



COURS DE PROGRAMMATION AVANCEE  
EN JAVA ET MICROSERVICES

# RAPPORT DU PROJET FINAL

**Auteur :**

- KOUDOSSOU M. D. Justin

**Chargé du cours :**

M. SEWAVI Maurice

**Samedi 05 avril 2025**

UCAO – UUT / Master 1 – Cycle Ingénieur

2024-2025

## Table des matières

Avant-Propos.....	3
Résumé .....	4
I. Introduction.....	5
II. Interactions entre l'application cliente et l'API Spring boot.....	6
a) Gestion des clients .....	7
b) Gestion des produits .....	11
c) Gestion des factures.....	15
III. Conclusion .....	24

## **Avant-Propos**

Ce document de rapport présente le résultat final d'une application développée en Java avec Spring Boot, qui offre une API consommée par une autre application. L'objectif principal de ce rapport est de démontrer les interactions entre cette application cliente et l'API, mettant ainsi en lumière les fonctionnalités, les performances et la robustesse de l'architecture de microservices. Ce rapport servira de référence pour les équipes souhaitant explorer les avantages d'une telle approche dans le développement d'applications.

## Résumé

Ce rapport présente le résultat final d'une application développée en Java avec Spring Boot, qui offre une API consommée par une application cliente. Il commence par une introduction à l'architecture de microservices, suivie d'une description des fonctionnalités clés de l'API. Ensuite, il illustre les interactions entre l'application cliente et l'API, en mettant en avant les performances et la robustesse de l'architecture. Enfin, le rapport conclut par une discussion sur les bénéfices de cette approche pour le développement d'applications, servant de référence pour les équipes intéressées.

# I. Introduction

L'API est développée avec Java et Spring Boot pour créer une architecture de microservices composée de **cinq services** principaux. Le **Service Gateway** joue un rôle crucial en servant de point d'entrée unique pour toutes les requêtes, simplifiant ainsi la communication entre les différents services. En parallèle, le **Service Eureka** gère la configuration et le registre des services, assurant une gestion efficace des microservices.

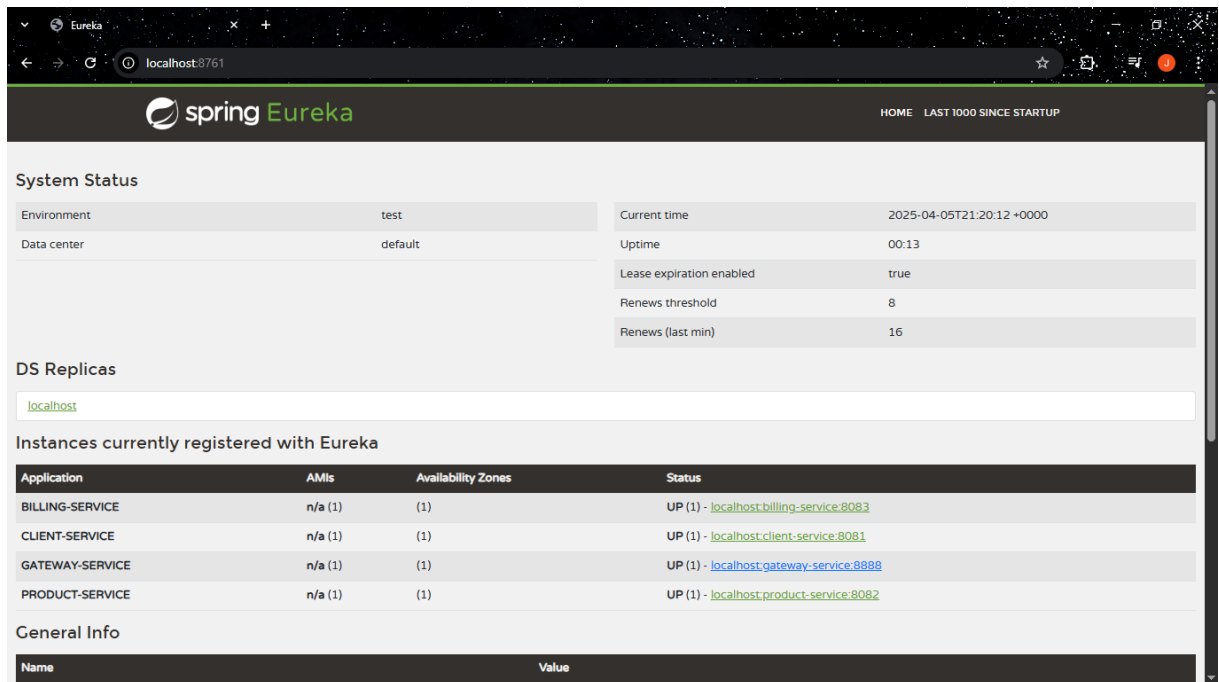
Les autres services incluent le **Service client**, dédié à la gestion des clients, le **Service produit**, qui s'occupe de la gestion des produits, et le **Service facturation**, responsable du traitement des factures. Chacun de ces services travaille de manière autonome tout en interagissant avec les autres pour offrir une expérience utilisateur fluide.

L'application cliente, développée en **Flutter** pour Android, permet aux utilisateurs de gérer facilement les clients, les produits et les factures, avec tous leurs détails. Elle interagit avec les microservices via le **Service Gateway**, ce qui facilite la gestion des données. Ce rapport mettra en lumière ces interactions et les avantages d'une telle architecture de microservices pour une gestion efficace et scalable des informations.

Codes sources disponibles sur : <https://github.com/Onzk/tp-microservice>

## II. Interactions entre l'application cliente et l'API Spring boot

Après lancement de tous les microservices, on obtient le résultat suivant sur l'interface du service de configuration.



The screenshot shows the Spring Eureka web interface in a browser. The address bar shows 'localhost:8761'. The page has a dark header with the 'spring Eureka' logo and navigation links 'HOME' and 'LAST 1000 SINCE STARTUP'. The main content area is divided into several sections:

- System Status:** A table showing environment details.

Environment	test	Current time	2025-04-05T21:20:12 +0000
Data center	default	Uptime	00:13
		Lease expiration enabled	true
		Renews threshold	8
		Renews (last min)	16
- DS Replicas:** A section with a search bar containing 'localhost'.
- Instances currently registered with Eureka:** A table listing registered services.

Application	AMIs	Availability Zones	Status
BILLING-SERVICE	n/a (1)	(1)	UP (1) - <a href="#">localhost:billing-service:8083</a>
CLIENT-SERVICE	n/a (1)	(1)	UP (1) - <a href="#">localhost:client-service:8081</a>
GATEWAY-SERVICE	n/a (1)	(1)	UP (1) - <a href="#">localhost:gateway-service:8888</a>
PRODUCT-SERVICE	n/a (1)	(1)	UP (1) - <a href="#">localhost:product-service:8082</a>
- General Info:** A section with a table for general information.

