



COMPTES RENDUS

PRÉPARÉ PAR
HAFID ISMAIL

ARCHITECTURE TECHNIQUE

L'architecture technique du projet devrait respecter l'architecture Technique suivante :

1. Partie Backend :

- L'application doit être basée sur une approche Micro services en utilisant l'écosystème de Spring Cloud.

- **Micro services à Développer :**

- o **Micro-services Fonctionnels :**

- Micro-service: client , produit , billing

- o **Micro-services Techniques :**

- Un Service Gateway basé sur Spring Cloud Gateway
- Un Service Discovery basé sur Netflix Eureka Server
- Un service Configuration basé sur Netflix Config server

- **Sécurité :**

- Les Micro services doivent être sécurisés par en utilisant des adaptateurs KeyCloak

- La communication entre les micro-services doit inclure les deux modes :

- Synchrone avec REST en utilisant Open Feign ou RestTemplate
- Asynchrone en utilisant le Broker KAFKA

- **Partie Stream Processing :**

- o Développer et mettre en œuvre une solution de Big Data Analytics de Real Time Stream Processing en utilisant Kafka Streams.

- Les Micro services fonctionnels peuvent être développés avec des langages différents

2. Partie Front end Web :

- a. Application Web basée sur Framework Angular

- b. Utiliser NgRX pour la gestion du state de l'application

- c. Utiliser Json Web Token et Keycloak Adapter pour la partie sécurité

TECHNOLOGIES UTILISÉES

- **JEE (Java Enterprise Edition)** : une plateforme de développement d'applications pour les entreprises qui permet de créer des applications robustes et évolutives.
- **Spring Boot** : un cadre de développement pour créer des applications Java basées sur Spring qui facilite la configuration et la mise en place de l'application.
- **Keycloak** : un système d'authentification et d'autorisation open-source qui a été utilisé pour sécuriser l'application.
- **Kafka** : un système de messagerie distribué utilisé pour gérer les flux de données de l'application. Il a été utilisé pour gérer les commandes en temps réel et pour garantir la scalabilité de l'application.
- **Angular** : un framework de développement d'applications web open-source utilisé pour créer l'interface utilisateur de l'application.
- **Bootstrap** : un framework de design de site web utilisé pour faciliter la mise en page et la mise en forme de l'application. Il a permis de créer une interface utilisateur responsive et adaptable aux différents appareils.
- **Git** : un système de contrôle de version utilisé pour gérer les versions de l'application et les contributions des différents membres de l'équipe. Il a permis de gérer les différentes branches de développement et de faciliter la collaboration entre les développeurs.
- **Kafka** : Il est conçu pour gérer des milliers de messages par seconde et est utilisé pour les cas d'utilisation tels que la collecte de données de capteurs, la gestion des transactions en ligne et la surveillance en temps réel des données.
- **Chart.js** : Il permet de créer des graphiques tels que des barres, des lignes, des camemberts, etc. en utilisant un format de données simple et en fournissant des options de personnalisation pour les styles et les animations.
- **Docker** : est un outil logiciel open-source qui permet de créer, déployer et exécuter des applications dans des conteneurs logiciels. Les conteneurs sont des environnements isolés qui permettent de faire fonctionner une application dans un environnement prévisible, indépendamment de l'infrastructure sur laquelle elle est déployée.

Ces technologies ont été utilisées pour créer une application robuste et évolutive, qui répond aux besoins des utilisateurs et garantit un bon fonctionnement des différentes fonctionnalités.

