Private Higher School Of Technologies And Engineering

Internship Report

Recommendation System Smart List

Hadhemi Wesleti

Supervised by
Mohamed Taher Ben Bahri

Academic year 2019-2020

Acknowledgments

Before presenting my work, I would like to express my deep appreciation to the people who have helped me, from near or far, to provide their support. May they find here collectively and individually the expression of all my gratitude.

I would like to thank especially and to express my gratitude to Mr Mohamed Taher Ben Bahri for the enriching and interesting experience he has given us during the internship period for me and my colleagues, for the time being. he was good enough to devote himself to the supervision and follow-up of this work; the advice he gave us after his meticulous readings and the meetings that punctuated the various stages of writing this report.

The discussions we held made it possible to guide this work in a safe and relevant way. We thank him very much for his efforts, his availability and especially his advice, which have greatly contributed to enhancing the value of this work.

Also, I express my perfect thanks to my colleagues who supported me and gave me a perfect working atmosphere during the internship period.

Contents

0.1 Introduction

The biggest challenge for e-commerce business is ensuring a superior customer service to shopper. Helping them find what they looking for and guiding and guiding their shopping experience is what make the process challengeable.
The e-commerce stores have chatbots that are supposed to replace the helping salespeople in shops. However, although online salespeople can handle some issues, they are unable to deliver as much value as the person in an actual store.
Product recommendation engines were created to overcome the issue of virtual communication in e-commerce stores with the help of personalization. One key procedure to acheive mass customization in e-commerce is the use of recommender systems.
Recommender system is a system that is able of predicting the future preference of a set of items for a user, and recommend the top items.
Recommender systems have gained the most attention in modern society because people have too much options to use from due to the prevalence of internet.

The main motivation of this project comes from a survey conducted by the host company Mind Engineering to try respond efficiently to customers expectations to provide a smart recommendation system. This survey reveals that 71% of respondents prefer to provide them with a tool that maintain their shopping list among the online shops. A large majority of them (81%) say that there is a huge difference between their expectation and real products and that they are not satisfied with their experience of shopping online.
According to respondents, this is due to the lack of products variety, features and specifications as well as the lack of comparison and research of products and prices. Also, the results show that 76% of the respondents are willing to purchase at a shop where they are getting maximum help through a recommendation system that suggests better prices and best deals. The overall aim of this project is to develop a system that makes it possible to manage shopping lists by promoting the social aspect and decision support. In order to fulfill this mission, the work is divided into two main parts per team : "Front-end" (part1) and "Back-end" (part2), we have chosen to work with the scrum as an agile method. The present work therefore outlines the maintaining of the back-end part based on microservices architecture. These microservices will be accessed through APIs *. Thus both end user-side server-side can communicate with each other via the APIs.
The manuscript is structured as follows:

- First, this chapter one will introduce the topic of this internship as well as the background information obtained prior to the beginning, and will define the requirements to be completed for this project.

- Then, chapter two will state the conceptual design and the project approach.

- And, we conclude by chapter three which will give a specific description of the microservices setting, testing and implementation.

CHAPTER 1

Requirements Analysis

1.1 General presentation

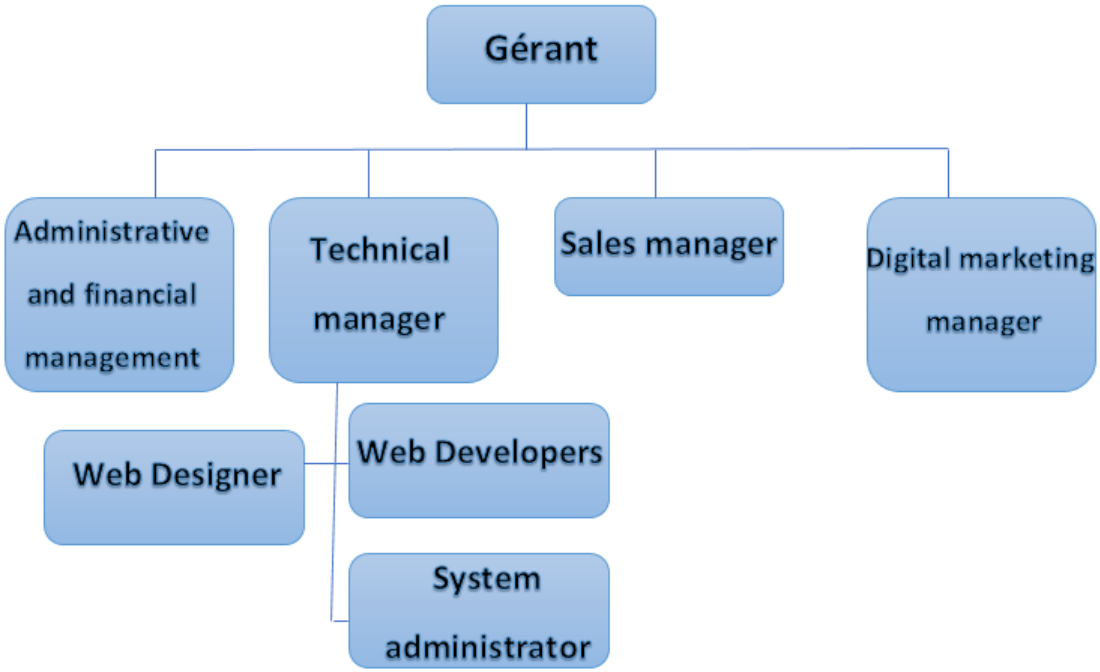
In this first chapter, we will first examine the general framework of our project by presenting the host organization. Once the problem is explained, we will study the current existing, then reveal the background of the project and determine the requirement identification.

1.1.1 Presentation of the host organisation

Mind Engineering was created 05/01/2016 in the form of an LLC. It is a web agency specialized in the design of innovative software solutions. Mind Engineering offers innovative solutions with new communications technologies.

1.1.2 Sector activities

1.1.2.1 Organigram



1.1.2.2 Technical Service

Aim for the production of tools or new ways of doing things through expressing, scientific knowledge and research data. They consist of three teams: web designer, system administrator and web developers.

-Web Developers: Design, define and carry out the design and development of new innovative products allowing the company to continuously adapt its products or processes and create new ones to meet the needs of the market.

-Web designer: Is a web artist his role is designing a web interface, a graphic identity, the organization of pages and the other navigation on the site.
Mind Engineering offers a wide range of web design: blogs, forums, online shops, information portals, online newspapers and multimedia galleries.

- System administrator: He is responsible for the availability of information, his role is not limited to the resolution of problems, but it must offer solutions in line with the news and needs of his client.

1.1.2.3 Administrative and Financial Service:

- The financial service: Ensures the preparation of the budget and its execution, consists in controlling the profitability of the company, with the management of the study and research agreements as well as the management of the public markets. It also has a consulting function on all financial data.

- Commercial service: Prospecting is his first job and involves bringing new customers to the company. Also allows him to transmit information about the product or service to the targeted customers, intervene before and after the sale by making himself available to the customer and listening to his remarks. In this

way, it is important to take into account the specificities of the target market and the market positioning of the company in each market.

1.2 Project background

In this section, we will give a brief introduction of the fundamentals of recommendation engines. Then we will provide some definitions and details about recommendation systems in the field of e-commerce.

1.2.1 Basics of recommendation systems

This part aims at defining recommendation systems and presenting its main functionalities. Afterwards, we explain the different representations of data within these systems as well as the techniques that have the potential to significantly augment current recommendation systems.

1.2.1.1 Definition

An information filtering technology, commonly used on e-commerce Web sites that uses a collaborative filtering to present information on items and products that are likely to be of interest to the reader. In presenting the recommendations, the recommender system will use details of the registered user's profile and opinions and habits of their whole community of users and compare the information to reference characteristics to present the recommendations. An example of a recommender system is WhatShouldIReadNext.com, a site where users can enter a title of a recent book they have read and enjoyed to see recommended books that they are likely to also enjoy.

1.2.1.2 Fonctionnalities of recommendation system

A recommendation system has different functionnalities that depend on the type of field of application. These are as follows:

-Prediction: The prediction feature is generally applied in the collaborative recommendation approach. At this level,the system is based on predict the ranking of a non-user reviewed item.

-Recommendation: This feature displays a list of items stored according to users needs and preferences.

-Constraint-based recommendation: In this type of recommendation, a user is required to define a set of constraints on items to recommend.

1.2.1.3 Data representation for a recommendation system

A recommendation system is an information processing system. To provide suggestions, this system uses three different types of data. Next, we present these three types of representations.

-Data representation for items: Depends on the recommendation technique.Indeed, in the case of the collaborative recommendation technique, an item is represented by an UID that allows it to be uniquely identified.

-Data representation for users: In recommendation systems, a user is represented by an UID that enables him/her to be identified. It can also be characterized by a set of demographic data. In this case, he/she is expected to complete a form by providing age, gender, geographical location (country,city).

-Recommending items based on user needs and preferences: This is done either explicitly by asking users to provide their preferences, for example by filling in a form, or implicitly by analyzing the past history of their interaction with the system. The purpose of the analysis is to deduct the evaluations given by users for each item they have viewed.

1.2.1.4 Recommendation techniques

Recommendation systems are mainly based on recommendation techniques. These have been classified into three categories:

-Content-based systems: Consider the properties of the items to be recommended.

-Collaborative filtering approach: Uses customer detailed, ratings, and reviews aggregated from all the customers to build recommendations.

Hybrid recommendation: Includes implement both filtering separately and combine the results, incorporating characteristics of content-based filtering.

1.2.2 E-commerce recommendation systems

Recommendation systems have gained importance in recent years, and appear very promising for e-commerce, especially in connection with Big Data. By working with large amounts of data and using sophisticated algorithms, you can increase the conversions of online stores with modern recommendation systems. You are no doubt familiar with the classic 'customers who bought this item also bought...' or 'the following products may interest you...' notifications in online shopping. These tips, corresponding to the individual preferences of the user, result from considerable computing power and complicated algorithms. In e-commerce, recommendation systems are already incorporated, but other sectors also benefit from continuously improving algorithms and technologies. The importance of Big Data is revealed in developing a more in-depth and insightful understanding of the company business which can let to a better productivity, a strengthened competitive position and innovation.
E-commerce applications and websites use different techniques to provide users with better experience.
An effective recommendation system is a core capability all e-commerce applications require. It is based upon powerful computer algorithms and robust mathematical models that quickly provide products that have a high probability of purchase, saving your time and money.

The outcome will be the suggested approach for implementing Smart List microservices.

1.3 Requirements identification

This section provides detailed information regarding the functional and non-functional requirements.

1.3.1 Actors

As a pre-requisite to identifying requirements, the first task is to define the entities that will interact with the back-end microservices:
-User: Who will simulate the actual front-end actor to test the proper running of microservices

1.3.2 Functional requirements

The purpose of this project is to build a system that makes it possible to manage shopping lists by promoting the social aspect and decision support. This will require fulfilling the following requirements per service:
-User Management Service: Allows users to manage their own group or join other groups.

-Product Management Service: Cover the operations that can be performed on each product(search,add,update,d

-Point Of Sale Management service: Cover the operations that can be performed on each point of sale(search,add,update,delete) based on mainly on google maps and places informations.

-List Management Service: This module is used to define the operations that will be applied to the shopping lists including shopping list creation, sharing lists and adding items to existing lists.

-Shopping Service: This microservice is considered as the most heavy microservice since, it allows the customer to choose the course mode, scan the desired products, change the price of a specific product, all of intelligent and algorithm is here. This module is devoted to the decision support part for optimally handling on-demand purchasing products to better guide customers in making decisions based on dynamic, real-time data.

