



جامعة مولاي إسماعيل
ትዕሊም ህጻናት ዕቅድ
UNIVERSITÉ MOULAY ISMAÏL



كلية العلوم
ትዕሊም ህጻናት ዕቅድ
FACULTÉ DES SCIENCES

Rapport de Mini-Projet

Systemes Distribués

“Gérer les factures”

Nom

EL IDRISSI Mohamed

Filière : IAAD

Année universitaire : 2021-2022

Table des matières

I. INTRODUCTION :	3
1. LES OUTILS DE DEVELOPPEMENT :	4
• Back-end	4
• Front-end	4
• Base de données	4
• Autres outils	5
2. METTRE EN PLACE LES MICRO-SERVICES	6
3. MISE EN PLACE DU SERVICE DE SECURITE AVEC KEYCLOAK	7
• Mettre en place le serveur d'authentification OAuth2 Keycloak version 12.0.1	7
• Créer un Realm	8
• Le client à sécuriser en mode public client	8
• Créer les rôles (USER, ADMIN, PRODUCT_MANAGER, CUSTOMER_MANAGER et BILLING_MANAGER)	9
• Créer quelques utilisateurs, Affecter les rôles aux utilisateurs	9
• Personnaliser le paramétrage des timeouts des tokens	10
4. SECURITE L'ENSEMBLE DES MICRO-SERVICES FONCTIONNELS EN MODE BEARER-ONLY	II
• Configuration real pour application.properties	11
• Keycloak configuration	11
5. DEVELOPPER UNE APPLICATION WEB FRONT END	II
• ROLE_CUSTOMER_SERVICE	12
• Configuration keycloak	12
• Home page	13
• Login page	13
• Home apres sign-in par un customer utilisateur	14
• Résultat de la consolation de la page produit par un customer	14
• Page customer	15
• Ajouter des produits au panier	15
• Générer et imprimer la facture	16

•	ROLE_PRODUCT_SERVICE	16
•	Interface product.....	16
•	Ajouter un produit 5.....	17
•	Modifié les informations de produit 1	17
•	Supprimé le produit 3.....	17
•	ROLE_ADMIN.....	18
•	Consulter toutes les factures.....	18
6.	SECURISER L'APPLICATION FRONT END EN MODE PUBLIC CLIENT	
	19	
7.	PERSONNALISER LA SECURITE DE LA PARTIE FRONTEND	20
•	Auto-inscription des utilisateurs	20
•	Politique des mots de passe	20
•	Double authentification OTP	21
8.	MISE EN PLACE D'UNE SOLUTION DE MESSAGERIE ASYNCHRONE	
	AVEC LE BROKER KAFKA.....	22
i.	Mettre en place le Broker KAFKA	22
ii.	Broker KAFKA permet d'envoyer à un tompic « FACTURATION »	22
•	Configuration application.properties de Producer	22
•	La Création des factures aléatoirement et de les envoyés au Broker Kafka chaque seconde.....	23
•	Run Producer	24
iii.	Consommer les messages du Topic « FACTURATION ».....	25
•	Consumer Deserializer.....	25
•	Lire les messages et les enregistrer dans BD et dans un fichier CSV.....	25
•	Base de données.....	26
•	Fichier .txt.....	27
•	Une API REST qui permet de consulter les factures	28
9.	LIEN DE GITHUB :.....	29

1. 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.

1. Les outils de développement :

- **Back-end**

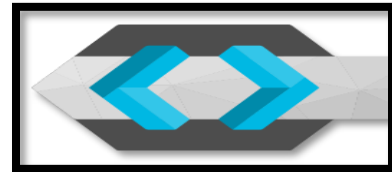
Spring boot : Spring est un framework open source pour construire et définir l'infrastructure d'une application Java3, dont il facilite le développement et les tests.



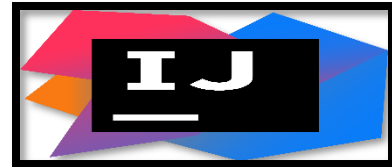
KAFKA : est une plateforme de streaming d'événements distribués open source utilisée par des milliers d'entreprises pour les pipelines de données hautes performances, l'analyse de streaming, l'intégration de données et les applications critiques.



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.

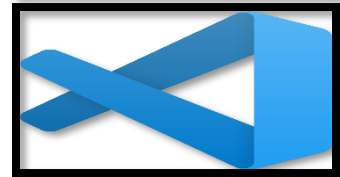


- **Front-end**

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.



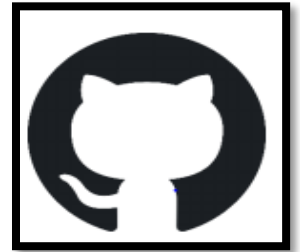
- **Base de données**

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.