Name	Value
------	-------

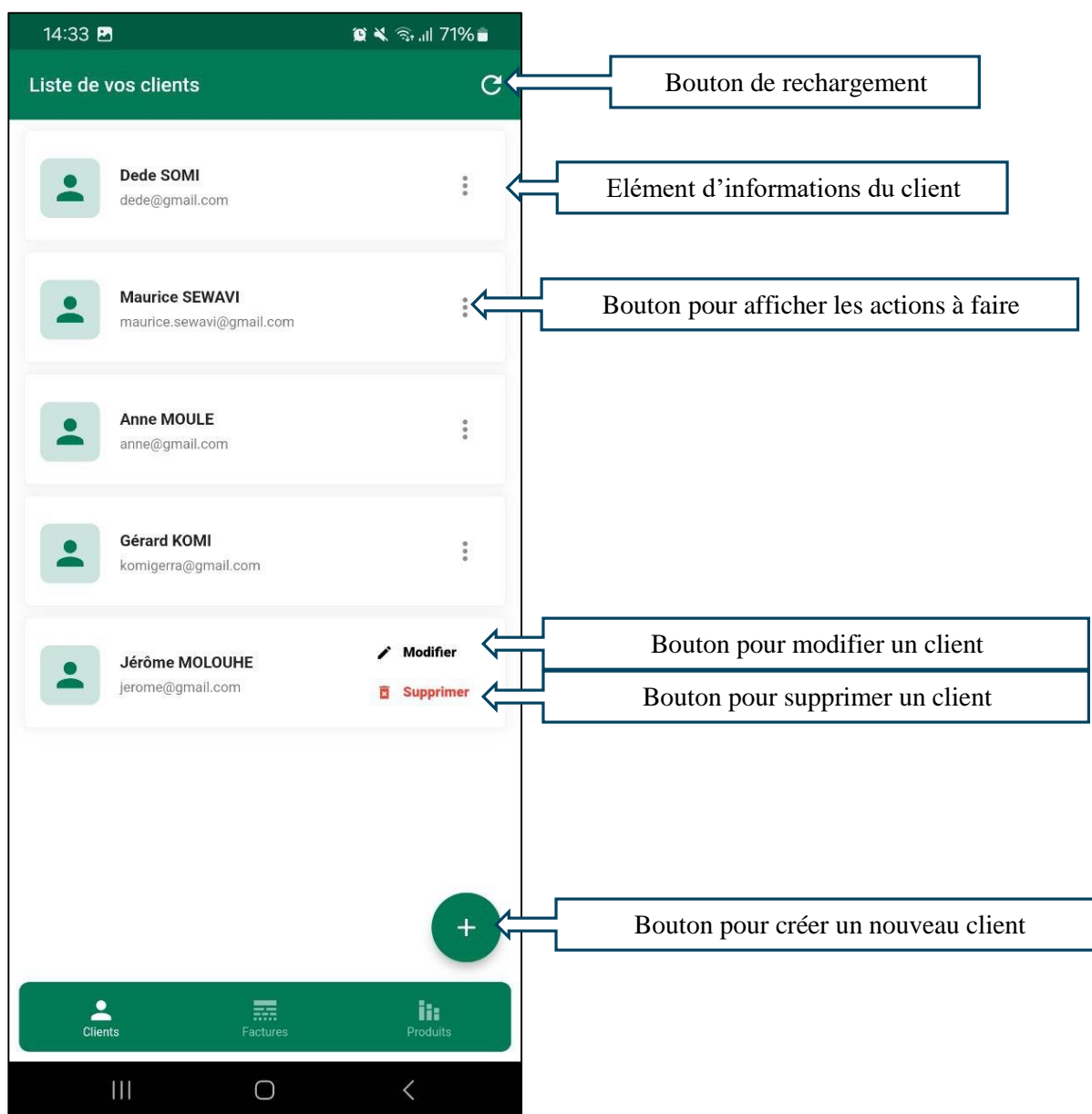
Nous pouvons désormais commencer les tests de chaque service en utilisant l'application mobile codée à cet effet.

## a) Gestion des clients

Pour gérer les clients, l'application offre quatre possibilités :

### - Lister les clients :

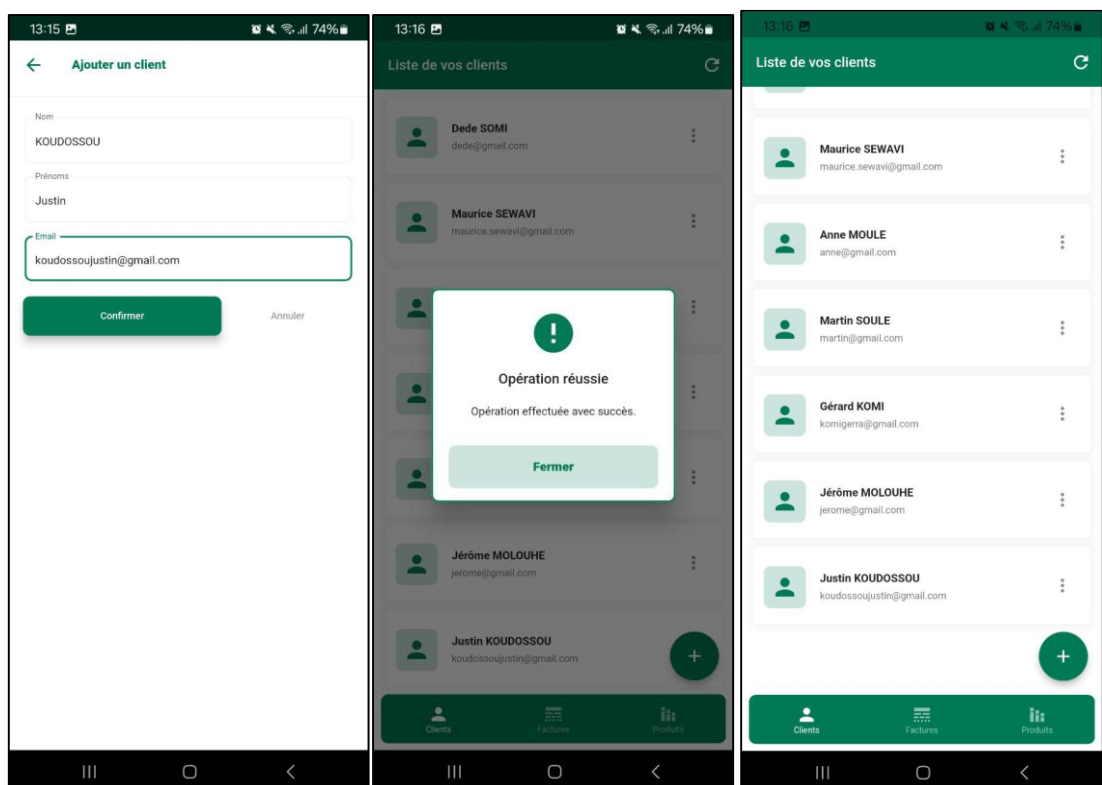
En récupérant les informations depuis <http://192.168.1.66:8888/CLIENT-SERVICE/clients>, l'application arrive à afficher les clients sous forme de liste comme ceci.



## - Créer un client :

Pour cela, après avoir appuyé sur le bouton de création d'un client, et en remplissant un formulaire avec nom, prénoms et adresse mail du client, l'application envoie les données via l'API au **Service Gateway** qui se charge de les renvoyer au **Service Client** pour création.

L'exemple suivant montre la création d'un nouveau client. Pour ce faire, après appui sur le bouton de création de client, l'on remplit le formulaire apparu avec les informations requises. Après cela, l'on peut voir le nouveau client apparaître en bas de la liste des clients. Le processus est représenté sur les trois images suivantes.

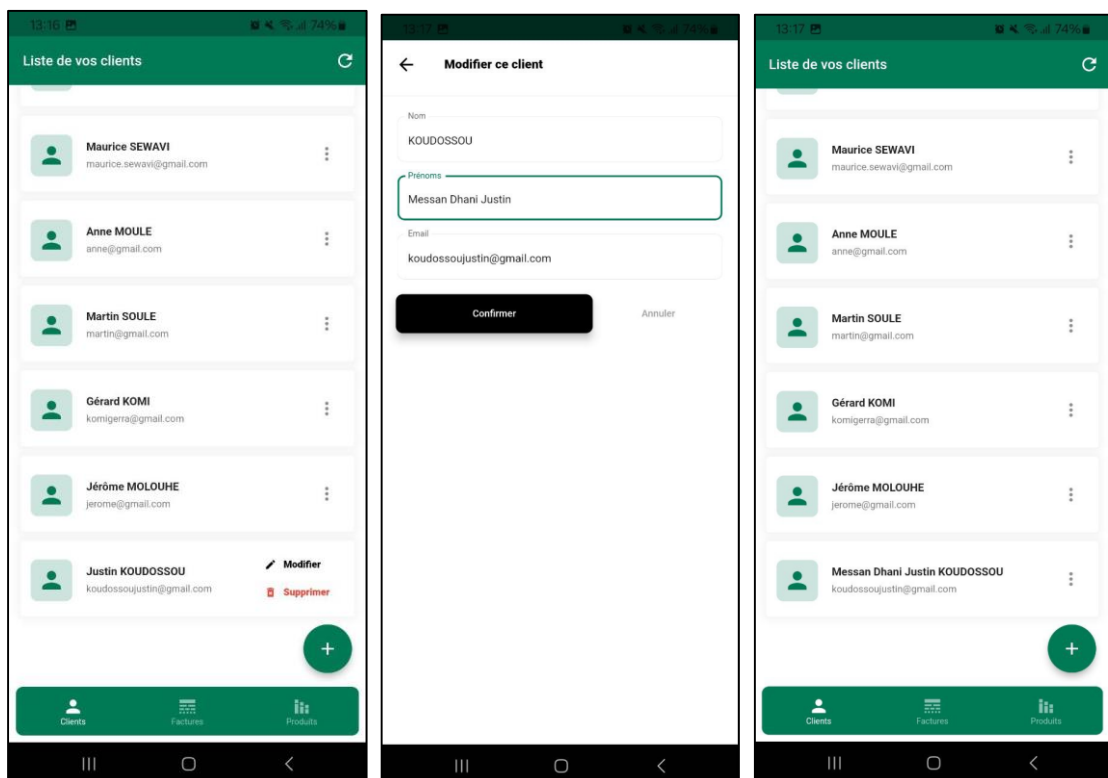




## - Modifier un client :

En mettant à jour les informations d'un client contenues dans un formulaire rempli avec son nom, ses prénoms et son adresse mail, l'on peut modifier les informations de ce client en utilisant l'application. Les nouvelles informations ainsi que l'identifiant du client à modifier sont transmis au **Service Gateway** qui à son tour les donne au **Service Client** pour modification.