-Notifications Service: This service keeps users informed of revelant events. The system can notify users by email, SMS text message, or push notification.

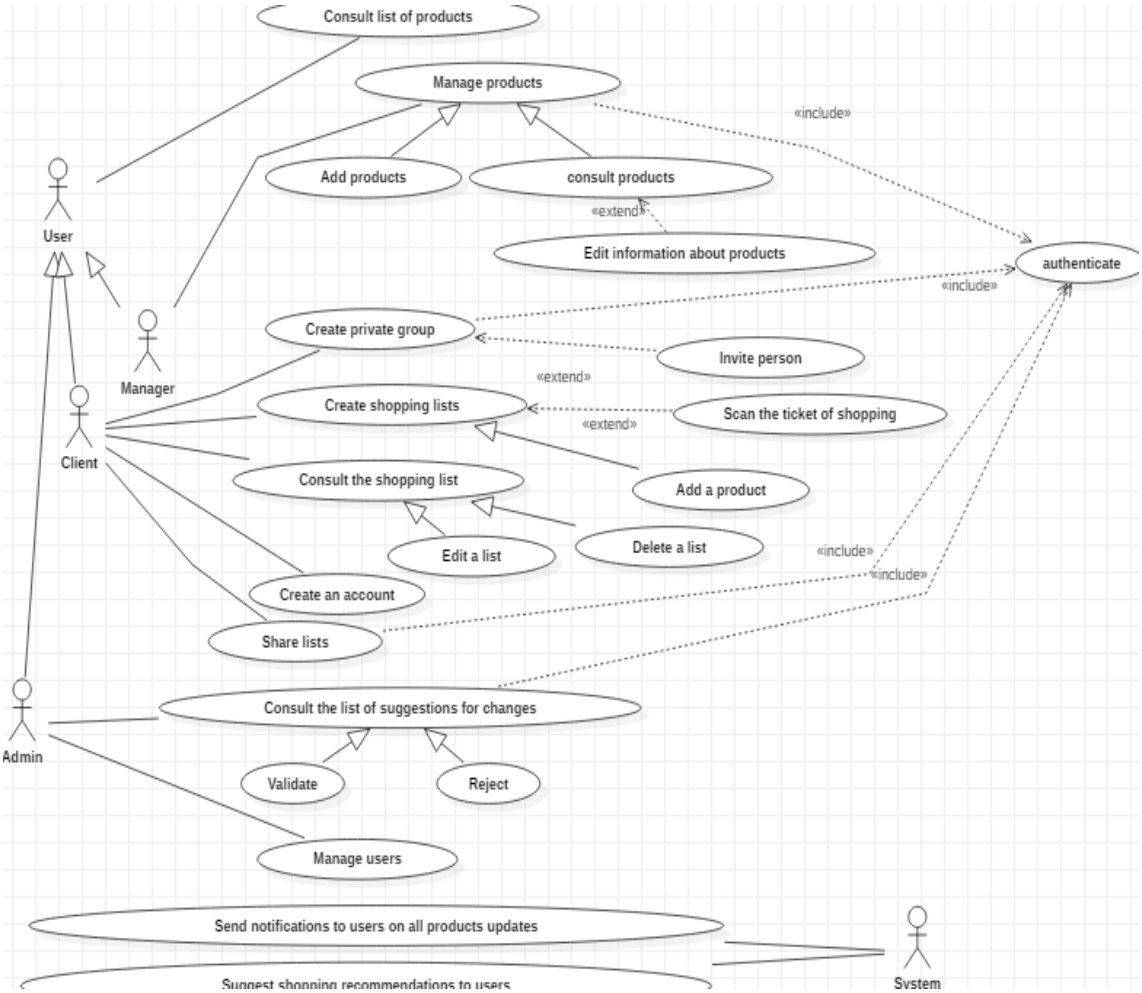
1.3.2.1 Use case diagram

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

-Scenarios in which your system or application interacts with people, organizations, or external systems

-Goals

-The scope of your system



dutilisation.png

Figure 1.1 – General use Case of Smart List

1.3.2.2 User Management Service

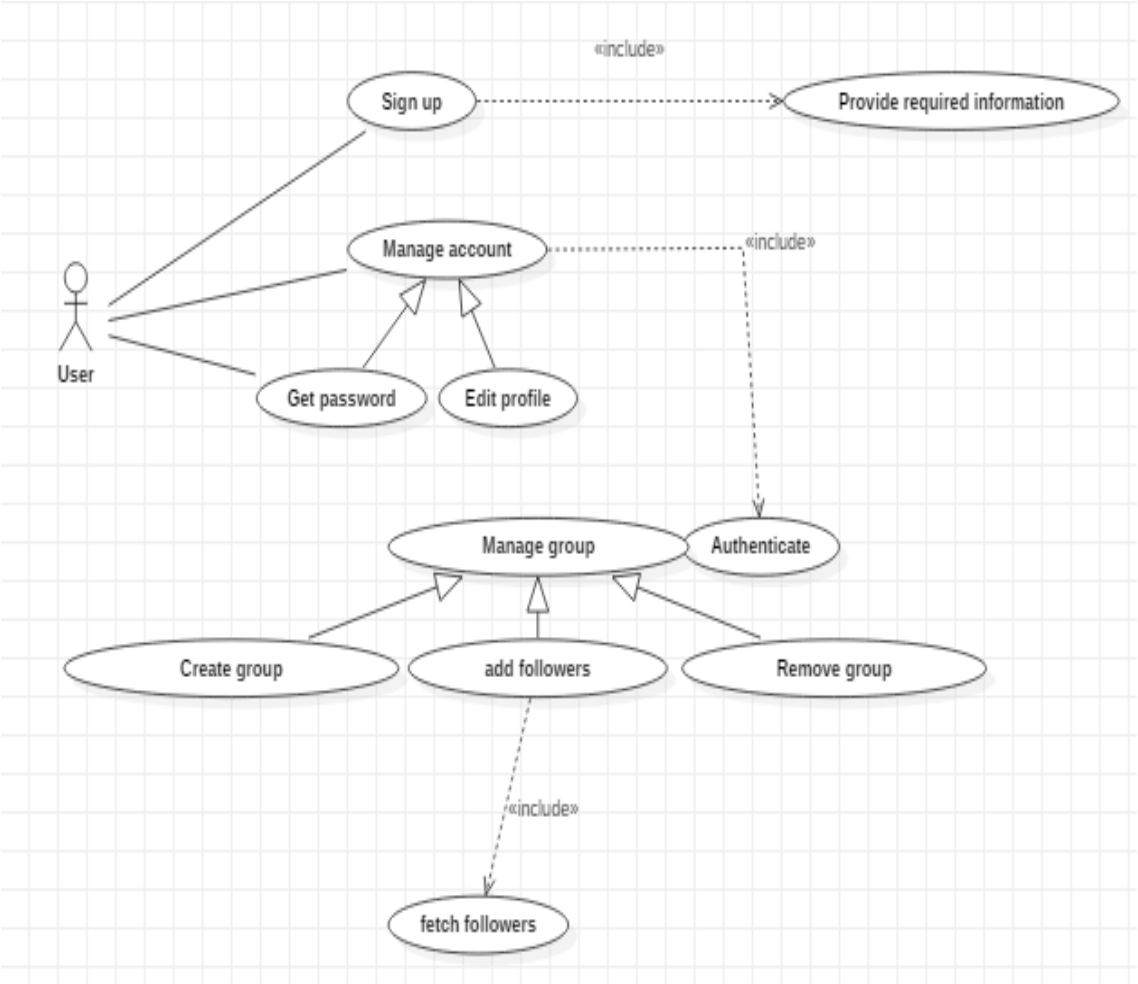


Figure 1.2 – Use case diagram for User Management Service

1.3.2.3 Product Management Service

The use case diagram displayed in the figure below outlines the different operations that can be performed through this microservice.

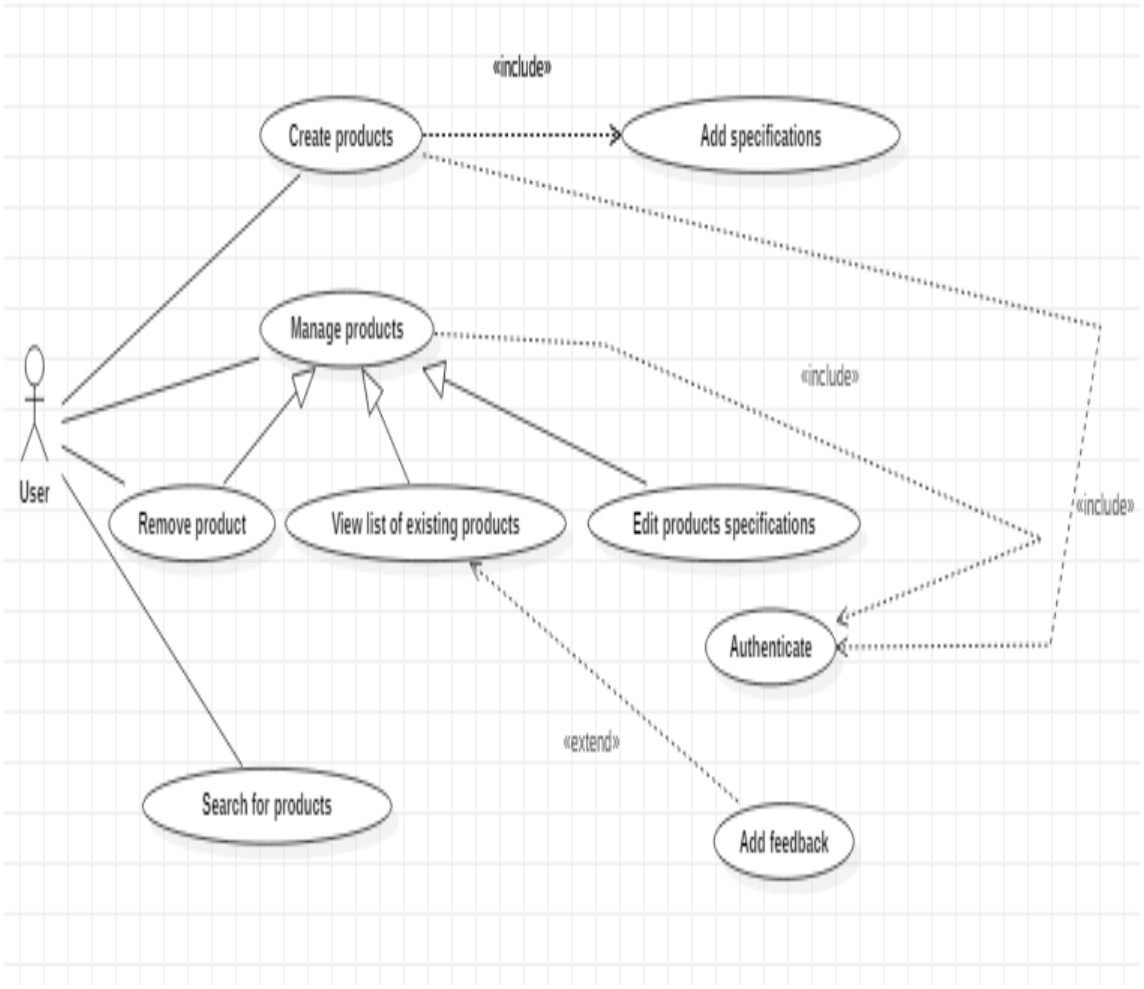


Figure 1.3 – Use case diagram for product Management Service

1.3.2.4 Point of Sale Management Service

The following diagram prevents the different use cases accomplished by the Point Of sale microservice .