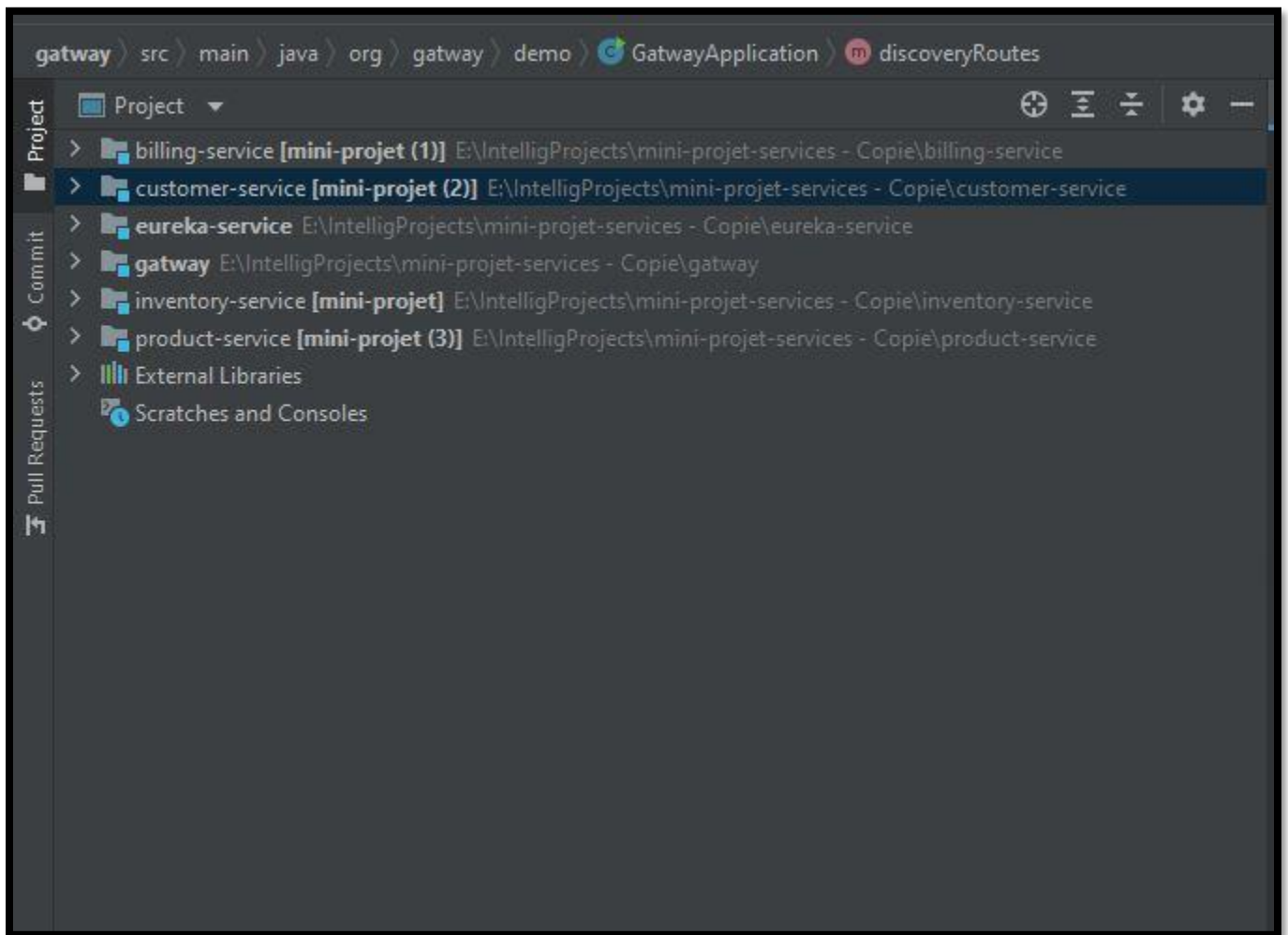


Git Bash : Git Bash est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. En 2016, il s'agit du logiciel de gestion de versions le plus populaire qui est utilisé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



2. 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



3. Mise en place du service de Sécurité avec Keycloak

- Mettre en place le serveur d'authentification OAuth2 Keycloak version 12.0.1

```
Mohamed@DESKTOP-4M318GB MINGW64 /e/IntelligProjects/keycloak-18.0.0 (main)
$ bin/standalone.bat
Calling "E:\IntelligProjects\keycloak-18.0.0\bin\standalone.conf.bat"
Setting JAVA property to "C:\Program Files\Java\jdk1.8.0_111\bin\java"
=====

JBoss Bootstrap Environment

JBOSS_HOME: "E:\IntelligProjects\keycloak-18.0.0"

JAVA: "C:\Program Files\Java\jdk1.8.0_111\bin\java"

JAVA_OPTS: "-server -Dprogram.name=standalone.bat -Xms64M -Xmx512M -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true "
=====

11:16:14,750 INFO [org.jboss.modules] (main) JBoss Modules version 2.0.2.Final
11:16:17,157 INFO [org.jboss.msc] (main) JBoss MSC version 1.4.13.Final
11:16:17,172 INFO [org.jboss.threads] (main) JBoss Threads version 2.4.0.Final
11:16:17,401 INFO [org.jboss.as] (MSC service thread 1-2) WFLYSRV0049: Keycloak 18.0.0 (WildFly Core 18.1.0.Final) starting
11:16:21,597 INFO [org.wildfly.security] (ServerService Thread Pool -- 19) ELY0001: WildFly Elytron version 1.19.0.Final
11:16:24,982 INFO [org.jboss.as.server] (Controller Boot Thread) WFLYSRV0039: Creating http management service using socket-binding (management-http)
```


- **Créer un Realm**

My-project-realm

General Login Keys Email Themes Localization Cache Tokens Client Registration Client Policies

* Name my-project-realm

Display name

HTML Display name

Frontend URL

Enabled ON

User-Managed Access OFF

Endpoints OpenID Endpoint Configuration
SAML 2.0 Identity Provider Metadata

Save Cancel

- **Le client à sécuriser en mode public client**

Settings Keys Roles Client Scopes Mappers Scope Revocation Sessions Offline Access Installation

Client ID factures-app

Name

Description

Enabled ON

Always Display in Console OFF

Consent Required OFF

Login Theme

Client Protocol openid-connect

Access Type public

Standard Flow Enabled ON

Implicit Flow Enabled OFF

Direct Access Grants Enabled ON

- Créer les rôles (USER, ADMIN, PRODUCT_MANAGER, CUSTOMER_MANAGER et BILLING_MANAGER)

Realm Roles		Default Roles	
Search...		View all roles	
		Add Role	
Role Name	Composite	Description	Actions
ROLE_ADMIN	False		Edit Delete
ROLE_BILLING_MANAGER	False		Edit Delete
ROLE_CUSTOMER_MANAGER	False		Edit Delete
ROLE_PRODUCT_MANAGER	False		Edit Delete
ROLE_USER	False		Edit Delete
default-roles-my-projet-realm	True	\$(role_default-roles)	Edit Delete
offline_access	False	\$(role_offline-access)	Edit Delete
uma_authorization	False	\$(role_uma_authorization)	Edit Delete

- Créer quelques utilisateurs, Affecter les rôles aux utilisateurs

Search...		View all users		Unlock users		Add user	
ID	Username	Email	Last Name	First Name	Actions		
0fb8ce1-7dcc-4ed4-ad0a-68...	m.elidrissi	elidrissi.mohamed.2000@gm...	IDRISSI	MOHAMED	Edit	Impersonate	Delete
8c7f4eb6-f512-47e4-85fc-cbc...	r.admin	admin@gmail.com	IDRISSI	MOHAMED	Edit	Impersonate	Delete
2c3656b2-e938-4630-865f-ad...	r.billing	billing@gmail.com	IDRISSI	MOHAMED	Edit	Impersonate	Delete
62496c2a-3e53-4391-b310-b...	r.customer	customer@gmail.com	EL IDRISSI	MOHAMED	Edit	Impersonate	Delete
ec533956-150f-47c8-b239-78...	r.customer1	customer1@gmail.com	1	customer	Edit	Impersonate	Delete
5127d9d-b195-472d-9ae4-8...	r.customer2	customer2@gmail.com	2	customer	Edit	Impersonate	Delete
edfa308a-5443-44f3-9057-ce...	r.product	product@gmail.com	IDRISSI	MOHAMED	Edit	Impersonate	Delete
af4e4038-fcec-4582-9ee8-27...	r.user	user@gmail.com	IDRISSI	MOHAMED	Edit	Impersonate	Delete

- **Personnaliser le paramétrage des timeouts des tokens**

Default Signature Algorithm ?	RS256	
Revoke Refresh Token ?	<input type="checkbox"/> OFF	
SSO Session Idle ?	30	Minutes
SSO Session Max ?	10	Hours
SSO Session Idle Remember Me ?	0	Minutes
SSO Session Max Remember Me ?	0	Minutes
Offline Session Idle ?	30	Days
Offline Session Max Limited ?	<input type="checkbox"/> OFF	
Client Session Idle ?	0	Minutes
Client Session Max ?	0	Minutes
Access Token Lifespan ?	20	Minutes
Access Token Lifespan For Implicit Flow ?	30	Minutes