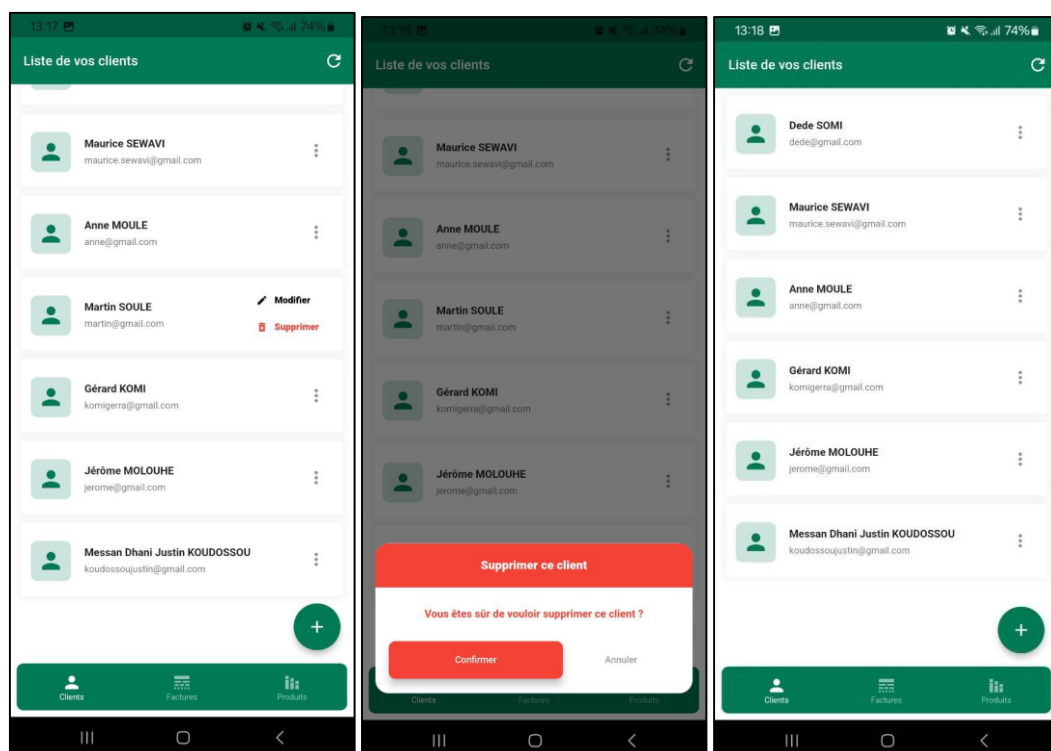
L'exemple suivant montre la modification du nouveau client créé précédemment, en cliquant sur le bouton des actions à faire sur ce client et en cliquant sur le bouton **Modifier**, puis en mettant à jour le formulaire apparu contenant ses informations. Après cela, l'on peut voir que le nouveau client apparaît avec ses informations bien modifiées. Le processus est représenté sur les trois images suivantes.



## - Supprimer un client :

Avec une confirmation à l'appui, l'on peut supprimer un client de la base de données des clients en utilisant l'API. L'application dans ce cas, transmet l'identifiant du client à supprimer à l'API via le **Service Gateway** qui se charge de l'envoyer au **Service Client** pour suppression.

L'exemple suivant montre la suppression d'un ancien client, en cliquant sur le bouton des actions à faire sur ce client puis en cliquant sur le bouton *Supprimer*. Il ne restera plus qu'à confirmer la suppression puis dans la liste, ce client aura disparu. Le processus est représenté sur les trois images suivantes.

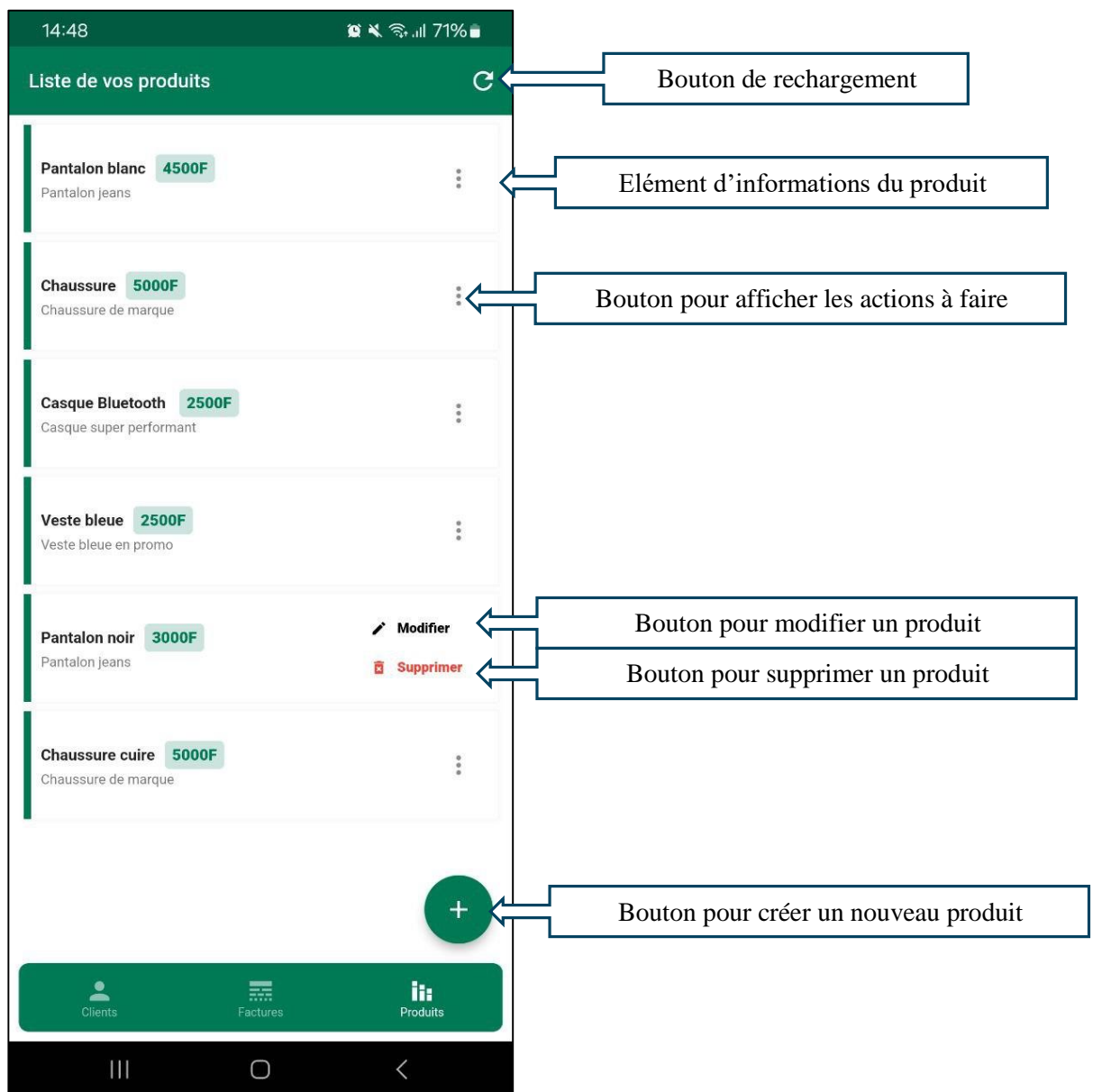


## b) Gestion des produits

Pour gérer les produits, l'application offre quatre possibilités :