ARCHITECTURE DE PROJET

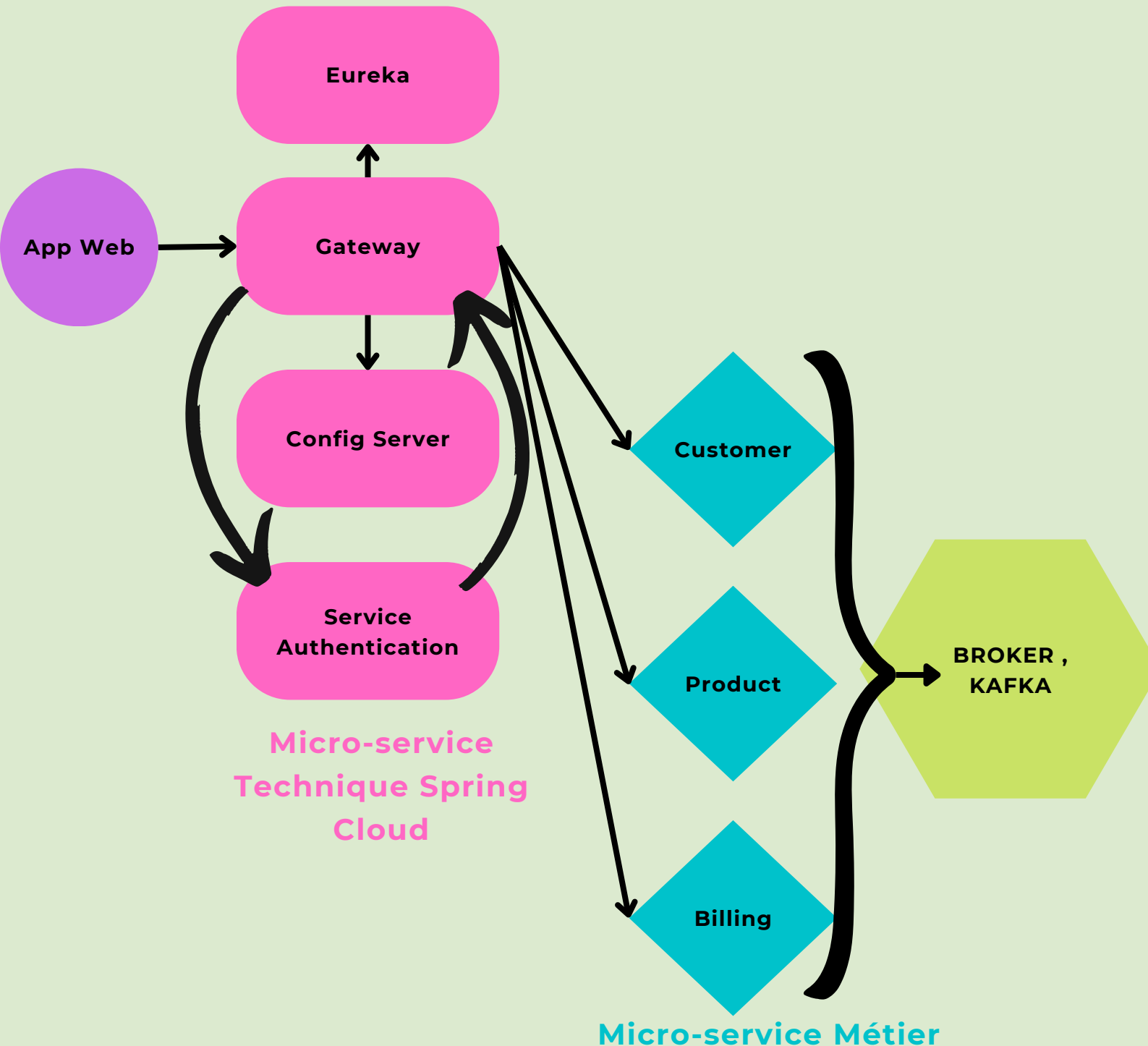