➔ Refresh token: 30min

➔ Access token: 20min

4. Sécurité l'ensemble des micro-services fonctionnels en mode Bearer-Only

- Configuration real pour application.proprties

```
13 keycloak.realm=my-project-realm
14 keycloak.auth-server-url=http://localhost:8080/auth/
15 keycloak.resource=factures-app
16 keycloak.bearer-only=true
```

- Keycloak configuration

```
12 @KeycloakConfiguration
13 public class KeycloakSpringSecurityConfig extends KeycloakWebSecurityConfigurerAdapter {
14
15     @Override
16     protected SessionAuthenticationStrategy sessionAuthenticationStrategy() {
17         return new RegisterSessionAuthenticationStrategy(new SessionRegistryImpl());
18     }
19
20     @Override
21     protected void configure(AuthenticationManagerBuilder auth) throws Exception {
22         auth.authenticationProvider(keycloakAuthenticationProvider());
23     }
24
25     @Override
26     protected void configure(HttpSecurity http) throws Exception {
27         super.configure(http);
28         http
29             .authorizeHttpRequests()
30             .antMatchers("/*").hasAnyAuthority("PRODUCT_MANAGER", "BILLING_MANAGER", "CUSTOMER_MANAGER");
31     }
32 }
```

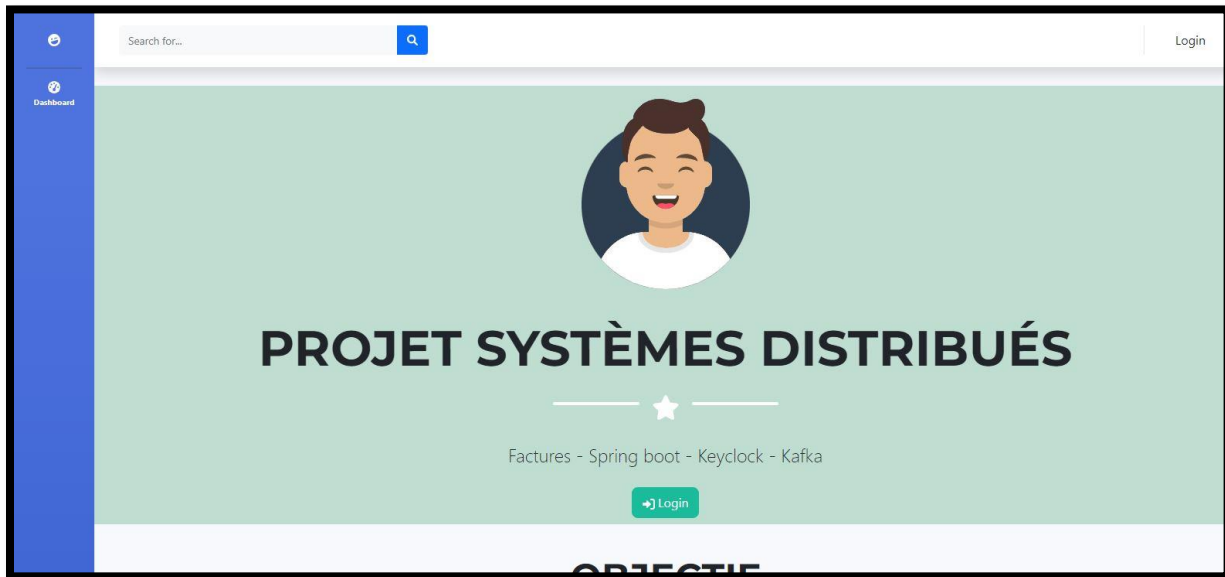
5. Développer une application Web Front End

Pour l'application web j'ai permis d'utiliser le framework de javascript Angular.

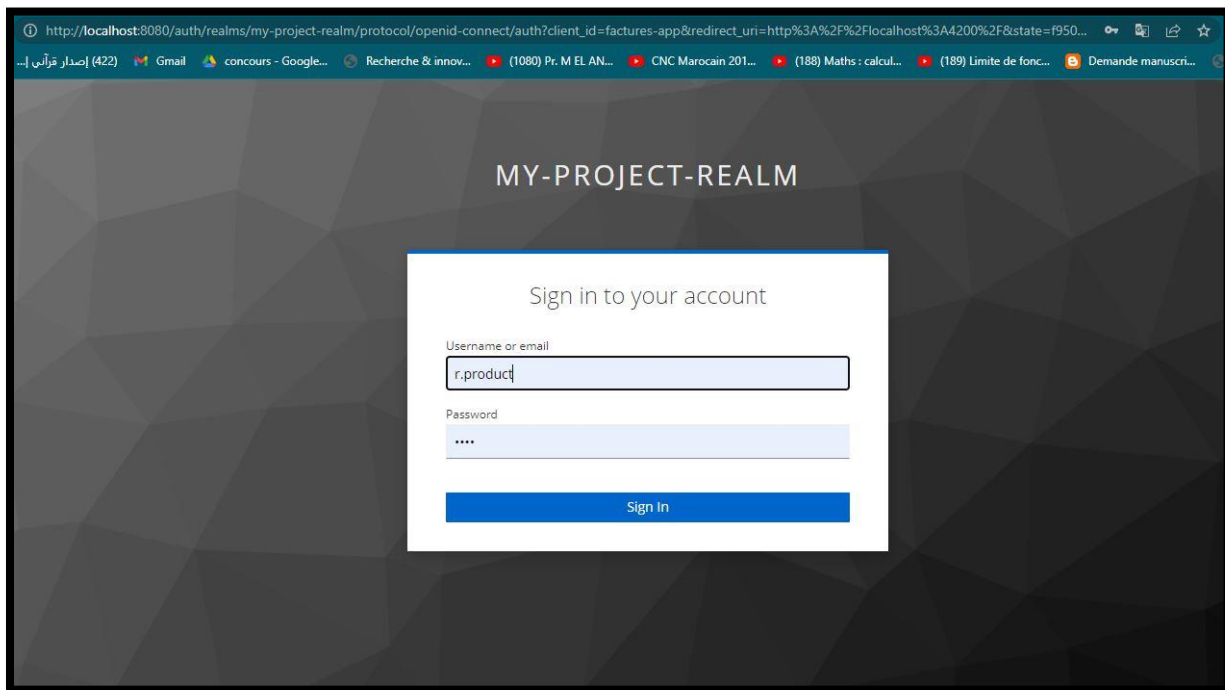
- **ROLE_CUSTOMER_SERVICE**
- Configuration keycloak

```
src > app > services > TS keycloak-security.service.ts > KeycloakSecurityService > init
1  import { Injectable } from '@angular/core';
2  import { KeycloakInstance } from 'keycloak-js';
3  declare var Keycloak:any;
4
5  @Injectable({
6    providedIn: 'root'
7  })
8  export class KeycloakSecurityService {
9    public kc:any;
10   constructor() { }
11
12   public async init(){
13     console.log("Init keycloak service")
14     this.kc = new Keycloak({
15       url:"http://localhost:8080/auth",
16       realm:"my-project-realm",
17       clientId:"factures-app"
18     });
19     await this.kc.init({
20       //onLoad:'login-required'
21       onLoad:"check-sso",
22     });
23     console.log(this.kc.token)
24   }
25 }
26
```