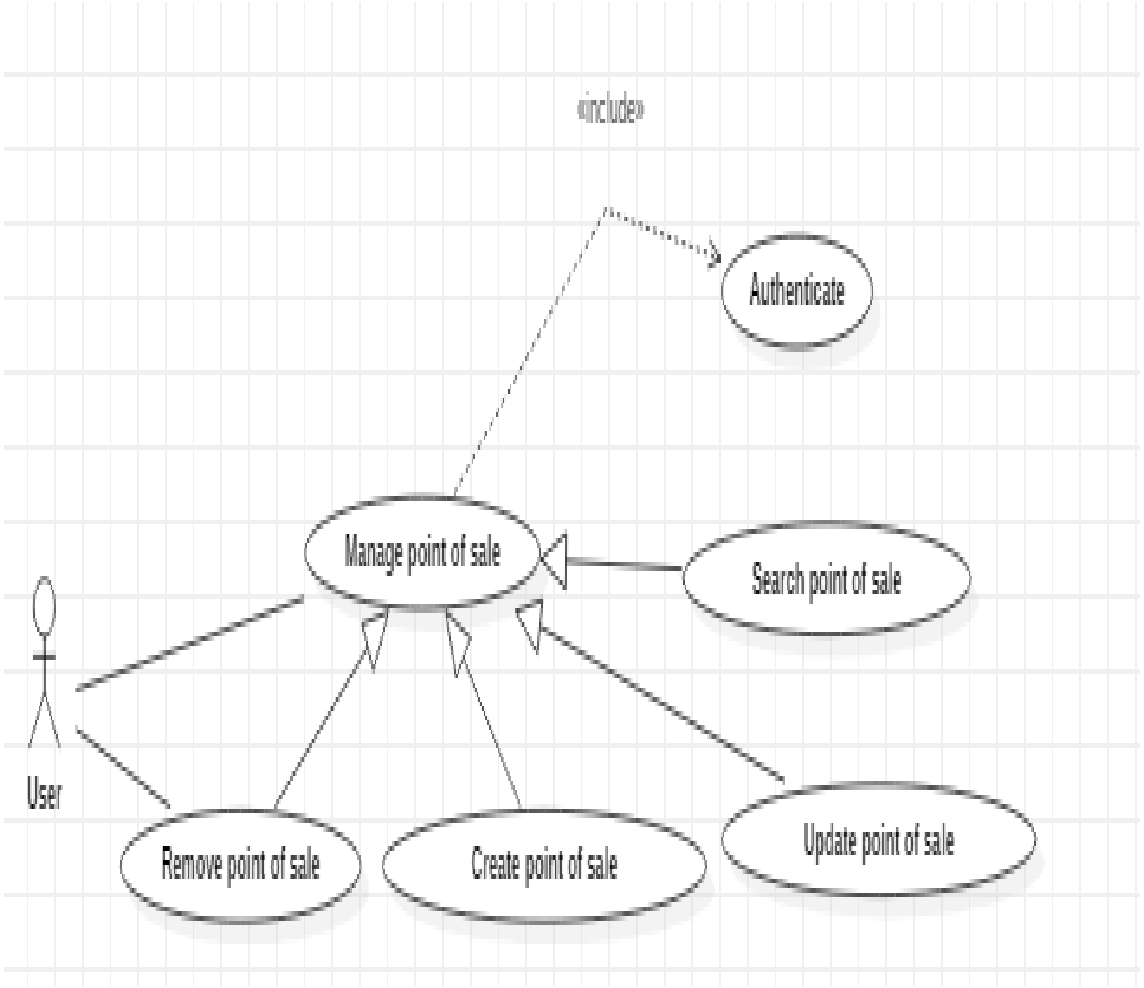


Figure 1.4 – Use case diagram for Point Of Sale Management Service

1.3.2.5 List Management Service

Figure 1.5 covers the use case diagram related to the list management microservice.

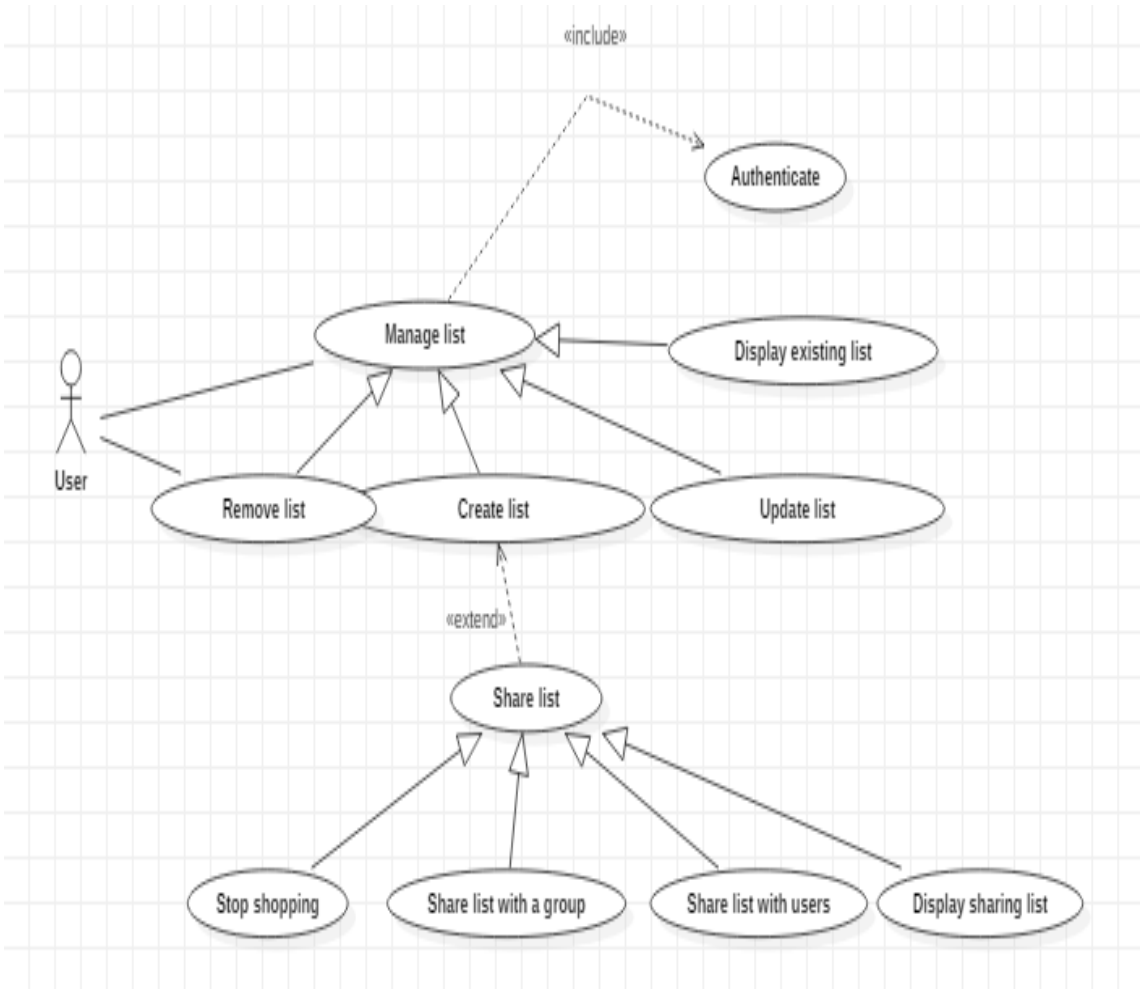


Figure 1.5 – Use case diagram for list Management Service

1.3.2.6 Use cases description

Table 1-6: User Microservice description

Action	Actor	Purpose	Post-condition
Sign up	User	Allow users to have access to the system's service	Account successfully created
Authenticate	User	Grant access for users to some of the application's features	User already has an account
Create group	User	Allow users to have access to the group creation feature	User logged in
Add user to a group	User	Allow users to add members to his own group	Group already created

Table 2-6: Product Microservice description

Action	Actor	Purpose	Post-condition
Add product	User	Allow users to a add a desired product	Application installed in the device
Validate a product	Admin	Validate a product added by a user based on specific criteria	Authentication as admin is required
Associate a product to a point of sale	User	Allow user to associate a new product to a specific point of sale(this will be not validate until it will validate by admin	Application installed in the device
Announce an offer on a product	User	Grant access for users to announce an offer on a product(this will be not validate an intelligent algorithm verify it based on user level-confidence)	Application installed in the device

Table 3-6: List Microservice description

Action	Actor	Purpose	Post-condition
Add list	User	Allow users to a add a desired list	Application installed in the device
Edit list	User	Allow user to edit its own list	Application installed in the device
Share a list	User	Give the permission to the user to share his list with a group or other user	Authentication
Delete a list	User	Grant access for users to delete a list	Application installed in the device
Attach a product to a list	User	Allow user to add a desired product into a list	Application installed in the device
Detach a product from a list	User	Grant access for users to detach a product from a list	Application installed in the device

1.3.3 Non-Functional requirements

Non-functional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs. Also known as system qualities, non-functional requirements are just as critical as functional Epics, Capabilities, Features, and Stories. They ensure the usability and effectiveness of the entire system.
The non-functional requirements that respond to our microservices:
-Functional completeness: The application must satisfy the expressed and clarified requirements under specific conditions of use.
Functional correctness: The application is expected to guarantee accurate results for the effective recommendation of purchased lists.
-Performance efficiency(Time behavior): The application needs to be able to manage concurrence. This is intended to complete a list of tasks as quickly as possible and to improve the overall responsiveness of the application.
-Maintainability(Analyzability + Reusability).

1.4 Product backlog

After listning the requirements of our application we outline in this section the product backlog as shown in table 3, but first let define a product backlog. As described in the Scrum Guide, the Product Backlog is an ordered list of everything that is known to be needed in the product. It is the single source of requirements for any changes to be made to the product.

Table 1-4: Product Backlog

ID	User stories	Priority
1	As a user, I want to create a new account Test the method create account using unittest	1
2	As a user, I want to login	1
3	As a user, I want to logout	1
4	As a user, I want to modify my profile Test the method edit profile using unittest	1
5	As a user, I want to create a group Develop the unitest of creating a group	1
6	As a user, I want to delete the group I have already created Test the method delete a group using unittes	1
7	As a user, I want to add members to the group I have already created Develop the unittest of adding members to the group	1
8	As a user, I want to remove members from the group I have already created	1
9	As a microservice I am asked to attach an account to a device Test the method attach an account to a device	1
10	As a microservice I am asked to detach an account to a device Test the method attach an account to a device	1
11	As a microservice I am asked to a add list Test the method add a list	1
12	As a microservice I am asked to edit a list Test the method edit a list	2
13	As a microservice I am asked to delete a list Test the method delete a list	2
14	As a microservice I am asked to attach a product to a list Test the method attach a product to a list	1
15	As a microservice I am asked to detach a product from a list Test the method detach a product from a list	1
16	As a microservice I am asked to share a list of products with a group or with a another user Test the method share a list of products with a group or with a another user	2
17	As a microservice list I am asked to use a microservice notification to remind the customer about his list Test this method	2
18	As a microservice I am asked to add a point of sale Using unittest to test the method add a point of sale	3
19	As a microservice I am asked to delete a point of sale Test the method delete a point of sale	3
20	As a microservice I am asked to modify a point of sale Test the method modify a point of sale	3
21	As a user of application I am to search a point of sale based on my dimension develop the unittest of search a point of sale based on my location	3
22	As a user of application I want to search a point of sale based on its designation develop the unittest of search a point of sale based on the designation of the point of sale	3
23	As a user I am want to get the nearest points of sale based on my location Test the method get the nearest points of sale based on the location of user	3

24	As a microservice I am asked to add products Test the method add products	4
25	As a user I want to consult available products Test the method consult available products	4
26	As an admin I want to delete a product Test the method delete	4
27	As an admin, I am asked to change the state of a product (validated or not validated). Develop the unittest of this method	4
28	As a a microservice I am asked to dissociate a product to a point of sale. Test the method dissociate a product to a point of sale	4
29	As a user I want to search a specific product Test the method of search a specific product	4
30	As a micorservice I am asked to add a category Test the method add category	4
31	As a micorservice I am asked to edit a category Test the method edit category	5
32	As a micorservice I am asked to delete a category Test the method delete category	5
33	As a user I want to consult a list of category Test the method consult categories	5
34	As a micorservice I am asked to add a new offer Test the method add offer	5
35	As a micorservice A am asked to validate an offer Test the method validate an offer	6
36	As a micorservice I am asked to delete a offer Test the method delete offer	6
37	As a micorservice I am asked to edit a offer Test the method edit offer	6

1.5 Sprint planning

Based on our product backlog, we split our release into 4 sprints as indicated in table 1-5. This plan is created by the collaborative work of the entire scrum team.
During each sprint, we will be developing several micorservices.