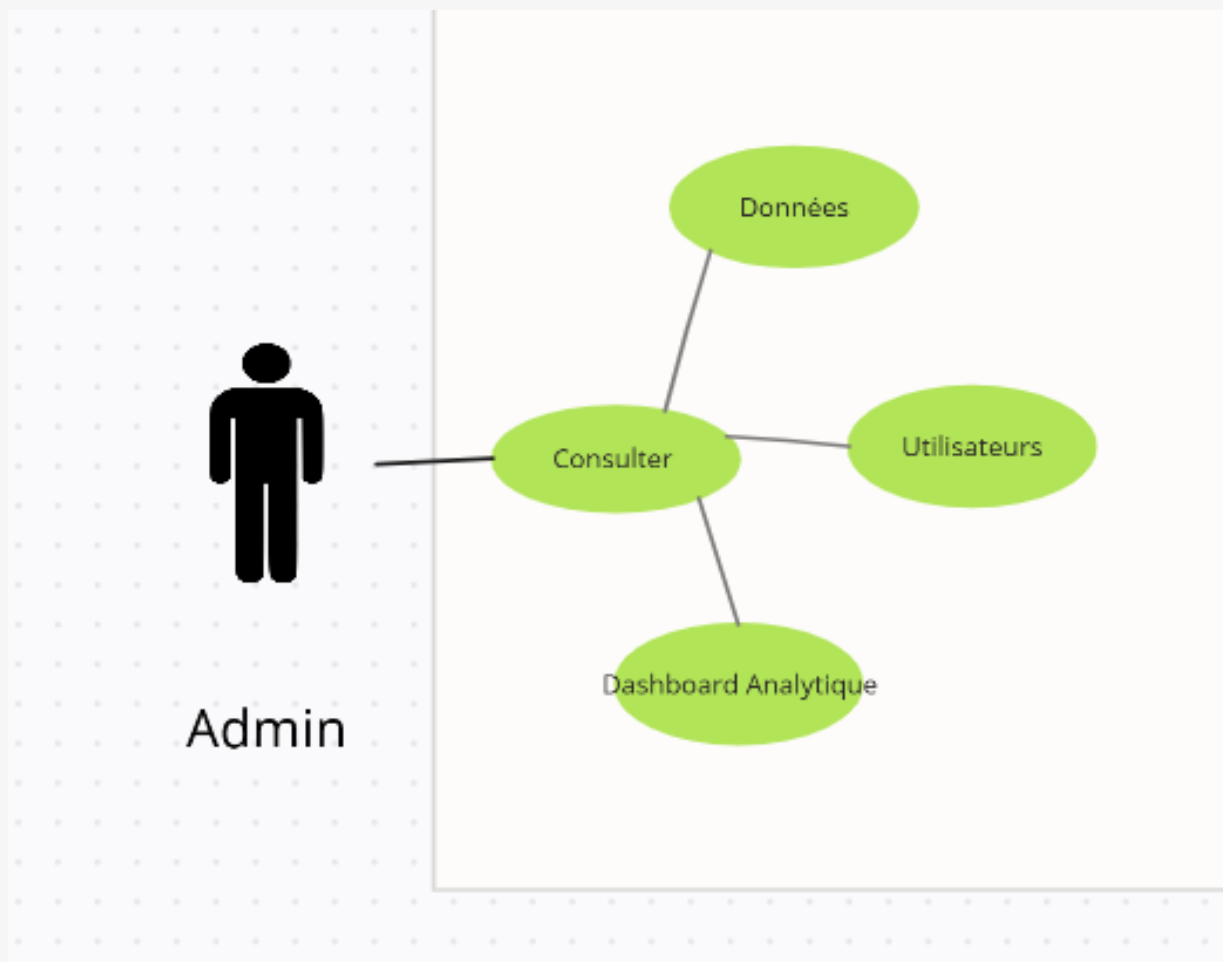
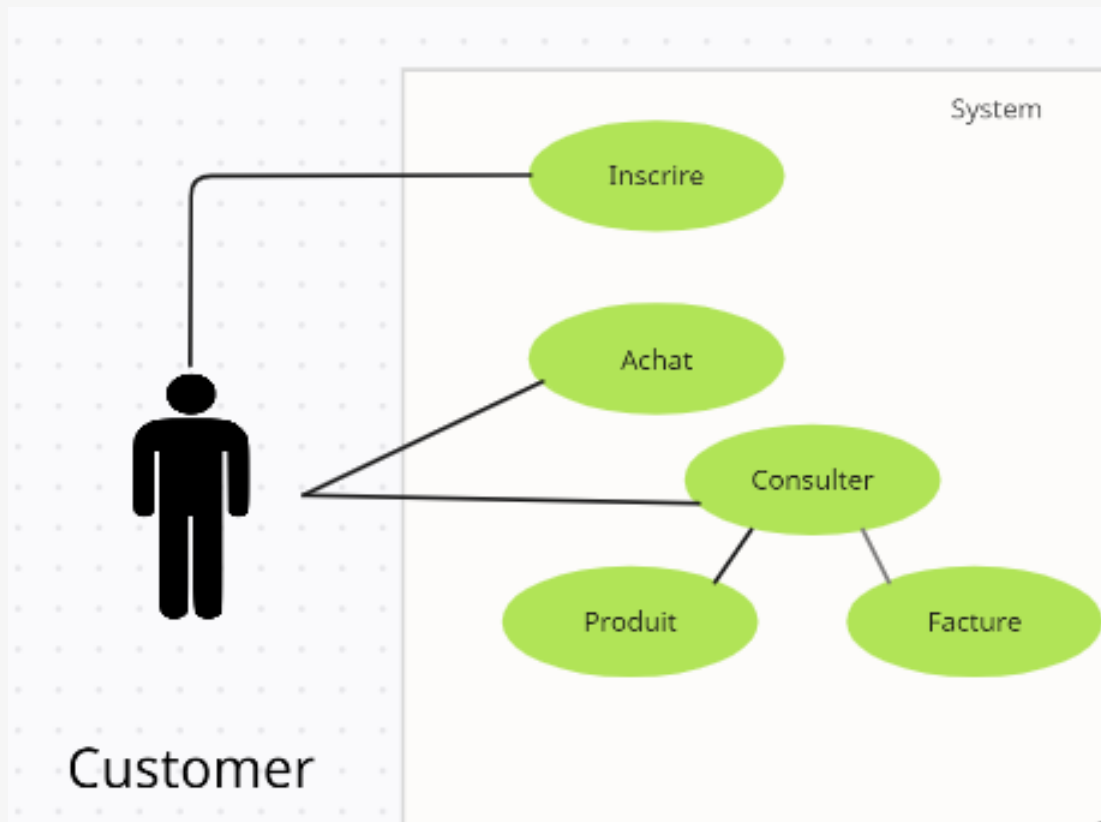