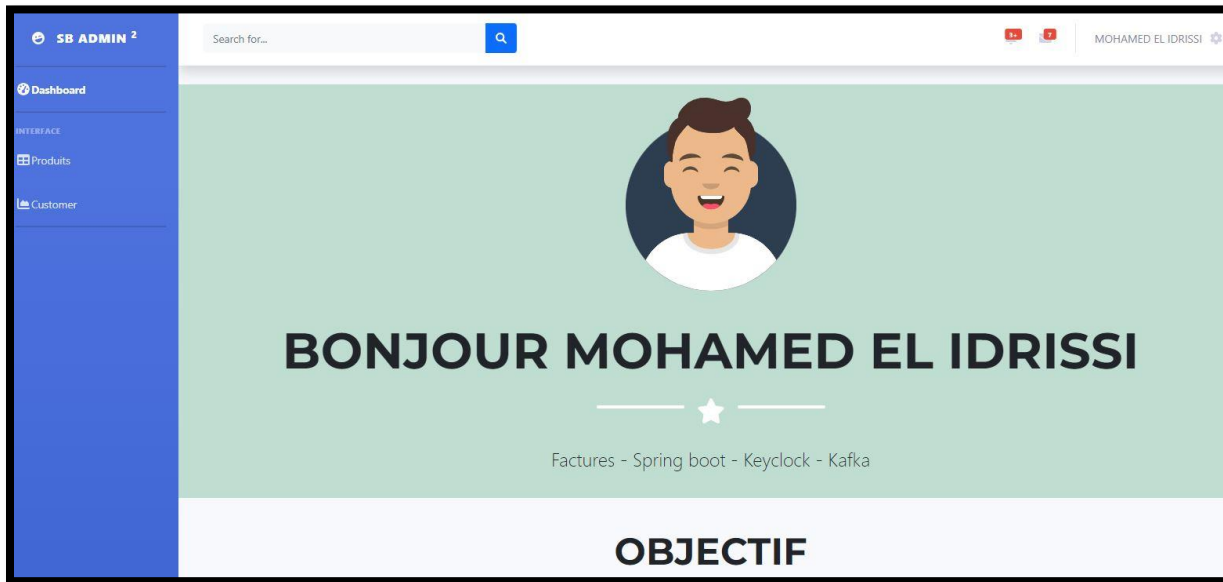
- Home page



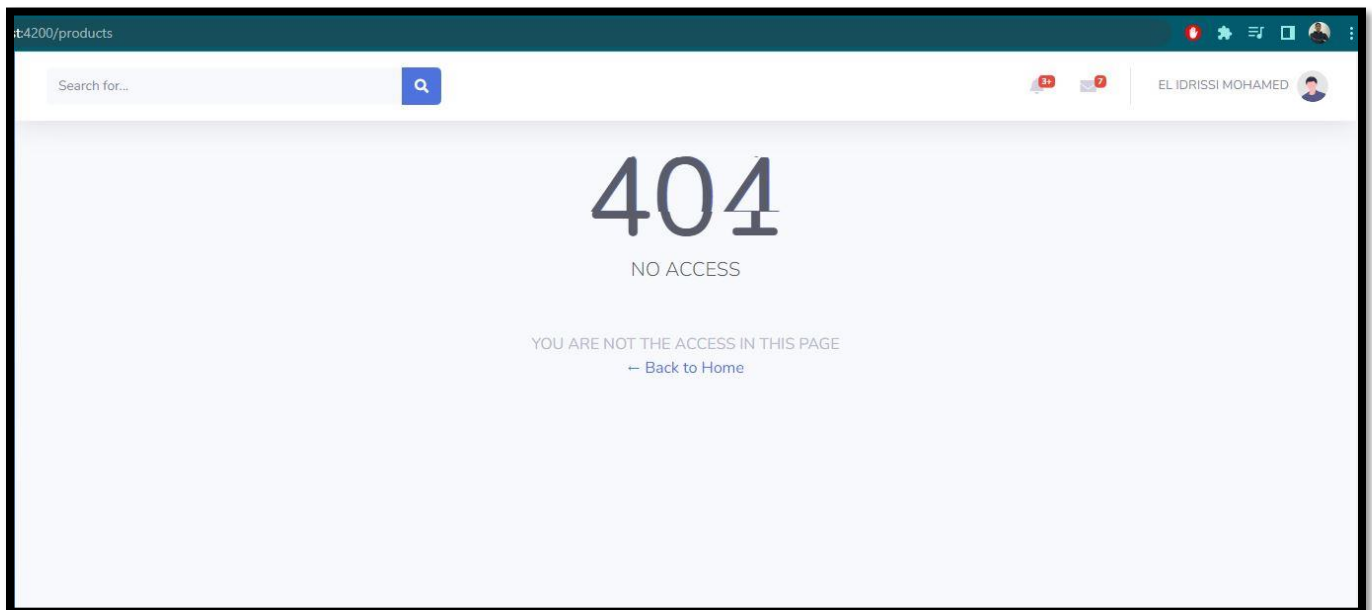
- Login page



- 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

Customer Page

Produit

Qte disponible

Prix

Qte demander

Prix total

Submit form

Produits Stock

Search:

Produit	Qte	Prix totale
<div>Acheter</div>		

- Ajouter des produits au panier

Customer Page

Produit

Qte disponible

Prix



Qte demander

Prix total

Submit form

Produits Stock

Search:

Produit	Qte	Prix totale	
product-1	3	152.82	
product-2	4	123.76	

Acheter

➔ Après l'achat la quantité de chaque produit a été modifiée

- Générer et imprimer la facture

Facture

Bienvenue MOHAMED EL IDRISSI

Search:

Produits	Qte Demande	Prix unitaire	Totale
product-1	100	500.94	50094
product-2	300	30.94	9282
product-4	100	99.64	9964
product-5	101	900	90900

Imprimer

- **ROLE_PRODUCT_SERVICE**

- Interface product

Tables

Libele

Prix

Quantite

Libele

Prix

Qte

Description

Date

Description

10/06/2022









Voir toutes les produits

Cancel











Submit form

Liste des produits











Search:

Libele	Prix	qte		
product-1	50.94	1000		
product-2	30.94	2300		
product-3	230.94	4500		
product-4	99.64	500		






- Ajouter un produit 5

Liste des produits				
			Search: <input type="text" value="Search.."/>	
Libele	Prix	qte		
product-1	50.94	1000		
product-2	30.94	2300		
product-3	230.94	4500		
product-4	99.64	500		
product-5	900	901		

- Modifié les informations de produit 1

Liste des produits				
			Search: <input type="text" value="Search.."/>	
Libele	Prix	qte		
product-1	500.94	2000		
product-2	30.94	2300		
product-3	230.94	4500		
product-4	99.64	500		
product-5	900	901		

- Supprimé le produit 3

Liste des produits				
			Search: <input type="text" value="Search.."/>	
Libele	Prix	qte		
product-1	500.94	2000		
product-2	30.94	2300		
product-4	99.64	500		
product-5	900	901		

- **ROLE_ADMIN**
- **Consulter toutes les factures**

Search for...