### - Lister les produits créés :

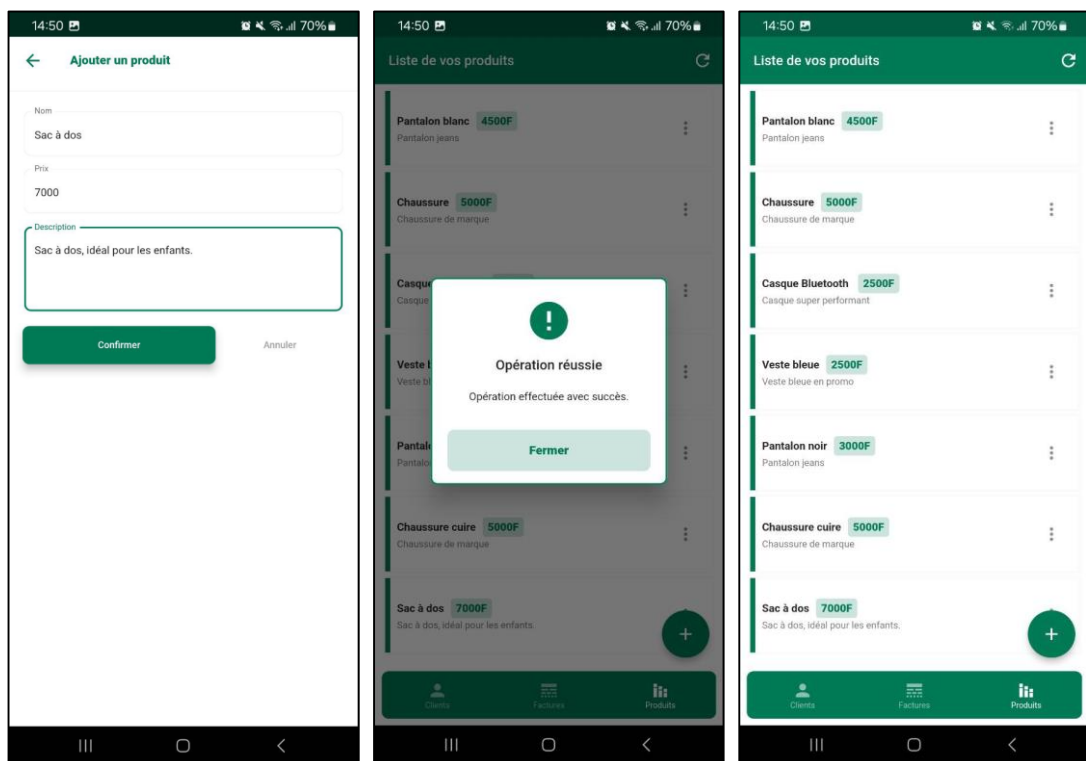
En récupérant les informations depuis <http://192.168.1.66:8888/PRODUCT-SERVICE/products>, l'application arrive à afficher les produits sous forme de liste comme ceci.



## - Créer un produit :

Pour cela, après avoir appuyé sur le bouton de création d'un produit, et en remplissant un formulaire avec nom, prix et description du produit, l'application envoie les données via l'API au **Service Gateway** qui se charge de les renvoyer au **Service Produit** pour création.

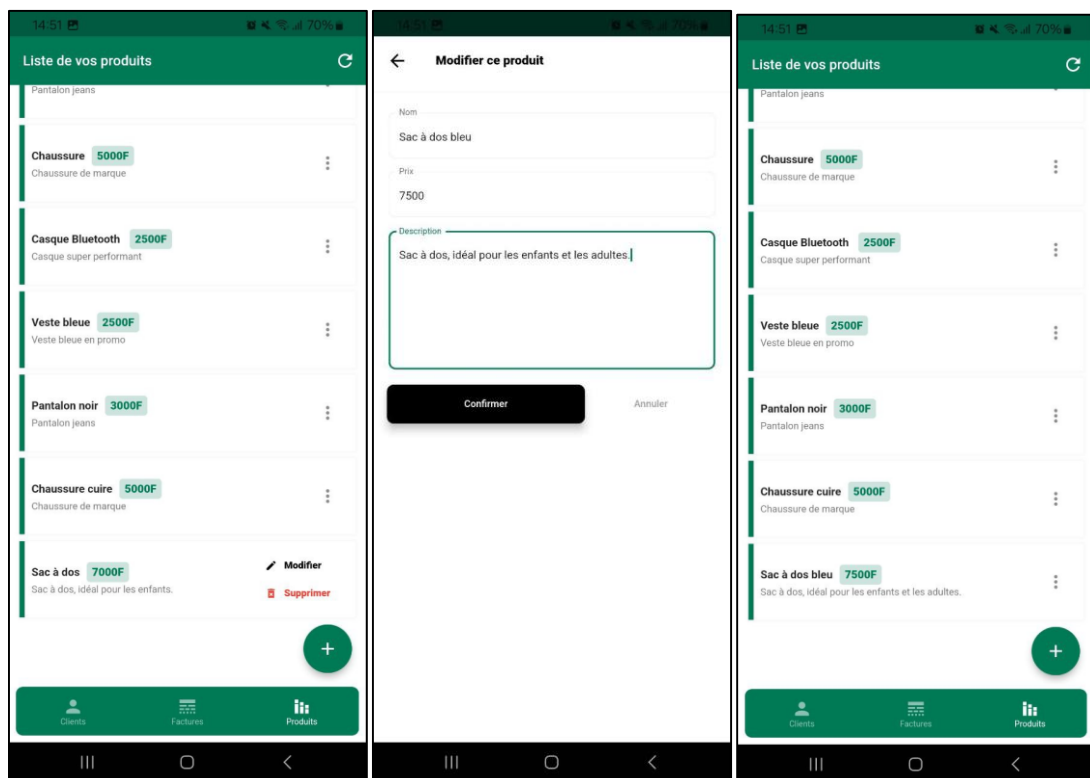
L'exemple suivant montre la création d'un nouveau produit. Pour ce faire, après appui sur le bouton de création de produit, l'on remplit le formulaire apparu avec les informations requises. Après cela, l'on peut voir le nouveau produit apparaître en bas de la liste des produits. Le processus est représenté sur les trois images suivantes.



## - Modifier un produit :

En mettant à jour les informations d'un produit contenues dans un formulaire rempli avec son nom, son prix et sa description, l'on peut modifier les informations de ce produit en utilisant l'application. Les nouvelles informations ainsi que l'identifiant du produit à modifier sont transmis au **Service Gateway** qui à son tour les donne au **Service Produit** pour modification.

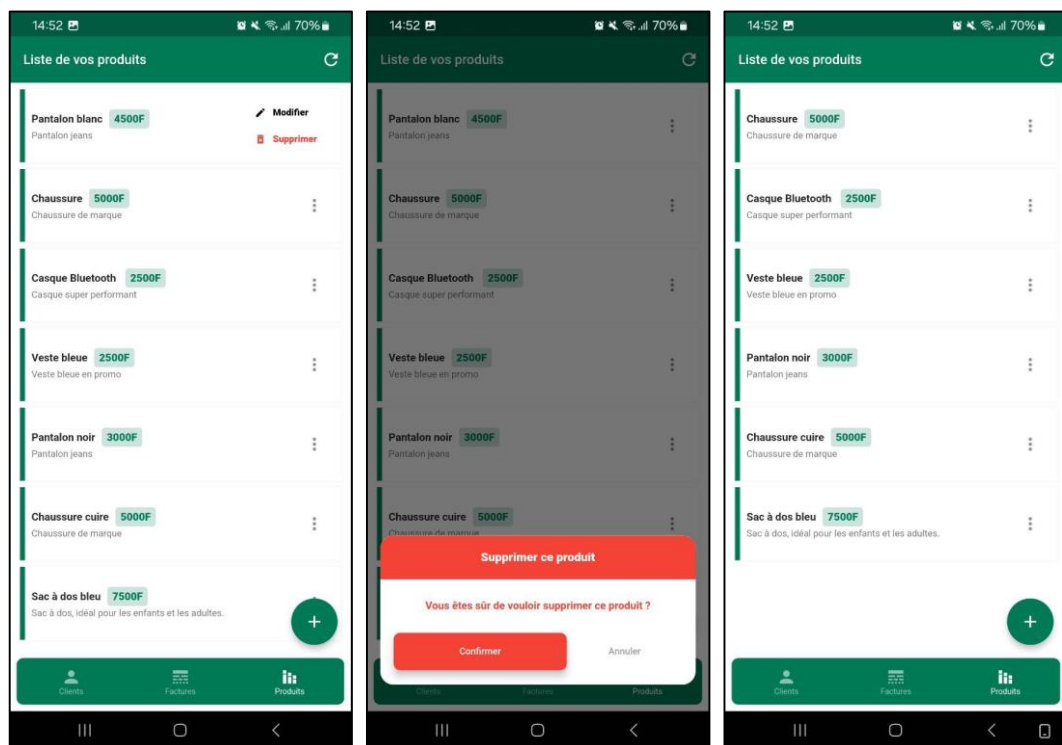
L'exemple suivant montre la modification du nouveau produit créé précédemment, en cliquant sur le bouton des actions à faire sur ce produit et en cliquant sur le bouton **Modifier**, puis en mettant à jour le formulaire apparu contenant ses informations. Après cela, l'on peut voir que le nouveau produit apparaît avec ses informations bien modifiées. Le processus est représenté sur les trois images suivantes.



