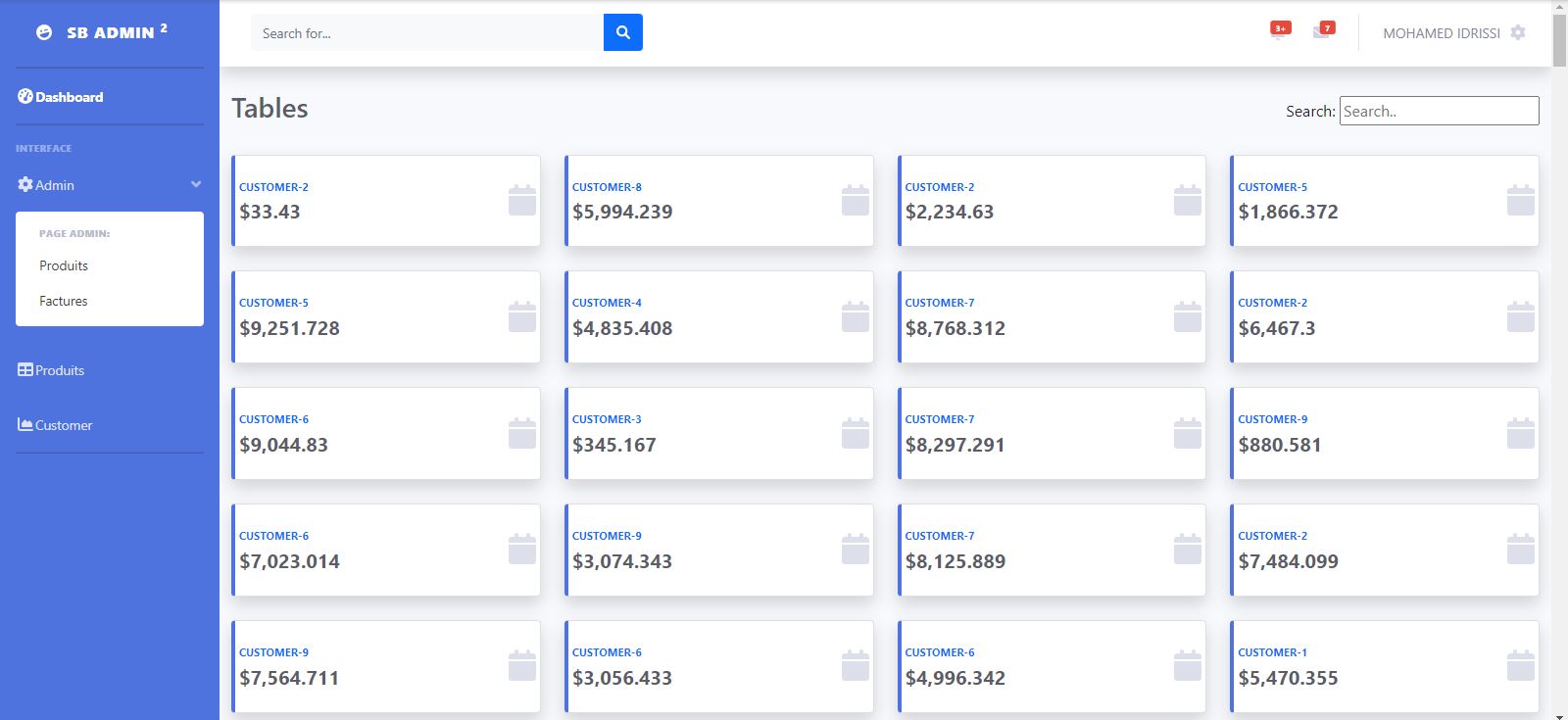
### Base de données



### Fichier .txt



### Une API REST qui permet de consulter les factures



# **Lien de GitHub** :

**Back-end**: https://github.com/ELIDRISSI-mohamed/mini-projet-microservices-angular-kafka-keyclock

**Front-end**: https://github.com/ELIDRISSI-mohamed/microservice-angular