MOHAMED IDRISSE

Search: Search..

Tables

CUSTOMER-2 \$33.43	CUSTOMER-8 \$5,994.239	CUSTOMER-2 \$2,234.63	CUSTOMER-5 \$1,866.372
CUSTOMER-5 \$9,251.728	CUSTOMER-4 \$4,835.408	CUSTOMER-7 \$8,768.312	CUSTOMER-2 \$6,467.3
CUSTOMER-6 \$9,044.83	CUSTOMER-3 \$345.167	CUSTOMER-7 \$8,297.291	CUSTOMER-9 \$880.581
CUSTOMER-6 \$7,023.014	CUSTOMER-9 \$3,074.343	CUSTOMER-7 \$8,125.889	CUSTOMER-2 \$7,484.099
CUSTOMER-9 \$7,564.711	CUSTOMER-6 \$3,056.433	CUSTOMER-6 \$4,996.342	CUSTOMER-1 \$5,470.355

6. Sécuriser l'application Front end en mode public client

```
src > app > services > TS keycloak-security.service.ts > KeycloakSecurityService > init
1  import { Injectable } from '@angular/core';
2  import { KeycloakInstance } from 'keycloak-js';
3  declare var Keycloak:any;
4
5  @Injectable({
6    providedIn: 'root'
7  })
8  export class KeycloakSecurityService {
9    public kc:any;
10   constructor() { }
11
12   public async init(){
13     console.log("Init keycloak service")
14     this.kc = new Keycloak({
15       url:"http://localhost:8080/auth",
16       realm:"my-project-realm",
17       clientId:"factures-app"
18     });
19     await this.kc.init({
20       //onLoad:'login-required'
21       onLoad:"check-sso",
22     });
23     console.log(this.kc.token)
24   }
25 }
26
```

7. Personnaliser la sécurité de la partie frontend

- Auto-inscription des utilisateurs

The screenshot shows the 'Login' tab in a configuration interface. It contains several toggle switches and a dropdown menu for user registration settings.

Setting	Value
User registration	ON
Email as username	OFF
Edit username	ON
Forgot password	ON
Remember Me	ON
Verify email	OFF
Login with email	ON
Require SSL	external requests
ACR to LoA Mapping	ACR LOA

Buttons: Save, Cancel

- Politique des mots de passe

The screenshot shows the 'Password Policy' tab in a configuration interface. It displays a table of policy rules with columns for Policy Type, Policy Value, and Actions.

Policy Type	Policy Value	Actions
Not Username		Delete
Minimum Length	4	Delete
Maximum Length	20	Delete
Hashing Algorithm	pbkdf2-sha256	Delete
Not Email		Delete


Buttons: Save, Cancel

- **Double authentication OTP**

Mobile Authenticator Setup

You need to set up Mobile Authenticator to activate your account.

1. Install one of the following applications on your mobile:
 - FreeOTP
 - Google Authenticator
2. Open the application and scan the barcode:



[Unable to scan?](#)
3. Enter the one-time code provided by the application and click Submit to finish the setup.

Provide a Device Name to help you manage your OTP devices.