Table 4: User stories partitioning per sprint

	Sprint 1	Sprint 2	Sprint 3	Sprint 4
User story ID	1,2,3,4,5,6,7,8,9,10	11,12,13,14,15,16,17	18,19,20,21,22,23	24,25,26,27,28,29,30,31,32 33,34,35,36,37

1.5.1 Technical restrintions

The implementation of the aplication Smart List requires certain technical constraints:

-Techologies used: Regading the server-side implementation, the microservices will be developped using Python and Flask as framework for the back-end and PHP and Laravel for the front-end.

-Architecture: As recommended by the host company, we must use a microservice-based architecture.

Conclusion
Throughout this chapter, we have exposed the main context of this project. We have focused on identifying functional and non-functional requirements.Then, we presented the product backlog adopted for the development of the required tasks and planned our sprints.The next step we will tackle the conceptuel design of our microservices

CHAPTER 2

Conceptual design

Introduction

In this chapter we present at the beginning the microservice architecture the design phase of our microservices. We will detail the class diagram, the package diagram, the activity diagrams of the different microservices.

2.1 Technical design

This section contains a technical description of the solution to the requirements outlined the previous chapter.
It covers two major parts: the overall conceptual design and the detailed design.

2.1.1 Overall design

The part below will show the decomposition of Smart List into microservices.
In fact, to proceed from a monolithic architecture to microservice architecture, we follow the pattern of the bounded context of the domain driven design.
We present in the figure below the context map of our system.

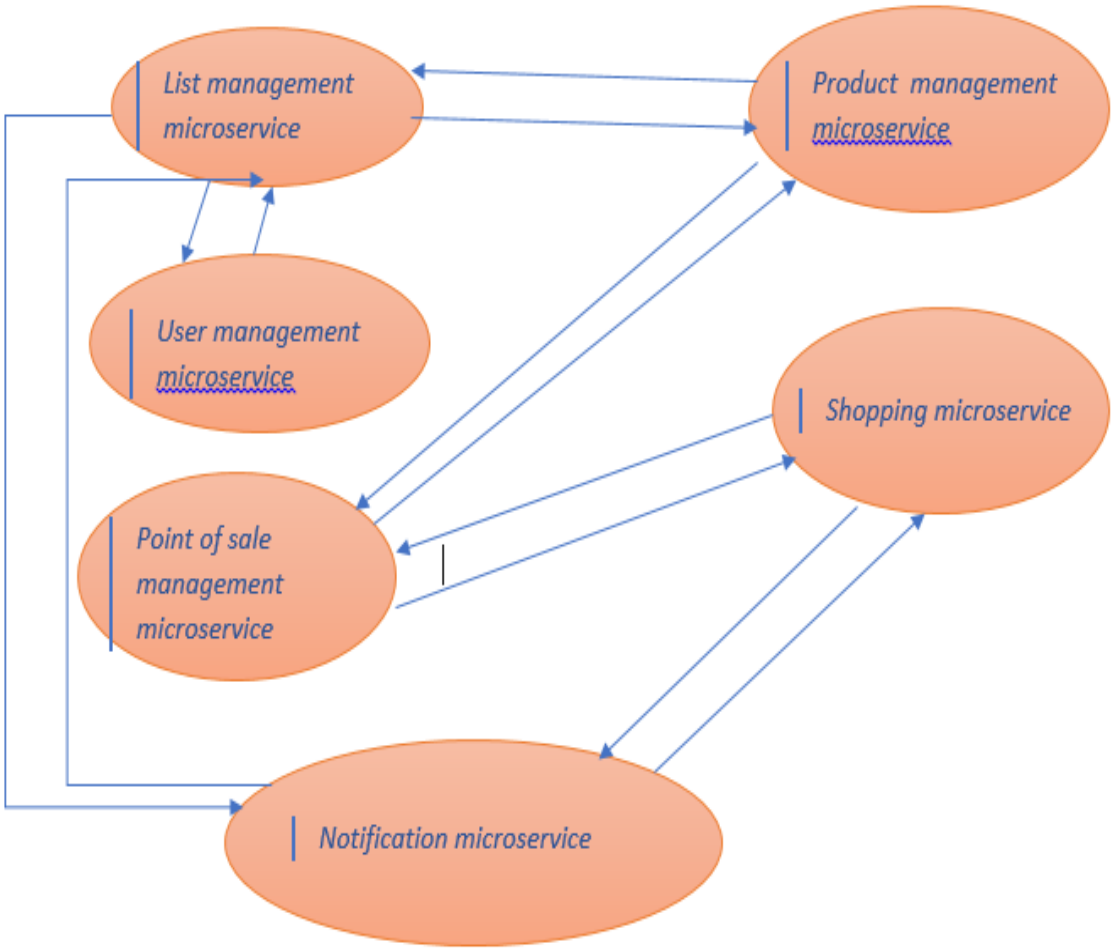


Figure 2.1 – Smart List context map

The application of bounded context pattern has given us six microservices:

-User management microservice

-Product management microservice

-List management microservice

-Point of sale management microservice

-shopping microservice

-Notification microservice

2.1.2 Microservice architecture

The term micro service has emerged in recent years to describe a particular style of architecture. this approach consists of developing a single application into a set of isolated and independently autonomous one.

Those services can communicate together to provide the necessary functionality. The microservices are in most cases implemented and driven by teams of small sizes with sufficient autonomy. each can change the implementation of each microservice, add or remove features of this service with minimal impact on other microservices.
This architectural style has several advantages such as technological heterogeneity, resistance to failure, custom scalability, ease of deployment, organizational alignment, reusability

2.1.2.1 Characteristics of microservice architecture

According to MARTIN FOWLER, the microservice architecture has nine main features that are essential to apply during the design and development of a microservice application.

-The division by dialing through the services: this feature is inherited from the component-based architecture. the services make it possible to avoid strong coupling between the different components by using explicit remote call mechanisms.

-The organization around business skills: Each microservice is autonomous with respect to the functionality that it realizes since it has its own code, its own interface and manages its own data.

-A product, not a project: In the microservices vision, a team is responsible for a product throughout its life cycle. She is entirely responsible software in production.

-Decentralized governance: With microservice architectures, we can use for each service the the most appropriate implementation language and technology platform to accomplish the need.

-Decentralized data management: The architecture in microservices admits the use of several databases. In the context of a monolithic application, we have only one database logic for persistent entities while the framework of an applcation in microservices, each service has its own conceptual model and manages its own database.

-Smart ends and stupid channels: With microservices, the use of stupid communications is favored. These non-intelligent communications only transmit the messages, while the microservice does the rest. Intercommunication between microservices via open protocols is privileged and many others interact with each other others via REST calls or through queue systems.

-Automation of infrastructure: Enterprises, who has migrated to microservice architecture, have gained experience in delivery continuous and continuous integration and they now use automation tools infrastructure.

-Design for failure: One of the major strengths of microservices is that they are designed to be tolerant to breakdowns. In a microservice application, if one service fails, the other services are not affected and adapt their operations according to the state of the system in which they evolve.

-An evolutionary design: One of the key elements in microservice architecture is the notion independence and scalability. In general, the evolution of an application is the addition of new features that results in the creation of new microservices and or by updating existing services which only involves updating and redeployment of the microservice concerned.

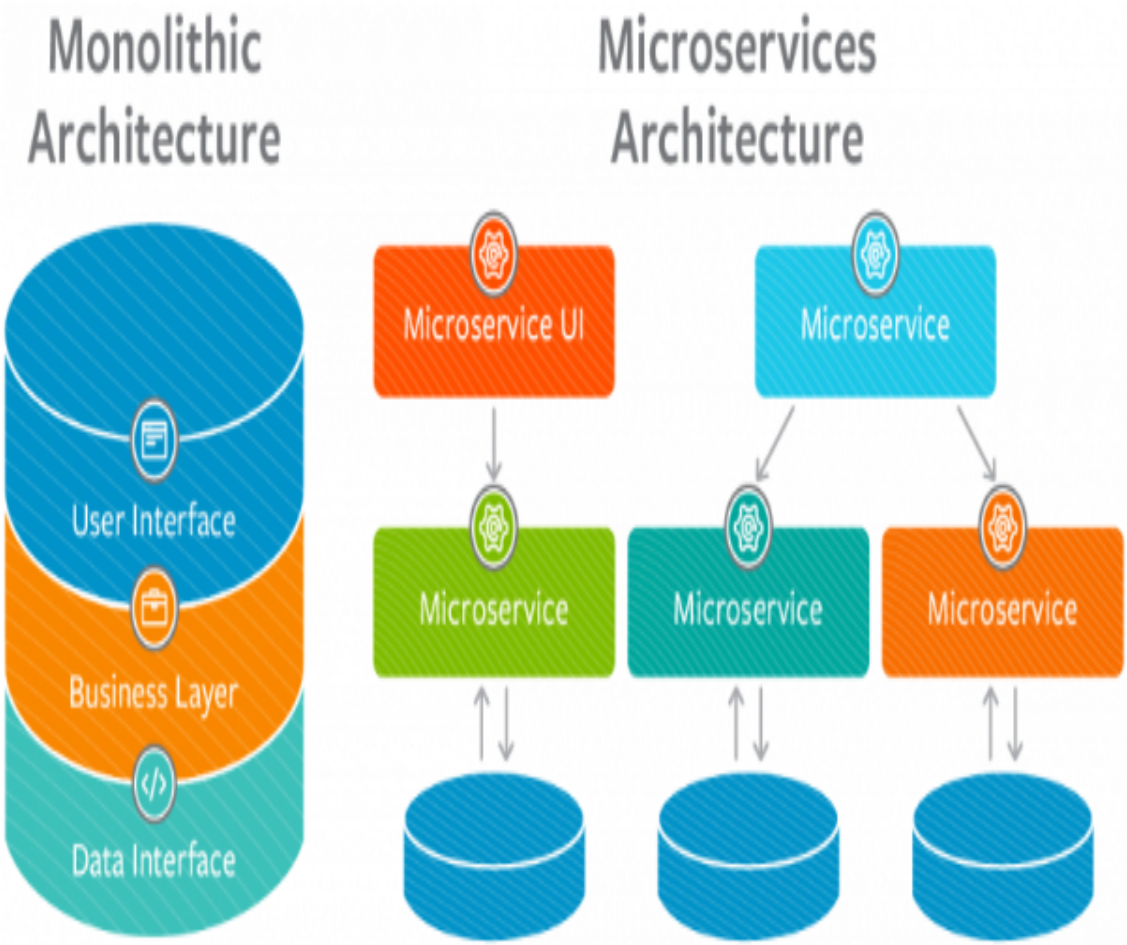


Figure 2.2 – Comparision between monolithic and microservice architecture

2.1.2.2 Motivations for the use of microservices

Several motivations push to use this architectural style:

- The frustration of not achieving the desired result with monolithic architectures in large projects encourages us to face the challenges of microservices.