## - Supprimer un produit :

Avec une confirmation à l'appui, l'on peut supprimer un produit de la base de données des produits en utilisant l'API. L'application dans ce cas, transmet l'identifiant du produit à supprimer à l'API via le **Service Gateway** qui se charge de l'envoyer au **Service Produit** pour suppression.

L'exemple suivant montre la suppression d'un ancien produit, en cliquant sur le bouton des actions à faire sur ce produit puis en cliquant sur le bouton *Supprimer*. Il ne restera plus qu'à confirmer la suppression puis dans la liste, ce produit aura disparu. Le processus est représenté sur les trois images suivantes.

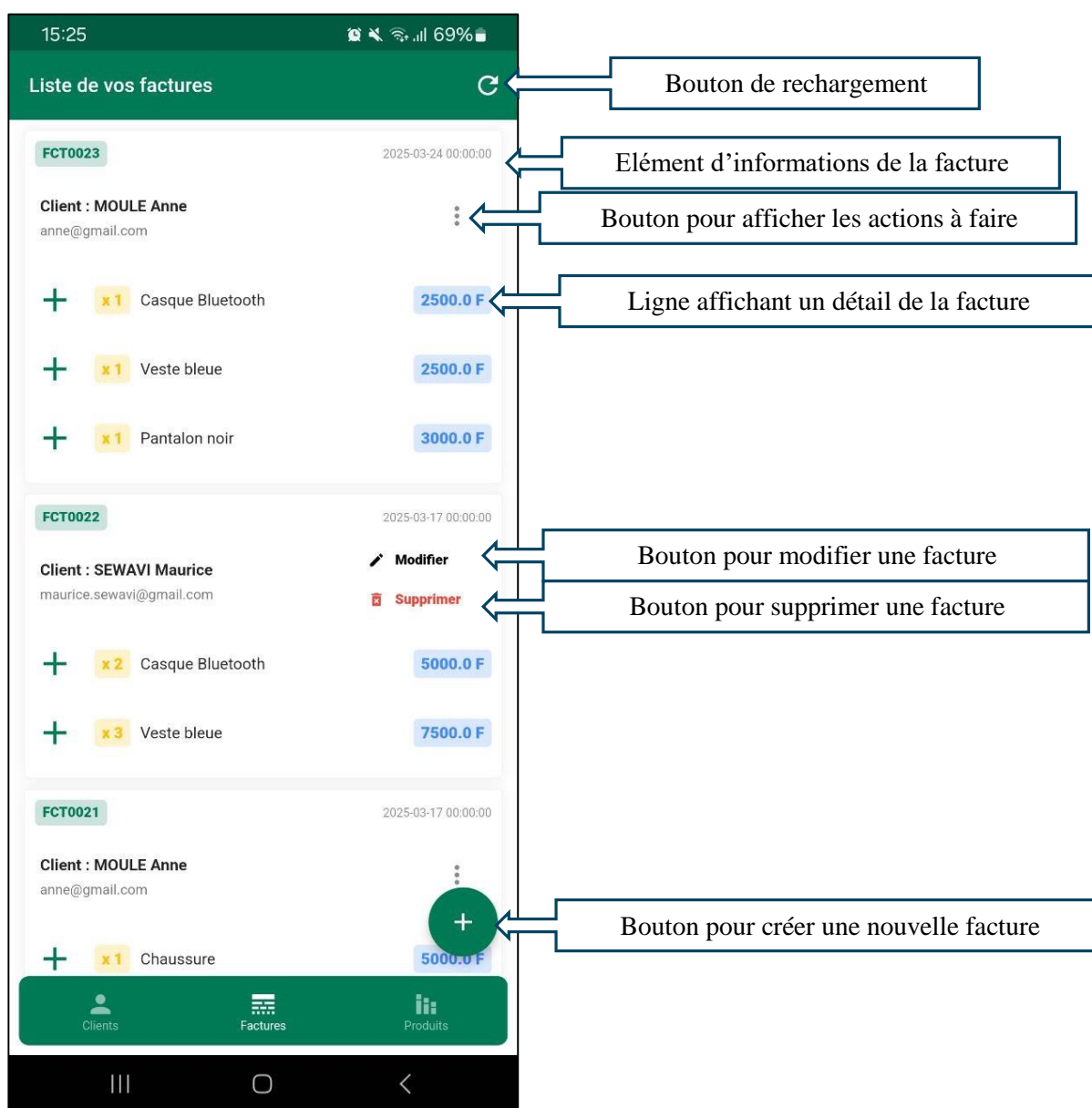


## c) Gestion des factures

Pour gérer les factures, l'application offre huit possibilités :

### - Lister les factures créées :

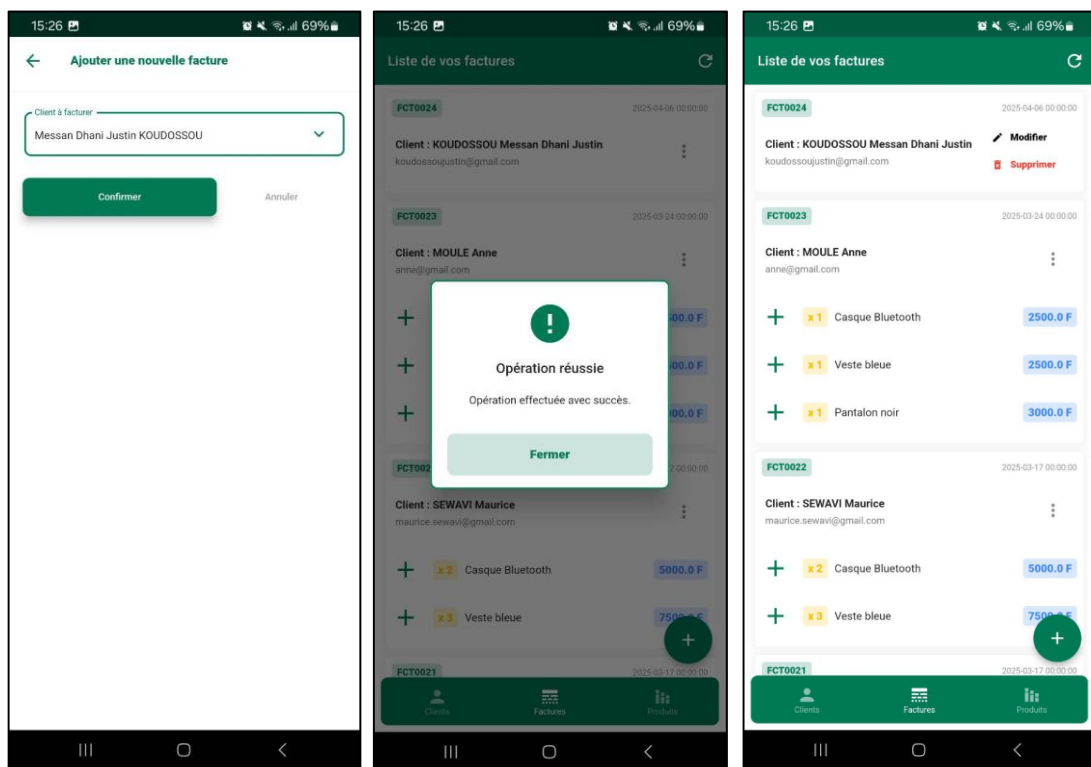
En récupérant les informations depuis <http://192.168.1.66:8888/BILLING-SERVICE/bills-rest>, l'application arrive à afficher les factures sous forme de liste comme ceci.



## - Créer une facture :

Pour cela, après avoir appuyé sur le bouton de création d'une facture, et en remplissant un formulaire en choisissant le client à facturer, l'application envoie les données via l'API au **Service Gateway** qui se charge de les renvoyer au **Service Facturation** pour création d'une facture.