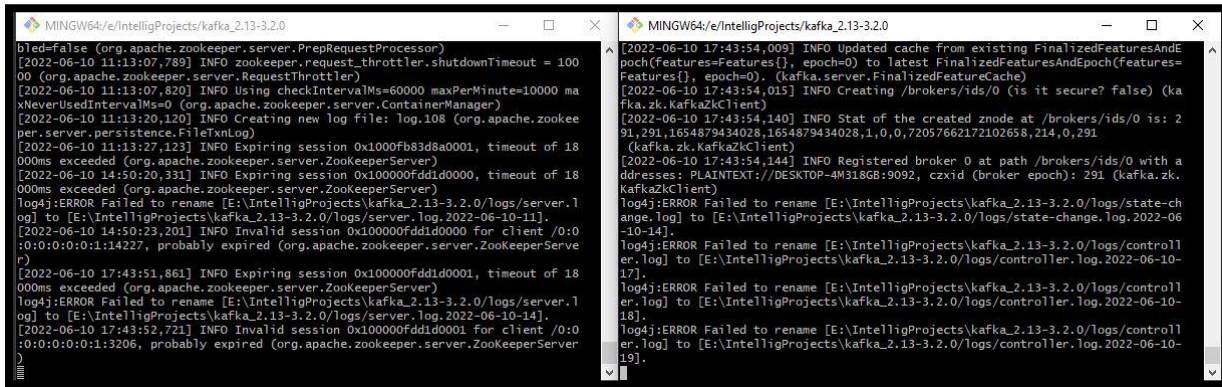
One-time code *

Device Name

Submit

8. Mise en place d'une solution de messagerie asynchrone avec le Broker KAFKA

i. Mettre en place le Broker KAFKA

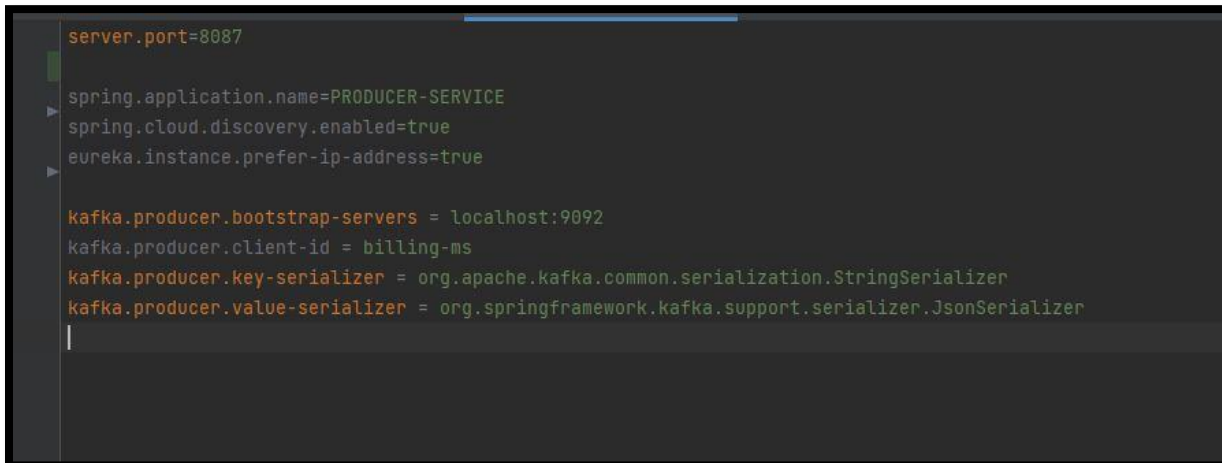


```
MINGW64: e:\IntelligProjects\kafka_2.13-3.2.0
bied=false (org.apache.zookeeper.server.PreRequestProcessor)
[2022-06-10 11:13:07,789] INFO zookeeper.request_throttler.shutdownTimeout = 100
00 (org.apache.zookeeper.server.RequestThrottler)
[2022-06-10 11:13:07,820] INFO Using checkIntervalMs=60000 maxPerMinute=10000 ma
xNeverUsedIntervalMs=0 (org.apache.zookeeper.server.ContainerManager)
[2022-06-10 11:13:20,120] INFO Creating new log file: log.108 (org.apache.zooke
per.server.persistence.FileTxnLog)
[2022-06-10 11:13:27,123] INFO Expiring session 0x1000fb83d8a0001, timeout of 18
000ms exceeded (org.apache.zookeeper.server.ZooKeeperServer)
[2022-06-10 14:50:20,331] INFO Expiring session 0x100000fdd1d0000, timeout of 18
000ms exceeded (org.apache.zookeeper.server.ZooKeeperServer)
log4j:ERROR Failed to rename [E:\IntelligProjects\kafka_2.13-3.2.0\logs\server.l
og] to [E:\IntelligProjects\kafka_2.13-3.2.0\logs\server.log.2022-06-10-11].
[2022-06-10 14:50:23,201] INFO Invalid session 0x100000fdd1d0000 for client /0:0
:0:0:0:0:1:14227, probably expired (org.apache.zookeeper.server.ZooKeeperServe
r)
[2022-06-10 17:43:51,861] INFO Expiring session 0x100000fdd1d0001, timeout of 18
000ms exceeded (org.apache.zookeeper.server.ZooKeeperServer)
log4j:ERROR Failed to rename [E:\IntelligProjects\kafka_2.13-3.2.0\logs\server.l
og] to [E:\IntelligProjects\kafka_2.13-3.2.0\logs\server.log.2022-06-10-14].
[2022-06-10 17:43:52,721] INFO Invalid session 0x100000fdd1d0001 for client /0:0
:0:0:0:0:1:3206, probably expired (org.apache.zookeeper.server.ZooKeeperServer
)

MINGW64: e:\IntelligProjects\kafka_2.13-3.2.0
[2022-06-10 17:43:54,009] INFO Updated cache from existing FinalizedFeaturesAndE
poch(features=Features[], epoch=0) to latest FinalizedFeaturesAndEpoch(features=
Features[], epoch=0). (kafka.server.FinalizedFeatureCache)
[2022-06-10 17:43:54,015] INFO Creating /brokers/ids/0 (is it secure? false) (ka
fka.zk.KafkaZkClient)
[2022-06-10 17:43:54,140] INFO Stat of the created znode at /brokers/ids/0 is: 2
91,291,1654879434028,1654879434028,1,0,0,72057662172102658,214,0,291
(kafka.zk.KafkaZkClient)
[2022-06-10 17:43:54,144] INFO Registered broker 0 at path /brokers/ids/0 with a
ddresses: PLAINTEXT://DESKTOP-4M318GB:9092, czxid (broker epoch): 291 (kafka.zk.
KafkaZkClient)
log4j:ERROR Failed to rename [E:\IntelligProjects\kafka_2.13-3.2.0\logs\state-ch
ange.log] to [E:\IntelligProjects\kafka_2.13-3.2.0\logs\state-change.log.2022-06
-10-14].
log4j:ERROR Failed to rename [E:\IntelligProjects\kafka_2.13-3.2.0\logs\controlle
r.log] to [E:\IntelligProjects\kafka_2.13-3.2.0\logs\controller.log.2022-06-10-
17].
log4j:ERROR Failed to rename [E:\IntelligProjects\kafka_2.13-3.2.0\logs\controlle
r.log] to [E:\IntelligProjects\kafka_2.13-3.2.0\logs\controller.log.2022-06-10-
18].
log4j:ERROR Failed to rename [E:\IntelligProjects\kafka_2.13-3.2.0\logs\controlle
r.log] to [E:\IntelligProjects\kafka_2.13-3.2.0\logs\controller.log.2022-06-10-
19].
```

ii. Broker KAFKA permet d'envoyer à un topic « FACTURATION »

- Configuration application.properties de Producer



```
server.port=8087

spring.application.name=PRODUCER-SERVICE
spring.cloud.discovery.enabled=true
eureka.instance.prefer-ip-address=true

kafka.producer.bootstrap-servers = localhost:9092
kafka.producer.client-id = billing-ms
kafka.producer.key-serializer = org.apache.kafka.common.serialization.StringSerializer
kafka.producer.value-serializer = org.springframework.kafka.support.serializer.JsonSerializer
```

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

```
14  @Service
15  @Transactional
16  @Slf4j
17  @EnableScheduling
18  public class BillServiceImpl implements BillService {
19
20
21      private final KafkaTemplate<String, BillDTO> template;
22
23      List<String> customers = Arrays.asList("customer-1", "customer-2", "customer-3", "customer-4",
24      "customer-5", "customer-6", "customer-7", "customer-8", "customer-9", "customer-10");
25
26      public BillServiceImpl(KafkaTemplate<String, BillDTO> template) {
27          this.template = template;
28      }
29
30      @Scheduled(cron = "*/1 * * * *")
31      public void sendBillsToKafka () {
32          BillDTO bill = new BillDTO();
33          bill.setId((long) (Math.random() * 999999999));
34          bill.setCustomer(customers.get((int) (Math.random() * 10)));
35          bill.setPrice(Math.random() * 9999);
36          System.out.println(bill);
37          template.send( topic, "FACTURATION", bill);
38      }
39  }
40
```


- Run Producer

The screenshot shows the IDE with the `ProducerServiceApplication` class open. The code is a Spring Boot application that uses a `@Scheduled` annotation to trigger a `sendBillsToKafka` method. This method generates random `BillDTO` objects and sends them to a Kafka topic named `FACTURATION` with a key derived from the bill's class.

```

29
30 @Scheduled(cron = "*/1 * * * *")
31 public void sendBillsToKafka () {
32     BillDTO bill = new BillDTO();
33     bill.setId((Long) (Math.random() * 999999999));
34     bill.setCustomer(customers.get((int) (Math.random() * 10)));
35     bill.setPrice(Math.random() * 9999);
36     System.out.println(bill);
37     template.send(topic, "FACTURATION", key: "key"+bill.getClass(), bill);
38 }
39

```

The output window shows a series of `BillDTO` objects being printed, each with a unique ID, customer name, and price.

```

BillDTO(id=588949370, customer=customer-10, price=2556.7732204028644)
BillDTO(id=476529709, customer=customer-9, price=8107.730514046063)
BillDTO(id=302286016, customer=customer-1, price=9047.813087018554)
BillDTO(id=767018352, customer=customer-7, price=3649.012078254935)
BillDTO(id=910595027, customer=customer-8, price=6903.16450220952)
BillDTO(id=116110794, customer=customer-6, price=2830.7434607818223)
BillDTO(id=771479501, customer=customer-10, price=9334.223890387228)
BillDTO(id=64977750, customer=customer-4, price=2054.685048846794)
BillDTO(id=348420680, customer=customer-6, price=6356.539698169901)
BillDTO(id=389299407, customer=customer-10, price=234.68087721812455)
BillDTO(id=567578930, customer=customer-10, price=7648.713571954129)
BillDTO(id=859422533, customer=customer-3, price=5976.961311721051)
BillDTO(id=516557750, customer=customer-1, price=1843.0960622073544)
BillDTO(id=440426826, customer=customer-8, price=4880.032917923285)
BillDTO(id=790764679, customer=customer-5, price=5065.89589673414)
BillDTO(id=932796796, customer=customer-5, price=2171.55766844228)
BillDTO(id=501534137, customer=customer-10, price=6630.590515006536)
BillDTO(id=821245902, customer=customer-6, price=4091.99377999519)
BillDTO(id=506292477, customer=customer-7, price=7970.271835446302)