- The emergence of new automation tools for testing, monitoring and deployment allow us to see microservices in a new light and remove some of the problems that their development and deployment had created.

- The use of microservices by large companies like Amazon, Netflix, eBay and others, give enough confidence that this style of architecture is ready to be evaluated and used by professional application developers.

2.1.2.3 Benefits of microservices architecture

-Reducing the time to market or "time to market".

- Technological agility.

- Extensibility.

- Factoring and reuse of microservices.

- Scalability.

2.1.2.4 Disadvantages of microservices architecture

Despite their advantages, microservice architectures are the subject of several criticisms and have some disadvantages. We can quote the most recurrent ones:

- The complexity of a distributed system.

- Higher initial cost.

- Deployment by microservices.

- Essential and complex monitoring.

2.1.3 Detailed design

The detailed design includes the class, activity and deployment diagrams.

2.1.3.1 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

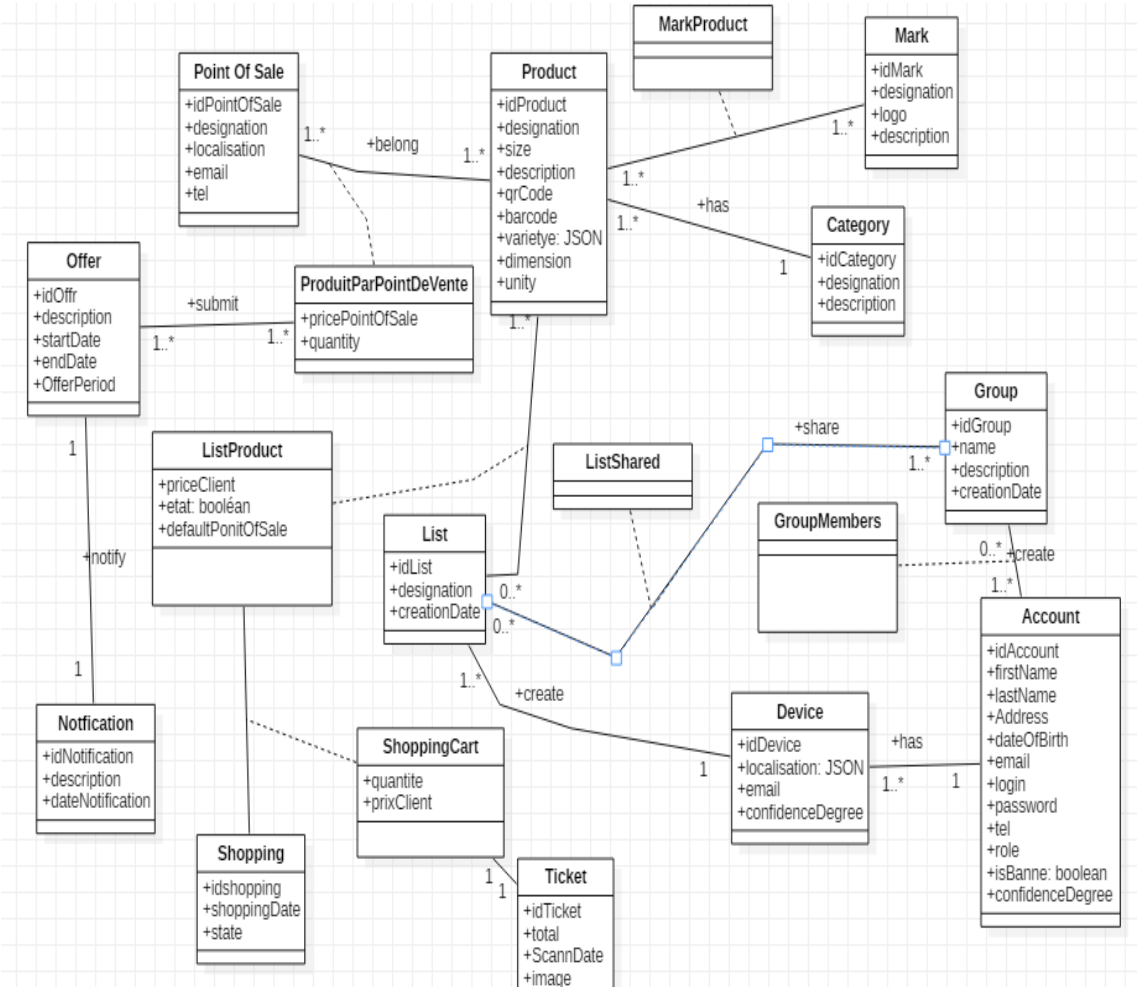


Figure 2.3 – General class diagram

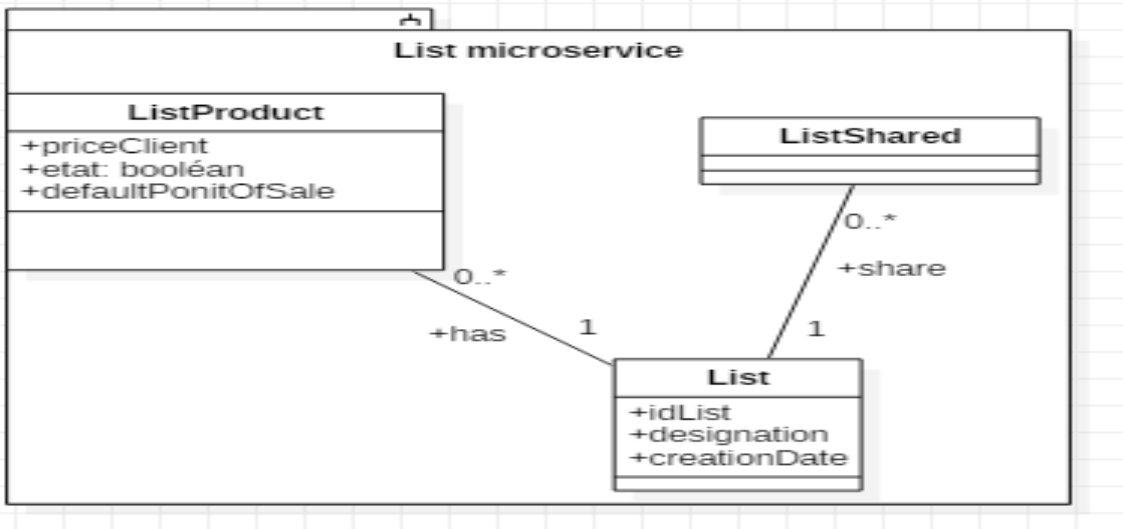


Figure 2.4 – List management microservice class diagram