L'exemple suivant montre la création d'une nouvelle facture. Pour ce faire, après appui sur le bouton de création de facture, l'on remplit le formulaire apparu en choisissant le client à facturer. Après cela, l'on peut voir la nouvelle facture apparaître en haut de la liste des factures. Le processus est représenté sur les trois images suivantes.

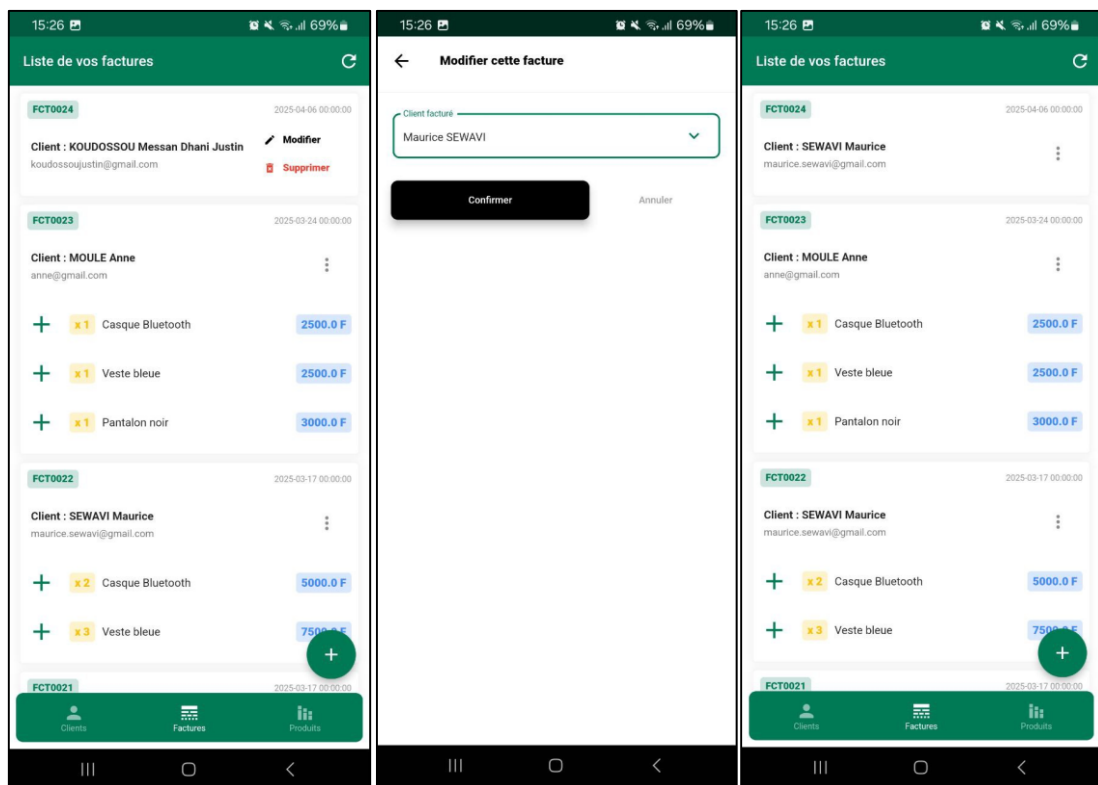




## - Modifier une facture :

En mettant à jour les informations d'une facture contenues dans un formulaire rempli avec le client facturé, l'on peut modifier les informations de cette facture en utilisant l'application. Les nouvelles informations ainsi que l'identifiant de la facture à modifier sont transmis au **Service Gateway** qui à son tour les donne au **Service Facturation** pour modification de la facture.

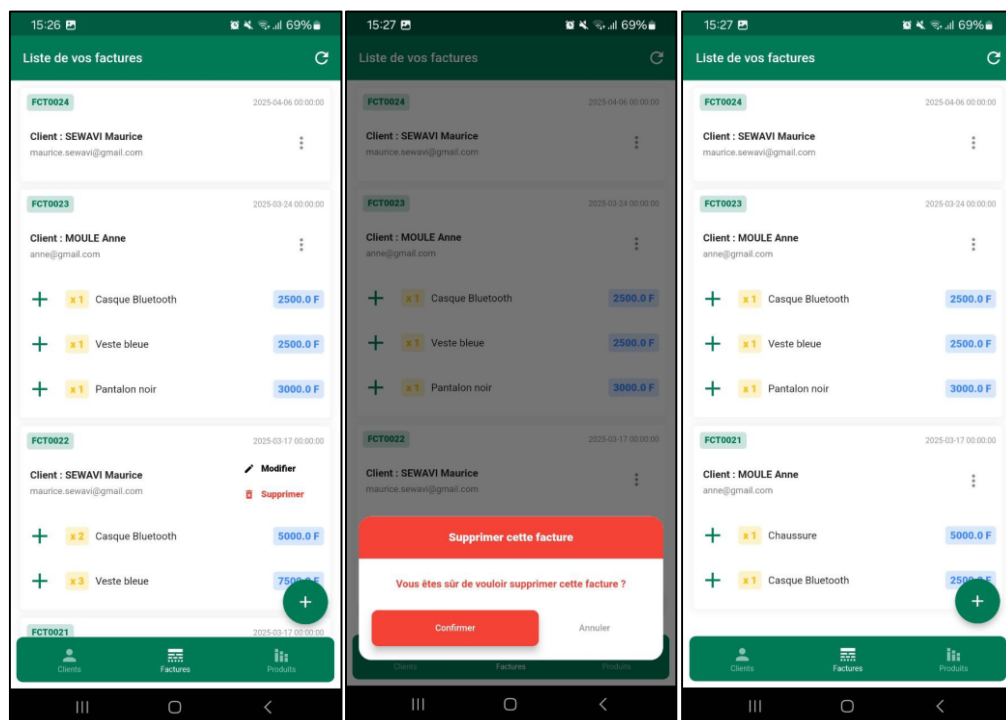
L'exemple suivant montre la modification de la facture créée précédemment, en cliquant sur le bouton des actions à faire sur cette facture et en cliquant sur le bouton *Modifier*, puis en mettant à jour le formulaire apparu contenant ses informations. Après cela, l'on peut voir que la nouvelle facture apparaît avec ses informations bien modifiées. Le processus est représenté sur les trois images suivantes.



## - Supprimer une facture :

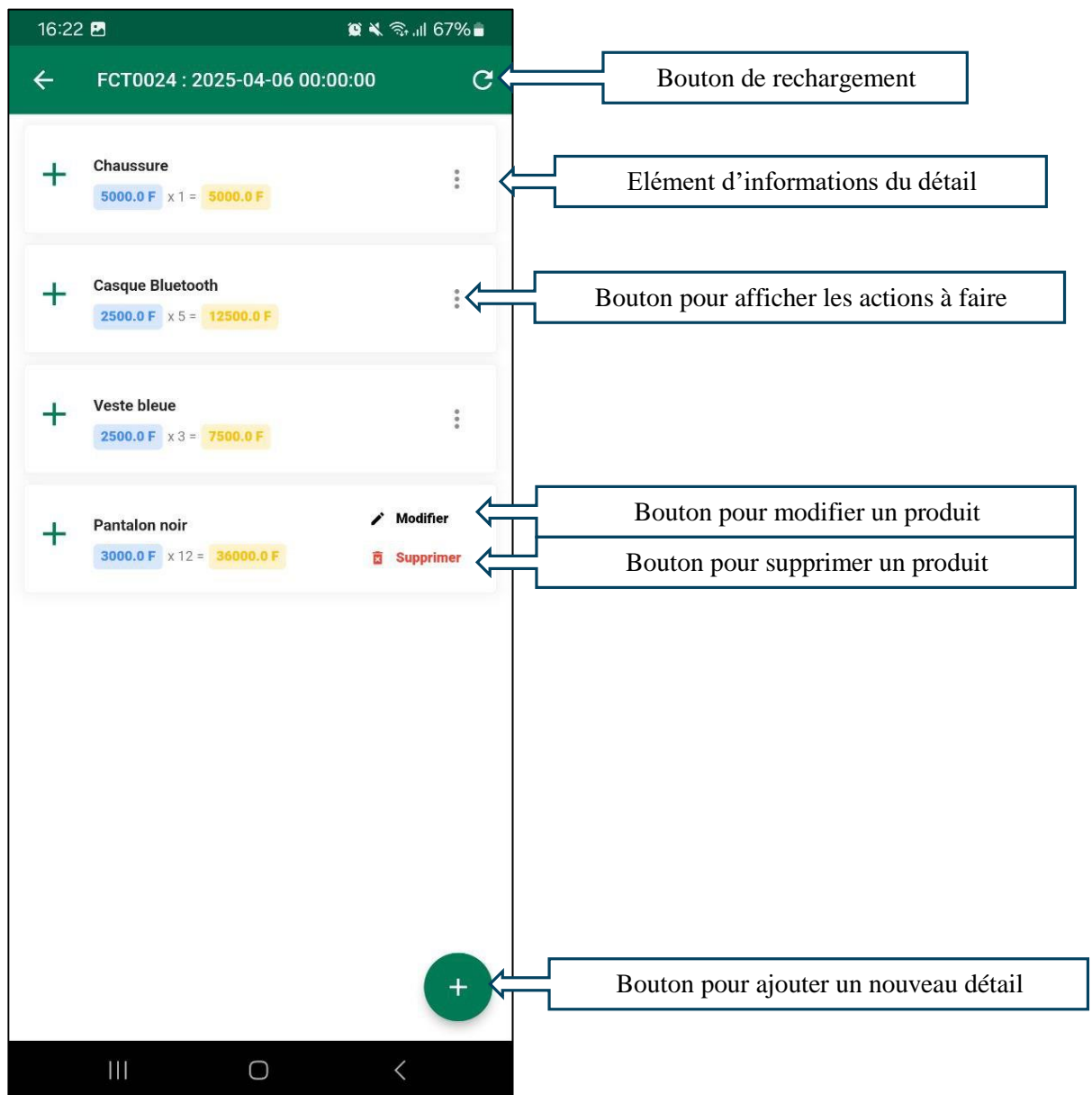
Avec une confirmation à l'appui, l'on peut supprimer une facture de la base de données de facturation en utilisant l'API. L'application dans ce cas, transmet l'identifiant de la facture à supprimer à l'API via le **Service Gateway** qui se charge de l'envoyer au **Service Facturation** pour suppression de facture.