```

The screenshot shows the IDE with the `ConsumerServiceApplication` class open. The code is a Spring Boot application that receives messages from a Kafka topic named `FACTURATION`. It uses a `BillMapper` to convert the received `BillDTO` into a `Bill` object, which is then saved to a database using a `BillRepository`.

```

36
37
38
39
40
41
42
43
44
45
46
47

```

The output window shows the application's logs, including Hibernate SQL statements and the received `BillDTO` objects.

```

Hibernate: insert into bill (customer, price) values (?, ?)
Received ==> BillDTO(id=654279542, customer=customer-8, price=9677.317898208124)
Hibernate: select bill0_.id as id1_0_0_, bill0_.customer as customer2_0_0_, bill0_.price as price3_0_0_ from bill bill0_ where bill0_.id=?
Hibernate: insert into bill (customer, price) values (?, ?)
Received ==> BillDTO(id=491521101, customer=customer-4, price=6616.671093292444)
Hibernate: select bill0_.id as id1_0_0_, bill0_.customer as customer2_0_0_, bill0_.price as price3_0_0_ from bill bill0_ where bill0_.id=?
Hibernate: insert into bill (customer, price) values (?, ?)
Received ==> BillDTO(id=794725251, customer=customer-8, price=4486.3361493185075)
Hibernate: select bill0_.id as id1_0_0_, bill0_.customer as customer2_0_0_, bill0_.price as price3_0_0_ from bill bill0_ where bill0_.id=?
Hibernate: insert into bill (customer, price) values (?, ?)
Received ==> BillDTO(id=806310176, customer=customer-7, price=9008.06749441832)
Hibernate: select bill0_.id as id1_0_0_, bill0_.customer as customer2_0_0_, bill0_.price as price3_0_0_ from bill bill0_ where bill0_.id=?
Received ==> BillDTO(id=732407813, customer=customer-5, price=7092.837342880395)
Hibernate: select bill0_.id as id1_0_0_, bill0_.customer as customer2_0_0_, bill0_.price as price3_0_0_ from bill bill0_ where bill0_.id=?
Hibernate: insert into bill (customer, price) values (?, ?)
Received ==> BillDTO(id=57626259, customer=customer-9, price=2413.124492381548)

```

iii. Consommer les messages du Topic « FACTURATION »

- Consumer Deserializer

```
29 @ private Map<String, Object> getConsumerProperties() {
30     Map<String, Object> propertiesMap = new HashMap<>();
31     propertiesMap.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapAddress);
32     propertiesMap.put(ConsumerConfig.GROUP_ID_CONFIG, groupId);
33     propertiesMap.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, autoOffset);
34
35     return propertiesMap;
36 }
37 @Bean
38 public ConsumerFactory<String, BillDTO> defaultConsumerFactory() {
39     Map<String, Object> consumerProperties = getConsumerProperties();
40     JsonDeserializer<BillDTO> jsonDeserializer = new JsonDeserializer<>(BillDTO.class);
41     jsonDeserializer.addTrustedPackages("*");
42     jsonDeserializer.setUseTypeHeaders(false);
43     return new DefaultKafkaConsumerFactory(consumerProperties, new StringDeserializer(), jsonDeserializer);
44 }
45
46
47 @Bean
48 public ConcurrentKafkaListenerContainerFactory<String, BillDTO> kafkaListenerContainerFactory() {
49
```

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

```
34 @KafkaListener(topics = "FACTURATION", containerFactory = "kafkaListenerContainerFactory")
35 @ public void consumeImportOffreTopic(BillDTO billDTO) throws Exception {
36
37     System.out.println("Received ==> "+billDTO.toString());
38     Bill bill = billMapper.billDTOToBill(billDTO);
39     billRepository.save(bill);
40     writeBill(billDTO);
41 }
42
43 @ private void writeBill(BillDTO bill) throws FileNotFoundException {
44     log.info("Write in file");
45     File file;
46     FileWriter fr;
47     try {
48         file = new File(pathname: "bill-facturation.csv");
49         fr = new FileWriter(file, append: true);
50         BufferedWriter br = new BufferedWriter(fr);
51         br.write( str. bill.getId() + "," + bill.getCustomer() + "," + bill.getPrice()+"\r\n");
52
53         br.close();
54         fr.close();
55     } catch (IOException e) {
56         e.printStackTrace();
57     }
58 }
```

- Base de données

✓ Affichage des lignes 0 - 323 (total de 324, traitement en 0,0036 seconde(s).)

`SELECT * FROM `bill``

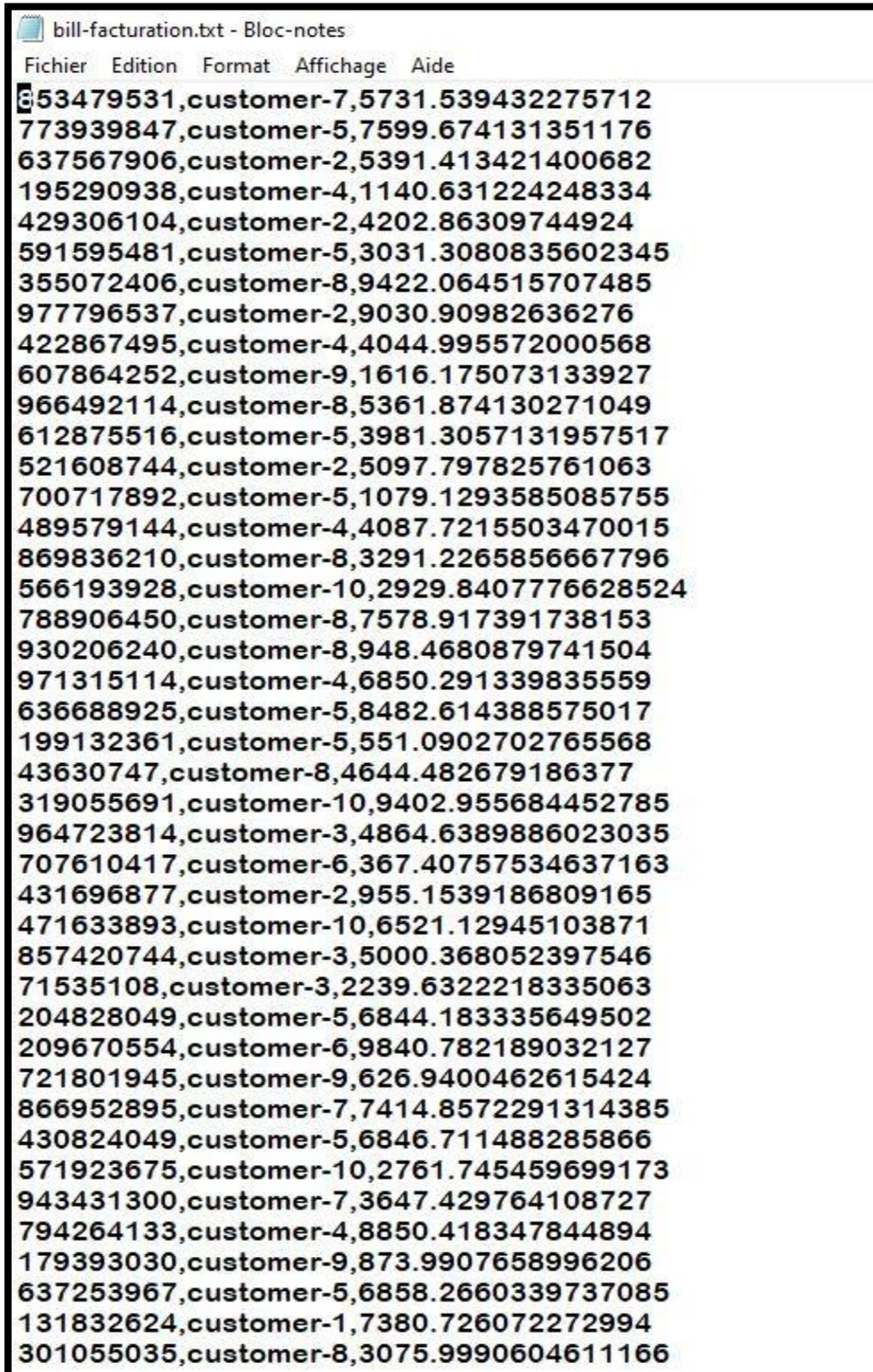
☐ Tout afficher | Nombre de lignes : 500 ▼ | Filtrer les lignes: | Trier par