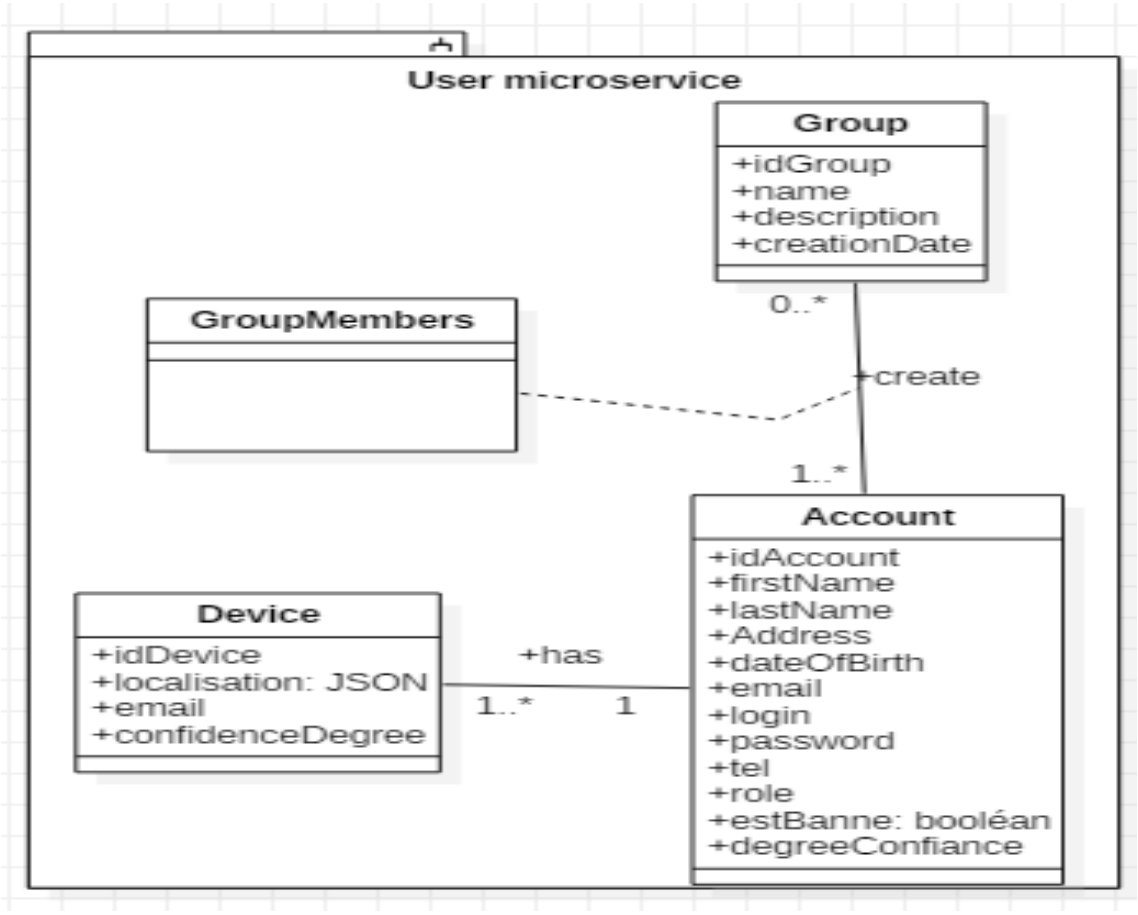


Figure 2.5 – User management microservice class diagram

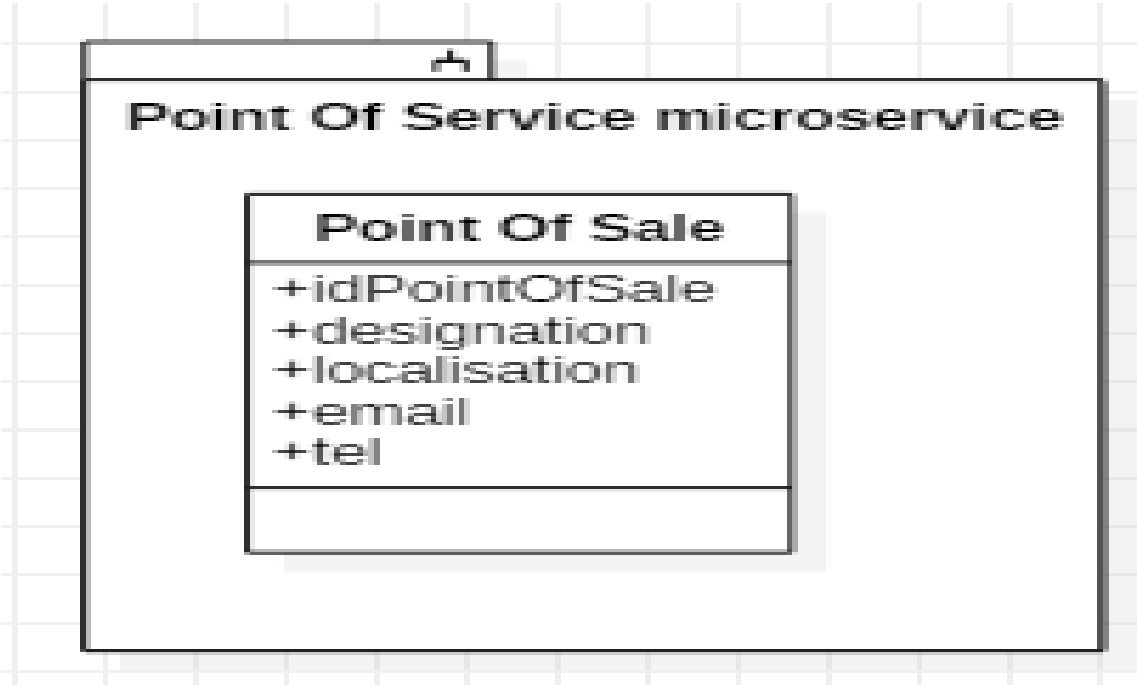


Figure 2.6 – Point Of Sale management microservice class diagram

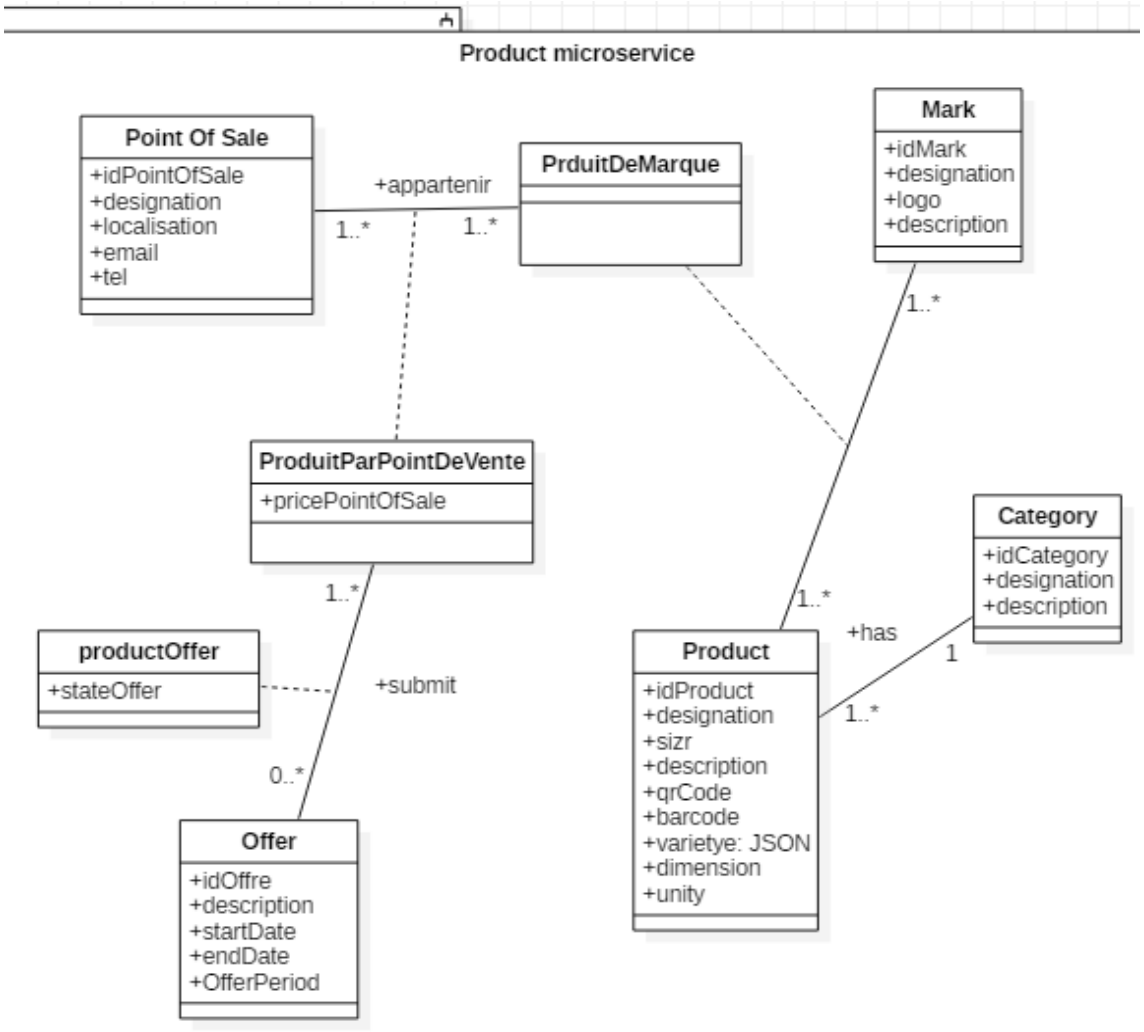


Figure 2.7 – Product management microservice class diagram

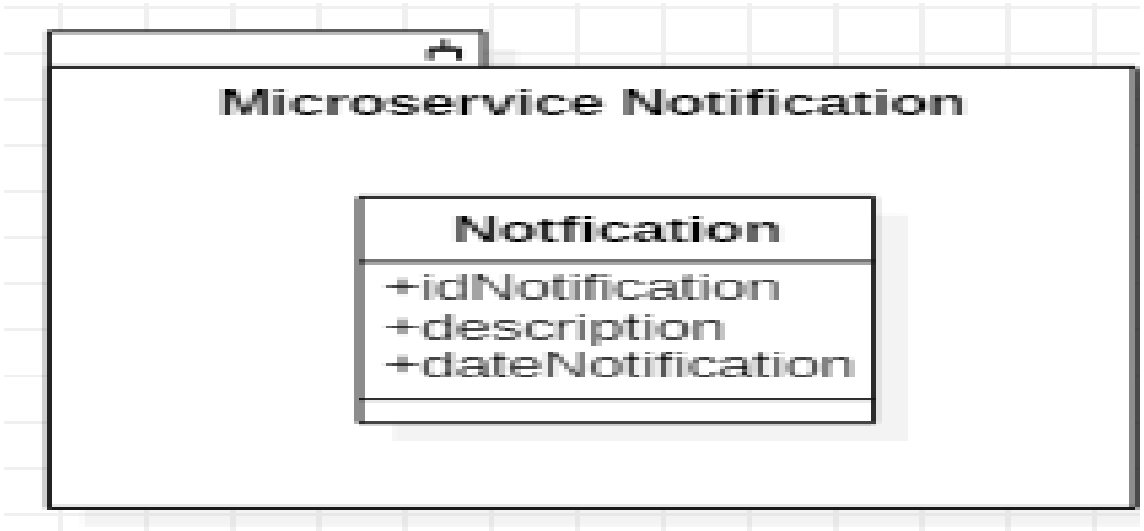


Figure 2.8 – Notification microservice class diagram

2.1.3.2 Activity Diagram

Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

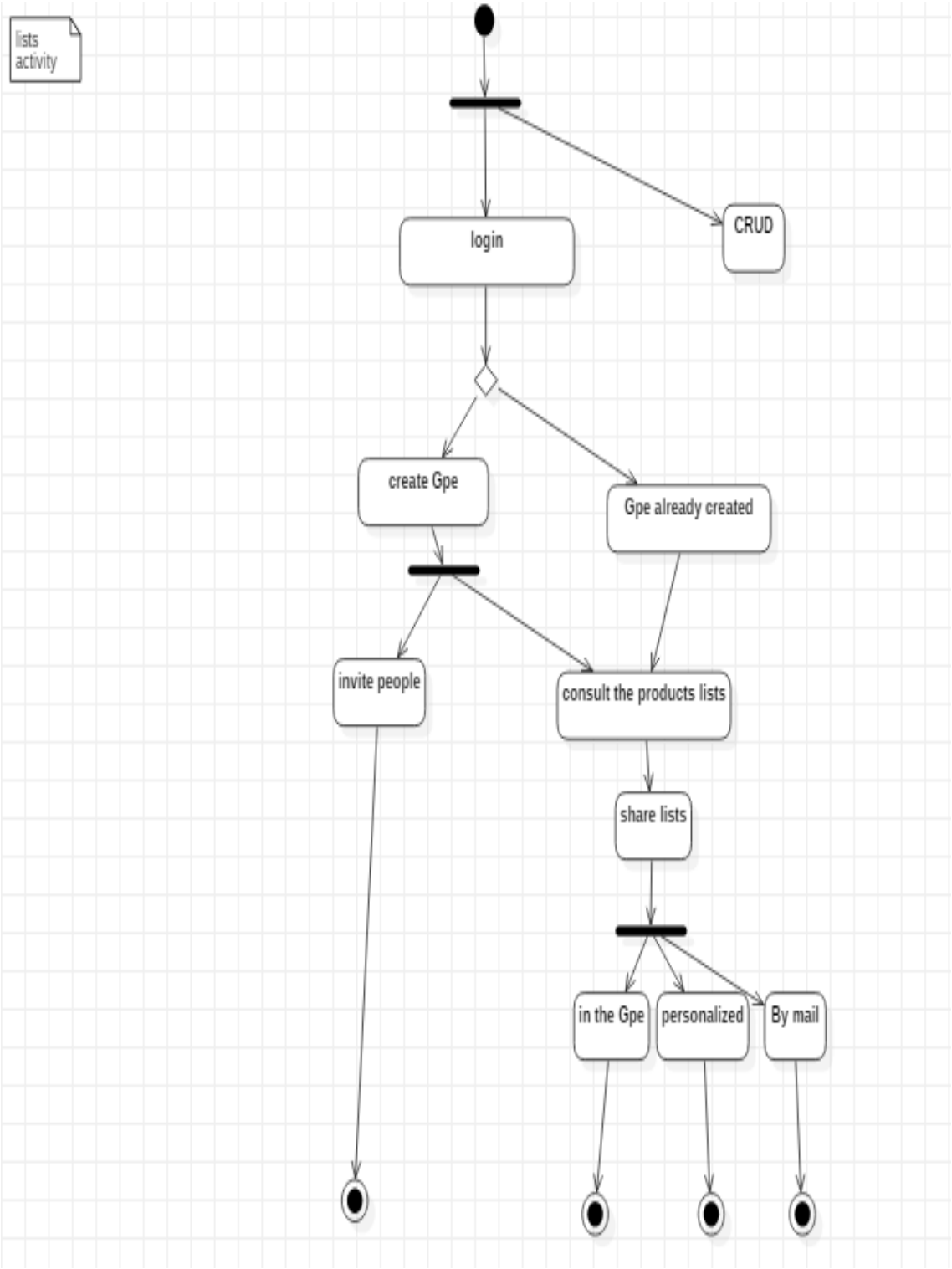


Figure 2.9 – User management microservice activity diagram

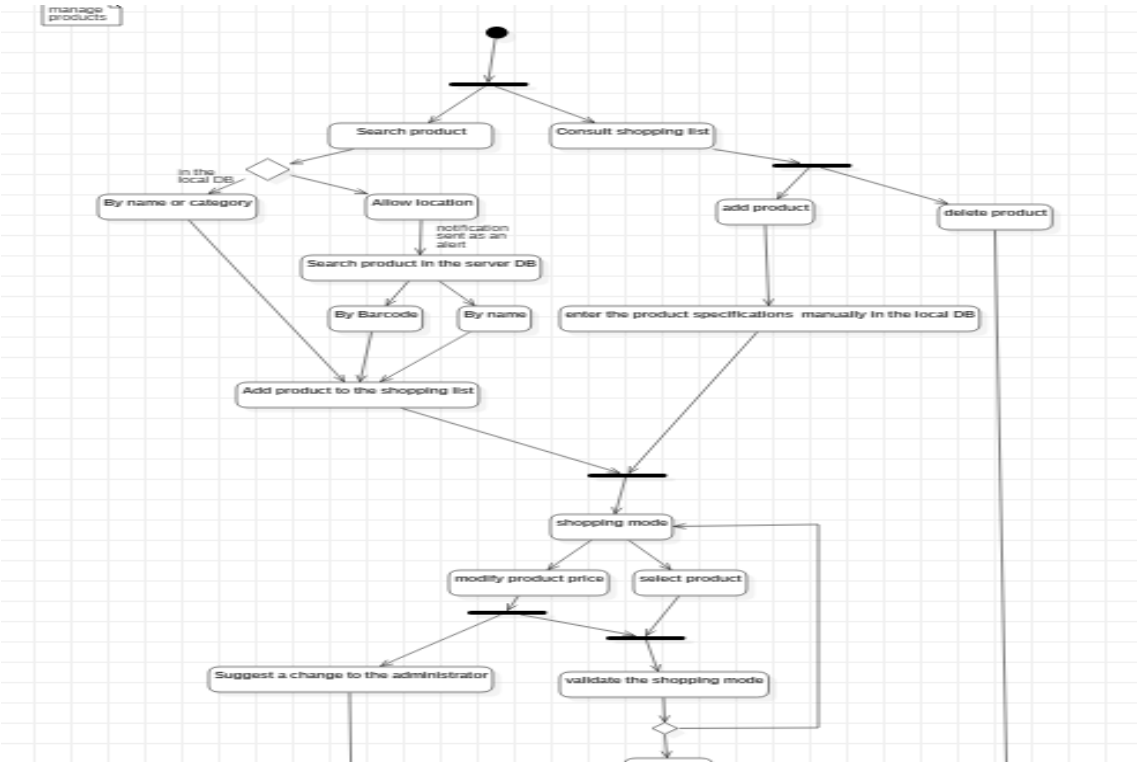


Figure 2.10 – Product management microservice activity diagram

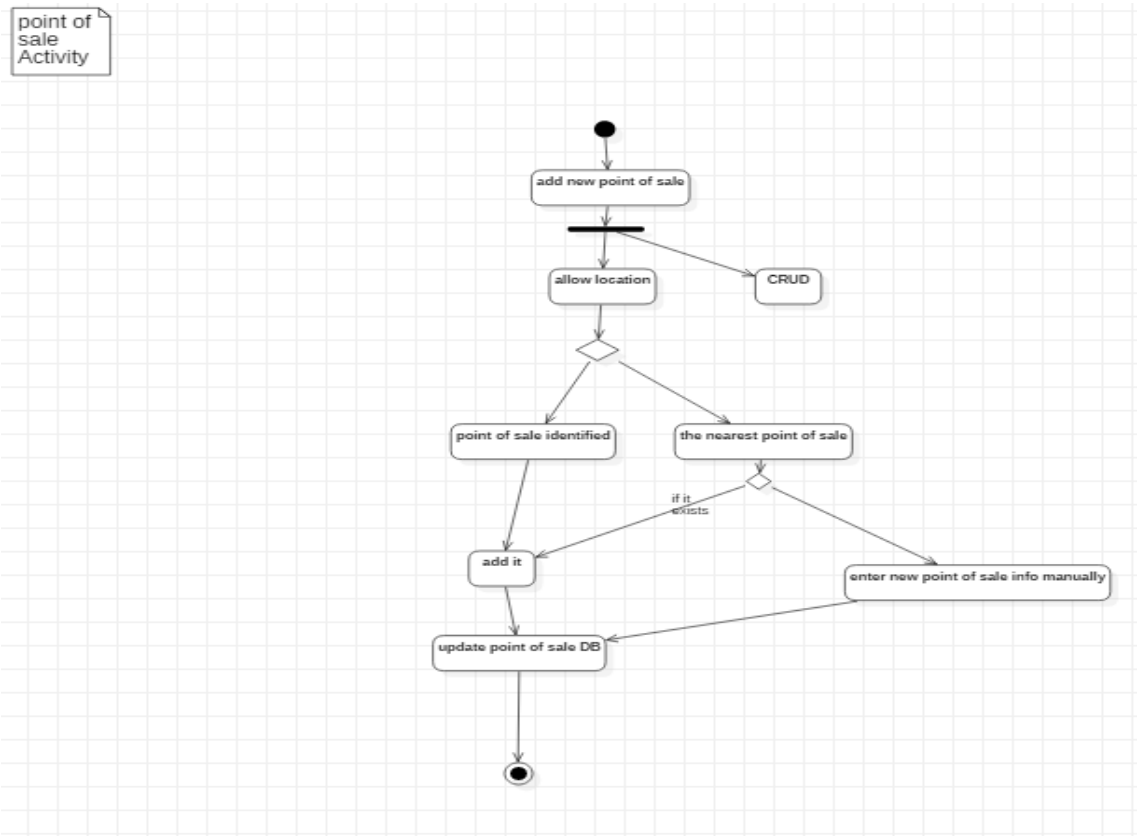


Figure 2.11 – Point of sale management microservice activity diagram

2.1.3.3 Deployment diagram

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware).

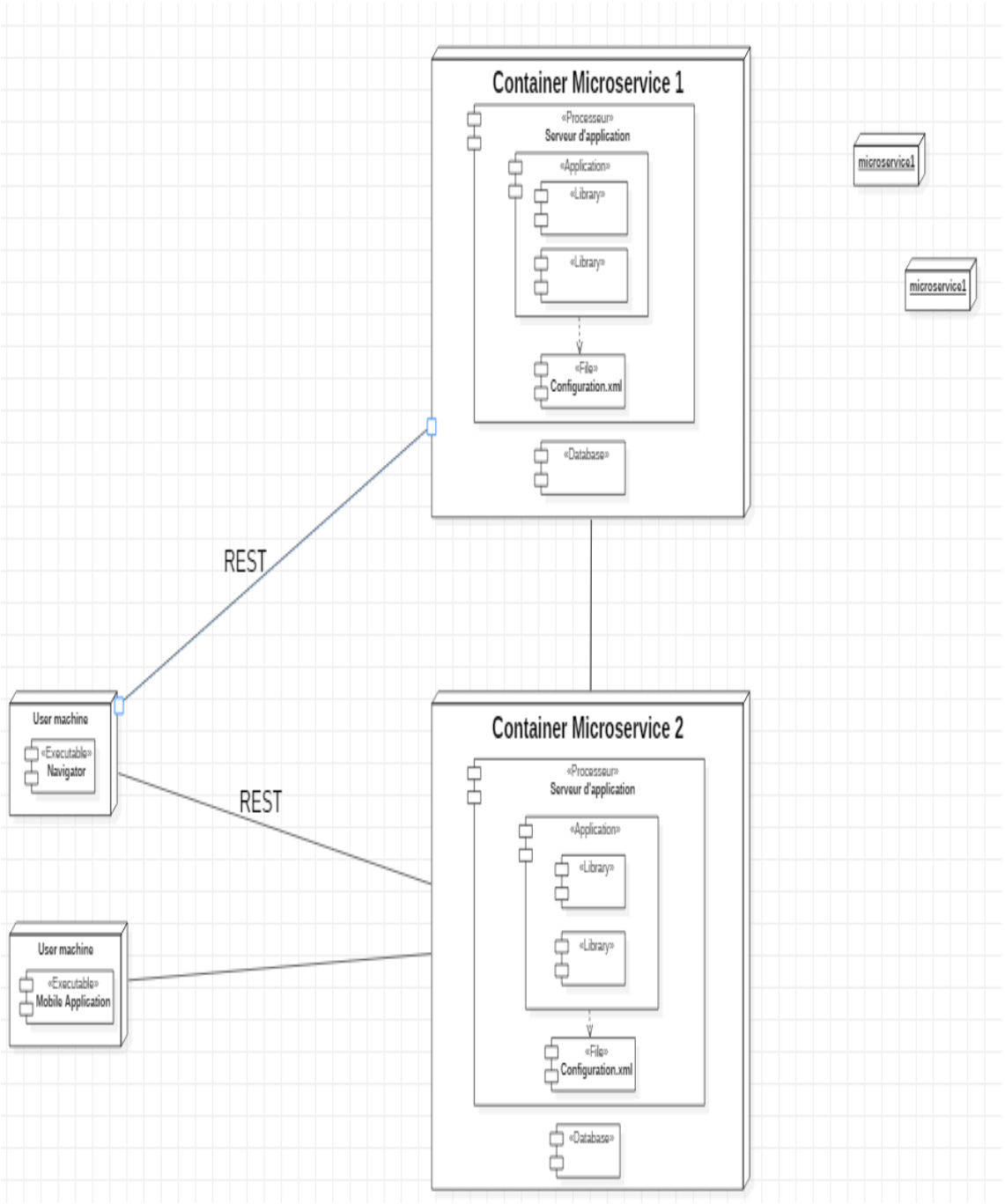


Figure 2.12 – Deployment diagram

Conclusion

In this chapter, we have established a global view on the microservice conceptual design, Throughout the detailed design, these microservices have been treated from various aspects: structural via class diagrams and behaviral through activity diagrams.
The following chapter will contain the different parts of the realization along with the integration of the microservices.

CHAPTER 3

Implementation

Introduction

With the prototype modelling approach for the microservices and the conceptual choice, we can proceed in this last chapter with the implementation and test phase to ensure that the proposed approach is fully validated.

3.1 Work environment

The work environment section presents the hardware and software components that have been part of the implementation of the current solution.

3.1.1 Hardware environment

To carry out the realization, we used as a hardware environment, a computer with the following characteristics:

-Processor: 2.7 GHz Intel Core i5.

-RAM: 8GO.

-Hard disk: SSD 256GO.

3.1.2 Software environment

Now, we present the software environment and the technologies used.

3.1.2.1 Deployment side

-Operating system: macOS Mojave 10.14.6 (18G103).

-Saas platform: Docker.

-Design tool: StarUML.

-Version control tools: Git and Gitlab.

3.1.2.2 Development side

-Integrated development environment (IDE): JetBrains PyCharm.
-Testing tool: Postman

3.1.3 Technologies used

-python 3 : Python is a general-purpose, versatile and popular programming language. It is the fastest growing and most popular programming language used by software engineers, analystes, data scientists, and machine learning engineers alike.