L'exemple suivant montre la suppression d'une ancienne facture, en cliquant sur le bouton des actions à faire sur cette facture puis en cliquant sur le bouton **Supprimer**. Il ne restera plus qu'à confirmer la suppression puis dans la liste, cette facture aura disparu. Le processus est représenté sur les trois images suivantes.



- **Lister les détails d'une facture créés :**

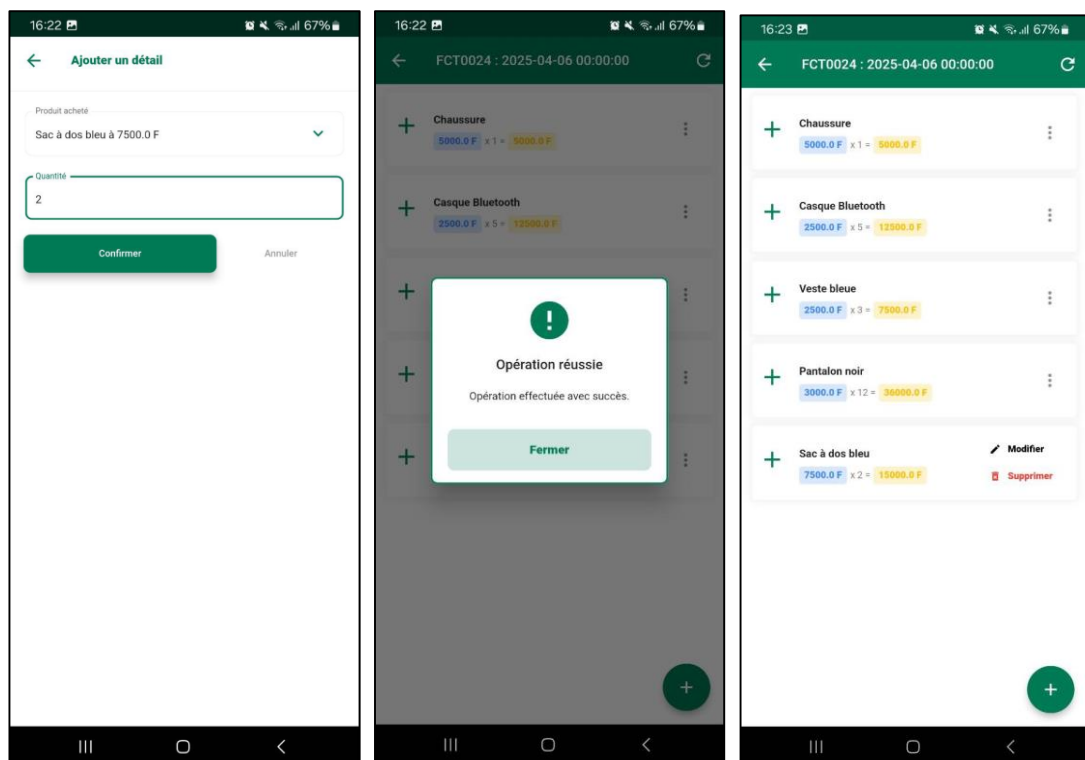
En cliquant sur une facture, l'application arrive à afficher les détails de cette facture préalablement chargée, sous forme de liste comme ceci.



- **Ajouter un détail à la facture :**

Pour cela, après avoir appuyé sur le bouton d'ajout d'un détail, et en remplissant un formulaire avec le produit et la quantité achetée, l'application envoie les données via l'API au **Service Gateway** avec l'identifiant de la facture, qui se charge de les renvoyer au **Service Facturation** pour création d'un nouveau détail pour la facture sélectionnée.

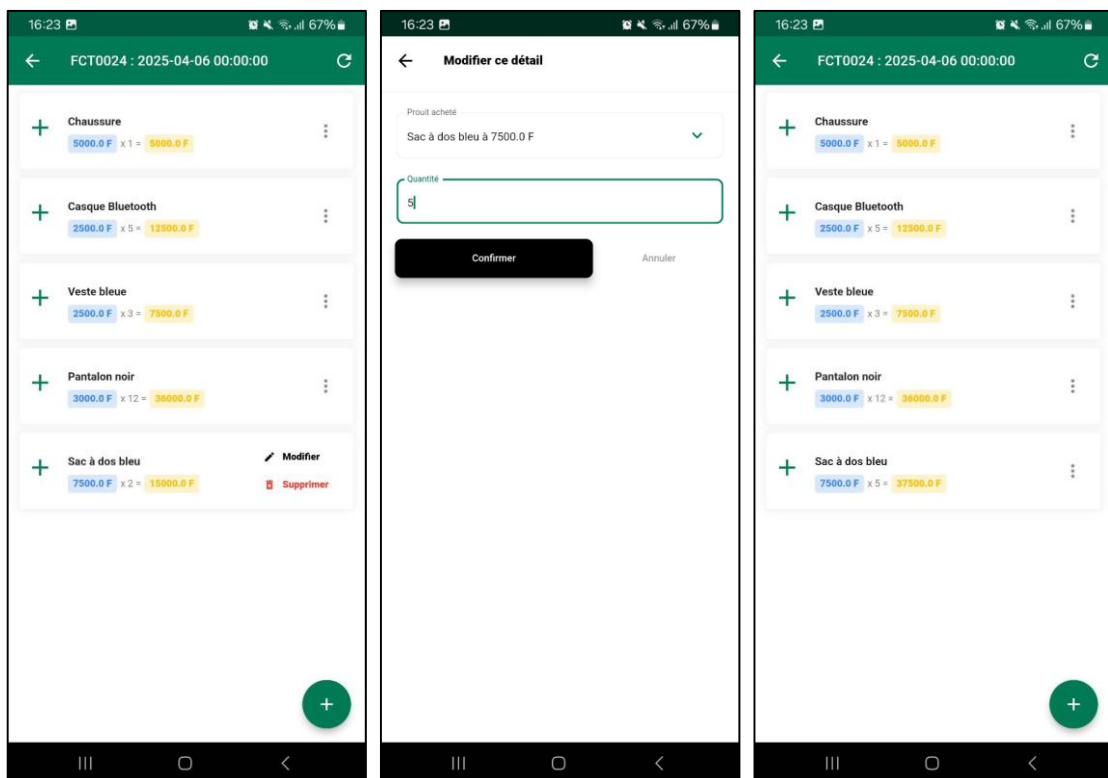
L'exemple suivant montre l'ajout d'un nouveau détail à la facture créée dans l'exemple de création d'une facture. Pour ce faire, après appui sur le bouton d'ajout de détail, l'on remplit le formulaire apparu avec les informations requises. Après cela, l'on peut voir le nouveau détail apparaître en bas de la liste des détails de la facture. Le processus est représenté sur les trois images suivantes.