DIAGRAMME DE CAS D'UTILISATION



INTERFACE GRAPHIQUE

liste de produit:

E-COMProducts ▾Customers ▾Data Analytics

Id	Nom	Prix	Quantité	Actions
1	Stylo	1	200	<div>Supprimer</div>
2	Gomme	3	70	<div>Supprimer</div>
3	Blanco	10	30	<div>Supprimer</div>

add product:

E-COM	Products ▾	Customers ▾	Data Analytics
<div><div>New Customer</div><div><div>Name:</div><div></div><div>Price:</div><div></div><div>Quantity:</div><div></div><div>Save Product</div></div></div>			

liste de clients:

E-COM Products ▾ Customers ▾ Data Analytics

Id	Nom	Email	Orders	Actions
1	Hatim	Hatim@email.com	<button>Orders</button>	<button>Delete</button>
2	Hafid	Hafid@email.com	<button>Orders</button>	<button>Delete</button>
3	Yasser	Yasser@email.com	<button>Orders</button>	<button>Delete</button>

INTERFACE GRAPHIQUE

new customer

[E-COM](#) [Products](#) [Customers](#) [Data Analytics](#)

New Customer

Name:

Email:

Save

Notifications (add customer/ delete customer)

localhost:4200 indique
Customer saved successfully

New Customer

Name:

Email:

Save

OK

ata Analytics

localhost:4200 indique
Are you sure?

Id	Nom	Email	Orders	Actions
1	Hatim	Hatim@email.com	Orders	Delete
2	Hafid	Hafid@email.com	Orders	Delete
3	Yasser	Yasser@email.com	Orders	Delete
4	Customer 4	Customer4@email.com	Orders	Delete

INTERFACE GRAPHIQUE

liste de commande de client

Customers ▾ Data Analytics

Id	Date	CustomerId	
1	24-55-2023	1	<button>Details</button>

les details de chaque commande:

Customers ▾ Data Analytics

Bill Id: 1
Bill Id: 1
Date: 24-55-2023
Customer Id: 1
Customer Name: Hatim
Customer Mail: Hatim@email.com