+ Options

			id	customer	price	
<input type="checkbox"/>	Éditer	Copier	Supprimer	1	customer-2	33.429591731911124
<input type="checkbox"/>	Éditer	Copier	Supprimer	2	customer-8	5994.239219407287
<input type="checkbox"/>	Éditer	Copier	Supprimer	3	customer-2	2234.6295243541936
<input type="checkbox"/>	Éditer	Copier	Supprimer	4	customer-5	1866.3715119807932
<input type="checkbox"/>	Éditer	Copier	Supprimer	5	customer-5	9251.728431456231
<input type="checkbox"/>	Éditer	Copier	Supprimer	6	customer-4	4835.40792370393
<input type="checkbox"/>	Éditer	Copier	Supprimer	7	customer-7	8768.311807558995
<input type="checkbox"/>	Éditer	Copier	Supprimer	8	customer-2	6467.299536633615
<input type="checkbox"/>	Éditer	Copier	Supprimer	9	customer-6	9044.82984498522
<input type="checkbox"/>	Éditer	Copier	Supprimer	10	customer-3	345.1665492700903
<input type="checkbox"/>	Éditer	Copier	Supprimer	11	customer-7	8297.290543434081
<input type="checkbox"/>	Éditer	Copier	Supprimer	12	customer-9	880.5807789532216
<input type="checkbox"/>	Éditer	Copier	Supprimer	13	customer-6	7023.014032436534
<input type="checkbox"/>	Éditer	Copier	Supprimer	14	customer-9	3074.3425975241967
<input type="checkbox"/>	Éditer	Copier	Supprimer	15	customer-7	8125.888937815861
<input type="checkbox"/>	Éditer	Copier	Supprimer	16	customer-2	7484.0990684508015
<input checked="" type="checkbox"/>	Éditer	Copier	Supprimer	17	customer-9	7564.711220478236

Console de requêtes SQL

- Fichier .txt



```
bill-facturation.txt - Bloc-notes
Fichier Edition Format Affichage Aide
53479531,customer-7,5731.539432275712
773939847,customer-5,7599.674131351176
637567906,customer-2,5391.413421400682
195290938,customer-4,1140.631224248334
429306104,customer-2,4202.86309744924
591595481,customer-5,3031.3080835602345
355072406,customer-8,9422.064515707485
977796537,customer-2,9030.90982636276
422867495,customer-4,4044.995572000568
607864252,customer-9,1616.175073133927
966492114,customer-8,5361.874130271049
612875516,customer-5,3981.3057131957517
521608744,customer-2,5097.797825761063
700717892,customer-5,1079.1293585085755
489579144,customer-4,4087.7215503470015
869836210,customer-8,3291.2265856667796
566193928,customer-10,2929.8407776628524
788906450,customer-8,7578.917391738153
930206240,customer-8,948.4680879741504
971315114,customer-4,6850.291339835559
636688925,customer-5,8482.614388575017
199132361,customer-5,551.0902702765568
43630747,customer-8,4644.482679186377
319055691,customer-10,9402.955684452785
964723814,customer-3,4864.6389886023035
707610417,customer-6,367.40757534637163
431696877,customer-2,955.1539186809165
471633893,customer-10,6521.12945103871
857420744,customer-3,5000.368052397546
71535108,customer-3,2239.6322218335063
204828049,customer-5,6844.183335649502
209670554,customer-6,9840.782189032127
721801945,customer-9,626.9400462615424
866952895,customer-7,7414.8572291314385
430824049,customer-5,6846.711488285866
571923675,customer-10,2761.745459699173
943431300,customer-7,3647.429764108727
794264133,customer-4,8850.418347844894
179393030,customer-9,873.9907658996206
637253967,customer-5,6858.2660339737085
131832624,customer-1,7380.726072272994
301055035,customer-8,3075.9990604611166
```

- Une API REST qui permet de consulter les factures

The screenshot displays the SB ADMIN 2 dashboard. On the left is a blue sidebar with navigation links: Dashboard, Admin (with a dropdown menu showing PAGE ADMIN, Produits, and Factures), Produits, and Customer. The main content area is titled 'Tables' and contains a grid of 20 invoice cards. Each card shows a customer ID, a monetary value, and a calendar icon. A search bar is located at the top right of the main area.

Search for...		MOHAMED IDRISI	
Search: Search..			
CUSTOMER-2	\$33.43	CUSTOMER-8	\$5,994.239
CUSTOMER-5	\$9,251.728	CUSTOMER-4	\$4,835.408
CUSTOMER-6	\$9,044.83	CUSTOMER-3	\$345.167
CUSTOMER-6	\$7,023.014	CUSTOMER-9	\$3,074.343
CUSTOMER-9	\$7,564.711	CUSTOMER-6	\$3,056.433
CUSTOMER-2	\$2,234.63	CUSTOMER-7	\$8,768.312
CUSTOMER-5	\$1,866.372	CUSTOMER-7	\$8,297.291
CUSTOMER-2	\$6,467.3	CUSTOMER-9	\$880.581
CUSTOMER-2	\$7,484.099	CUSTOMER-1	\$5,470.355
CUSTOMER-6	\$4,996.342		

9. 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>