## - Modifier un détail de la facture :

En mettant à jour les informations d'un détail contenues dans un formulaire rempli avec le produit et la quantité achetée, l'on peut modifier les informations de ce détail en utilisant l'application. Les nouvelles informations ainsi que l'identifiant du détail à modifier sont transmis au **Service Gateway** qui à son tour les donne au **Service Facturation** pour modification d'un détail.

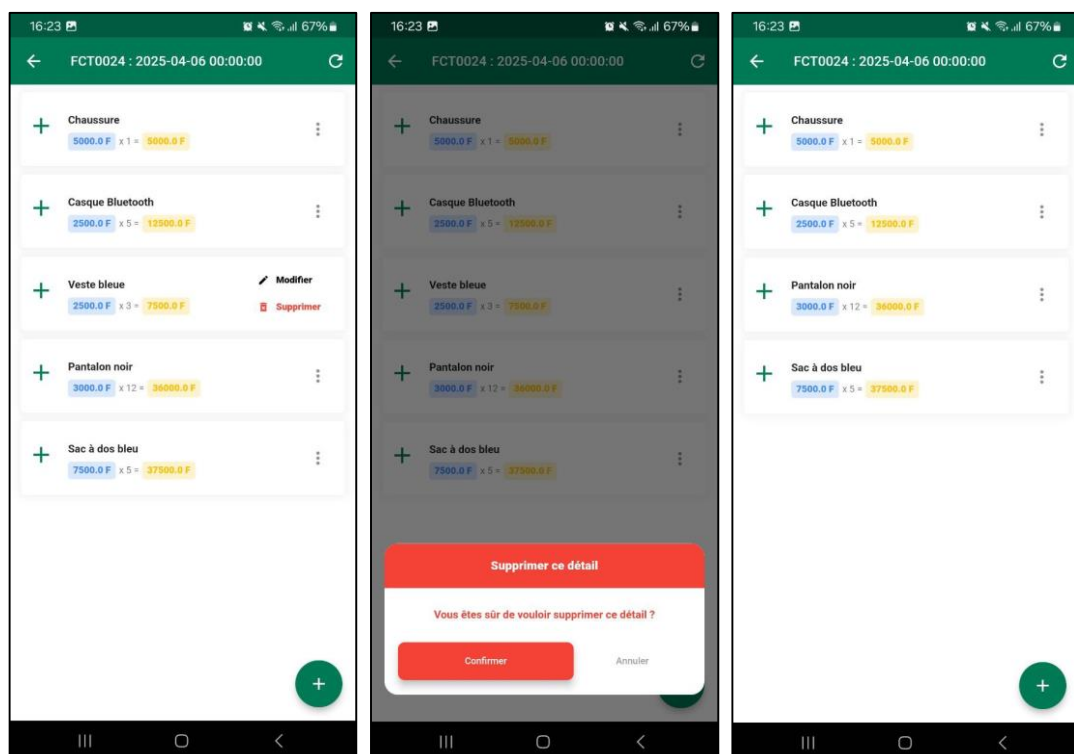
L'exemple suivant montre la modification du nouveau détail créé précédemment, en cliquant sur le bouton des actions à faire sur ce détail et en cliquant sur le bouton **Modifier**, puis en mettant à jour le formulaire apparu contenant ses informations. Après cela, l'on peut voir que le nouveau détail apparaît avec ses informations bien modifiées. Le processus est représenté sur les trois images suivantes.



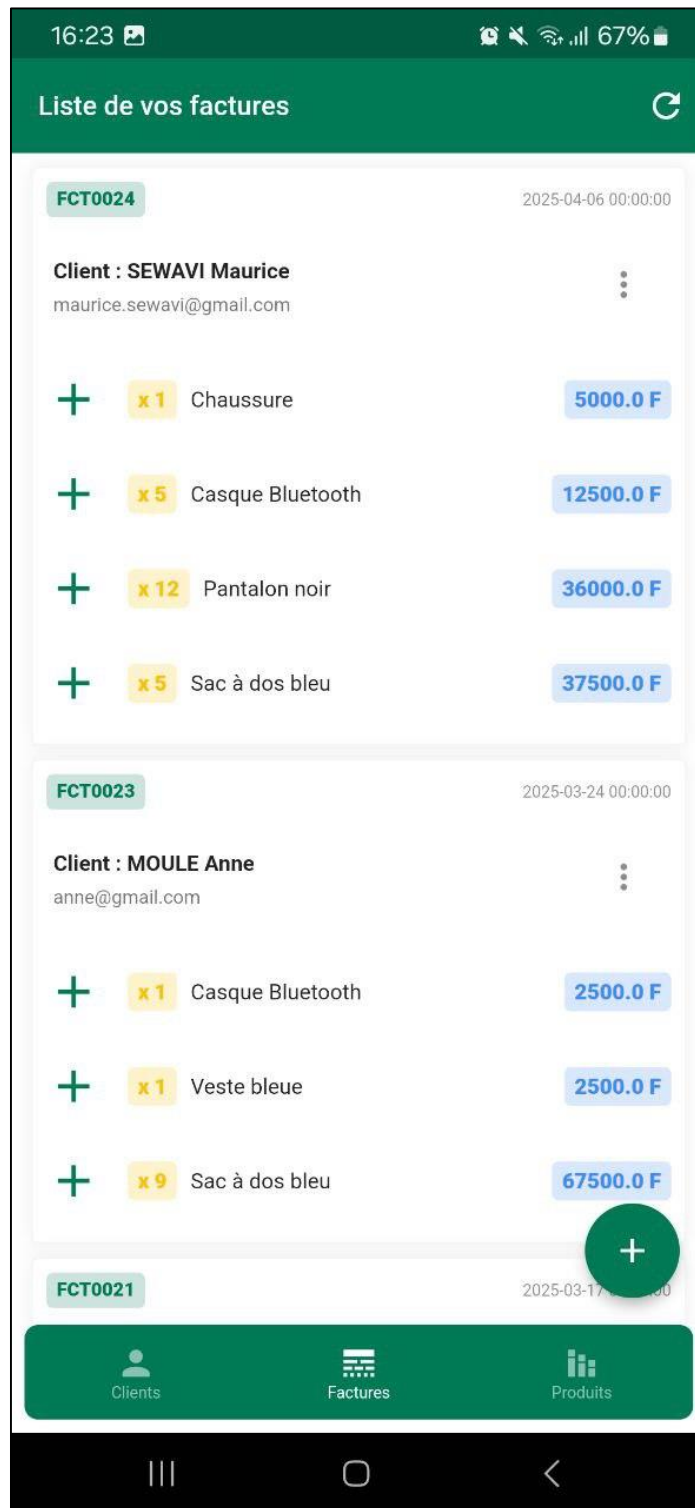
- **Supprimer un détail parmi ceux d'une facture :**

Avec une confirmation à l'appui, l'on peut supprimer un détail d'une facture de la base de données de facturation en utilisant l'API. L'application dans ce cas, transmet l'identifiant du détail à supprimer à l'API via le **Service Gateway** qui se charge de l'envoyer au **Service Facturation** pour suppression d'un détail.

L'exemple suivant montre la suppression d'un ancien détail, en cliquant sur le bouton des actions à faire sur ce détail puis en cliquant sur le bouton *Supprimer*. Il ne restera plus qu'à confirmer la suppression puis dans la liste des détails de la facture, ce détail aura disparu. Le processus est représenté sur les trois images suivantes.



Après, en retournant sur la page d’affichage des factures, l’on constate effectivement que les actions effectuées ont réussi.



### III. Conclusion

En conclusion, ce rapport a présenté une application backend développée en Java avec Spring Boot, exploitant une architecture de microservices efficace pour offrir une API fonctionnelle. Grâce aux cinq services principaux — **Gateway**, **Eureka**, **Client**, **Produit** et **Facturation** — l'application démontre une gestion fluide et intégrée des données, permettant aux utilisateurs de gérer facilement les clients, les produits et les factures via une application cliente en Flutter.

Les interactions entre l'application cliente et l'API, facilitées par le **Service Gateway**, illustrent les avantages d'une telle architecture, notamment en termes de scalabilité, de flexibilité et de maintenance. En adoptant cette approche, les équipes de développement peuvent créer des applications modernes et réactives, adaptées aux besoins changeants du marché. Ce projet sert ainsi de référence pour ceux qui souhaitent explorer ou mettre en œuvre des solutions similaires basées sur des microservices, soulignant l'importance d'une architecture bien conçue pour le succès des applications contemporaines.