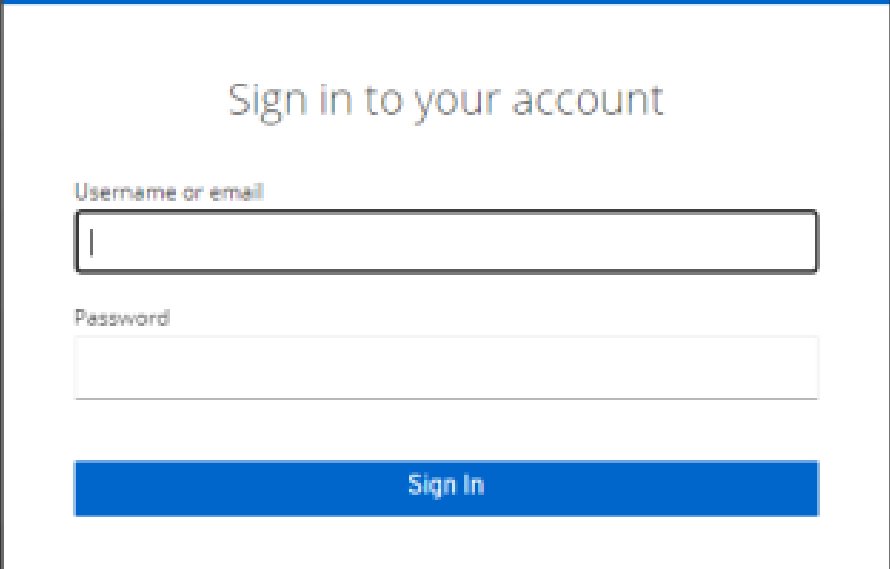
Product Items				
Product Id	Product Name	Quantity	Price	Amount
1	Stylo	39	1.00	39.00
2	Gomme	86	3.00	258.00
3	Blanco	61	10.00	610.00
			Total	907.00

Keycloak

Pour sécuriser notre front-end/back-end on a utilisé keycloak pour limiter l'accès par utilisateur non connecté et non admin.

Configuration de Keycloak pour l'application : cela inclut la configuration des rôles et des autorisations pour les utilisateurs de l'application, ainsi que la configuration des flux d'authentification et d'autorisation.

Donc avant de accéder à les données ou interface graphique , l'utilisateur doit s'authentifier pour avoir le droit de naviguer dans l'application web.

A screenshot of a Keycloak login form. The form is white with a blue border and is centered on a dark gray background with a geometric pattern. It contains the text "Sign in to your account", two input fields for "Username or email" and "Password", and a blue "Sign In" button.

Sign in to your account

Username or email

Password

Sign In


USER

[Details](#)[Attributes](#)[Users in Role](#)

Username	Last Name
ismail	hafid

Roles

[Realm Roles](#)[Default Roles](#)




[View all roles](#)

Role Name	Com
ADMIN	Fals
USER	Fals
offline_access	Fals
uma_authorization	Fals

Clients

[Lookup !\[\]\(c50c8b7b2cc2cf9ff925edec0ee94c0d_img.jpg\)](#)

<input type="text" value="Search..."/>	
Client ID	Enabled
account	True
account-console	True
admin-cli	True
billing-service	True
broker	True
customer-service	True
front-end	True
inventory-service	True
realm-management	True
security-admin-console	True

Conclusion

En conclusion, le projet d'application web en microservice développé en utilisant Spring Boot, Keycloak, Kafka, Chart.js et Docker a été un succès.

Les technologies utilisées ont permis de créer une architecture robuste et évolutive, offrant une sécurité renforcée grâce à l'intégration de Keycloak et une gestion efficace des flux de données grâce à Kafka.

De plus, l'utilisation de Chart.js a permis une visualisation claire et intuitive des données, tandis que l'utilisation de Docker a facilité le déploiement et la scalabilité de l'application.

En somme, ce projet a démontré que l'utilisation de ces technologies modernes peut permettre de créer des applications web performantes et adaptées aux besoins des utilisateurs.