- Flask: Flask is a lightweight WSGI web application framewoek. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper arround Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

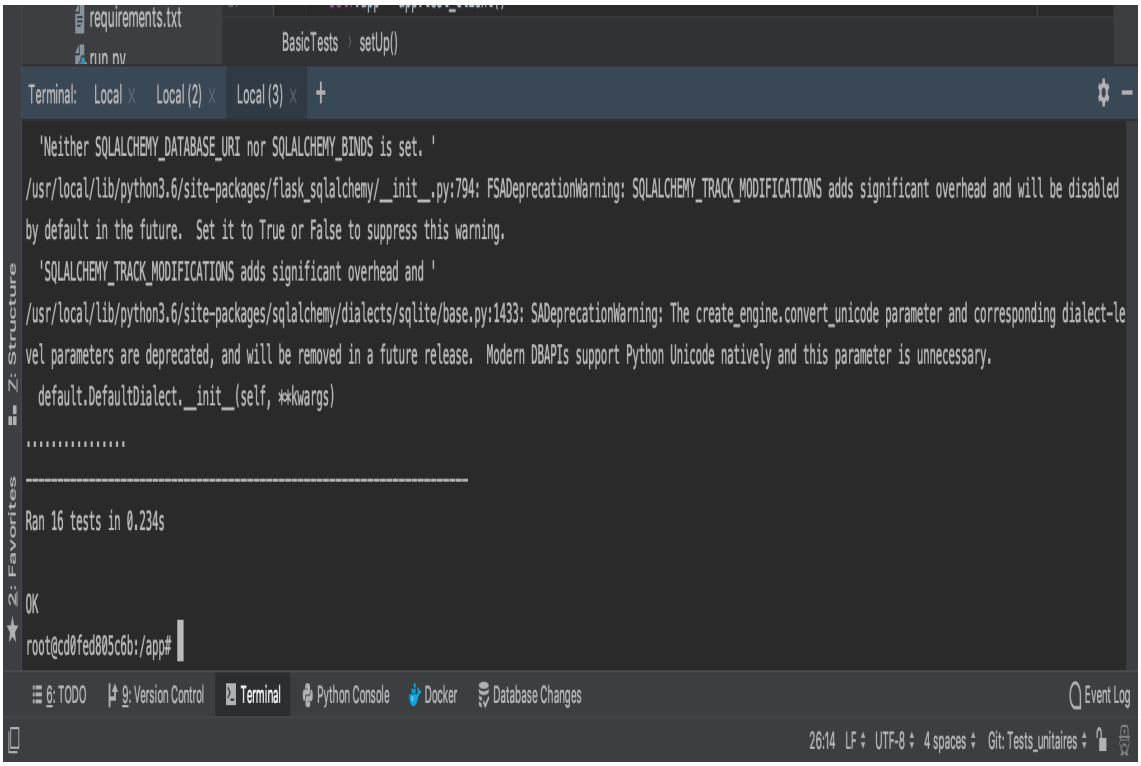
-MongoDB: MongoDB is a document-oriented, open-source database program that is platform-independent. MongoDB, like some other NoSQL databases (but not all!), stores its data in documents using a JSON structure. This is what allows the data to be so flexible and not require a schema.

-SQLAlchemy: Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy to your application. It aims to simplify using SQLAlchemy with Flask by providing useful defaults and extra helpers that make it easier to accomplish common tasks.

-Hive: Apache Hive is a data warehouse system built on top of Apache Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in various databases and file systems that integrate with Hadoop, including the MapR Data Platform with MapR XD and MapR Database. Hive offers a simple way to apply structure to large amounts of unstructured data and then perform batch SQL-like queries on that data.

3.2 Unittest

Unit testing simply verifies that individual units of code (mostly functions) work as expected. Usually you write the test cases yourself, but some can be automatically generated. Performing unit tests is designed to be simple, generally the tests are written in the form of functions that will determine whether a returned value equals the value you were expecting when you wrote the function.



unittest.png

Figure 3.1 – Microservice management user: Client unittest

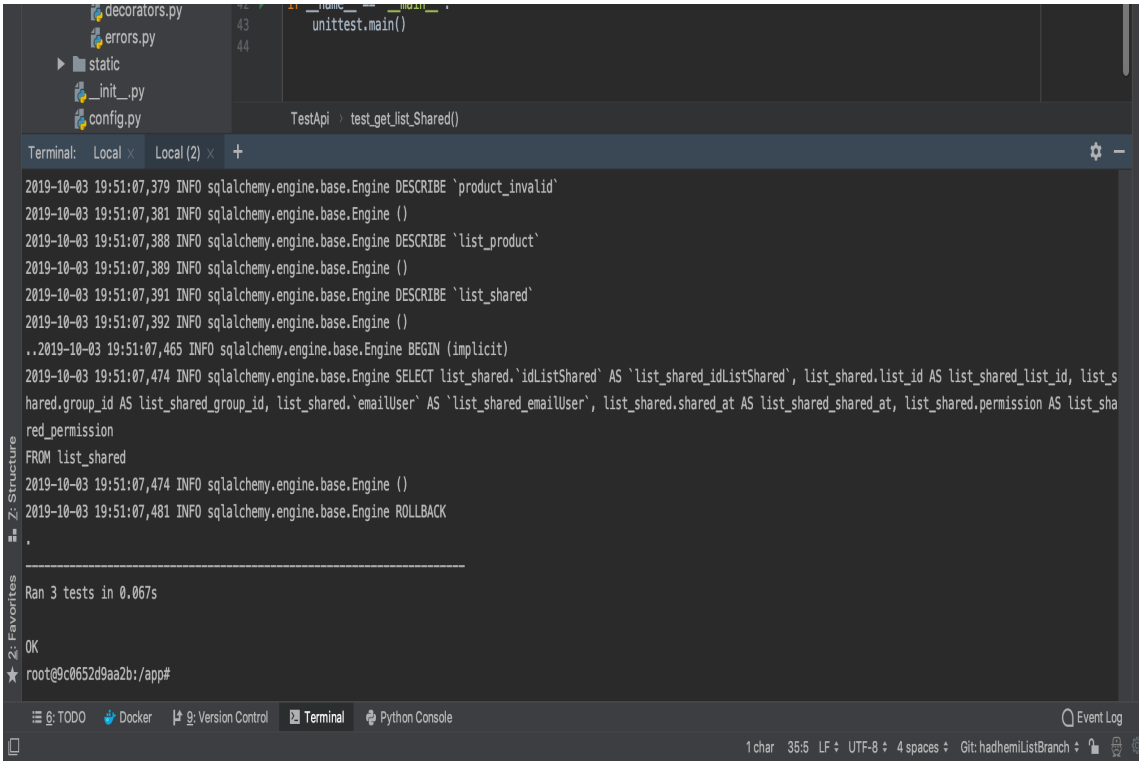


Figure 3.2 – Microservice management ListShared: unittest listShared

3.3 Smart List functionalities

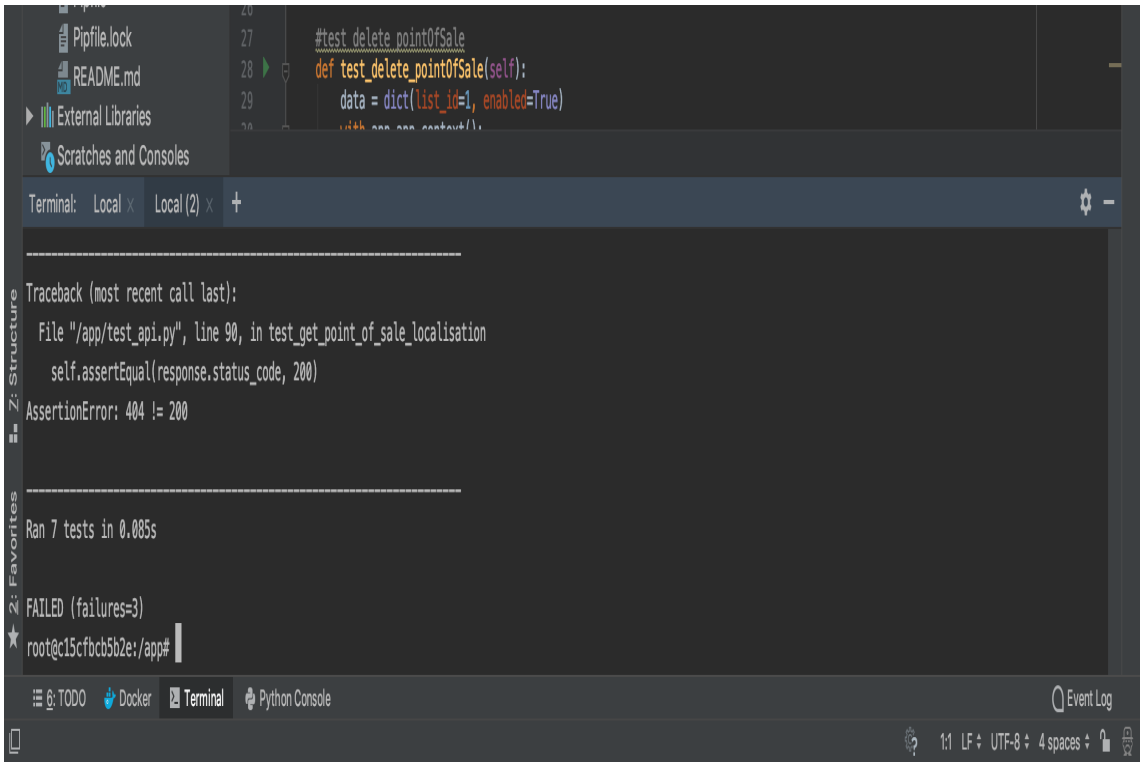


Figure 3.3 – Microservice management Point of sale: unittest PointOfSale

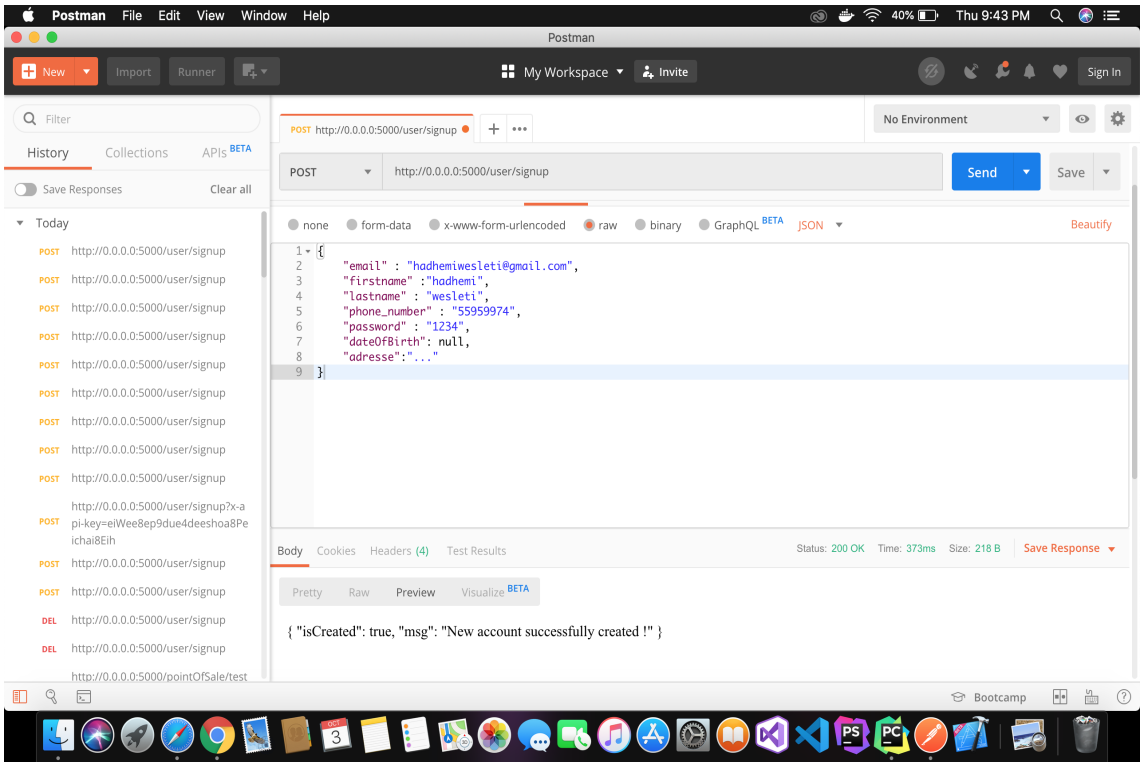


Figure 3.4 – Microservice management user: Signup

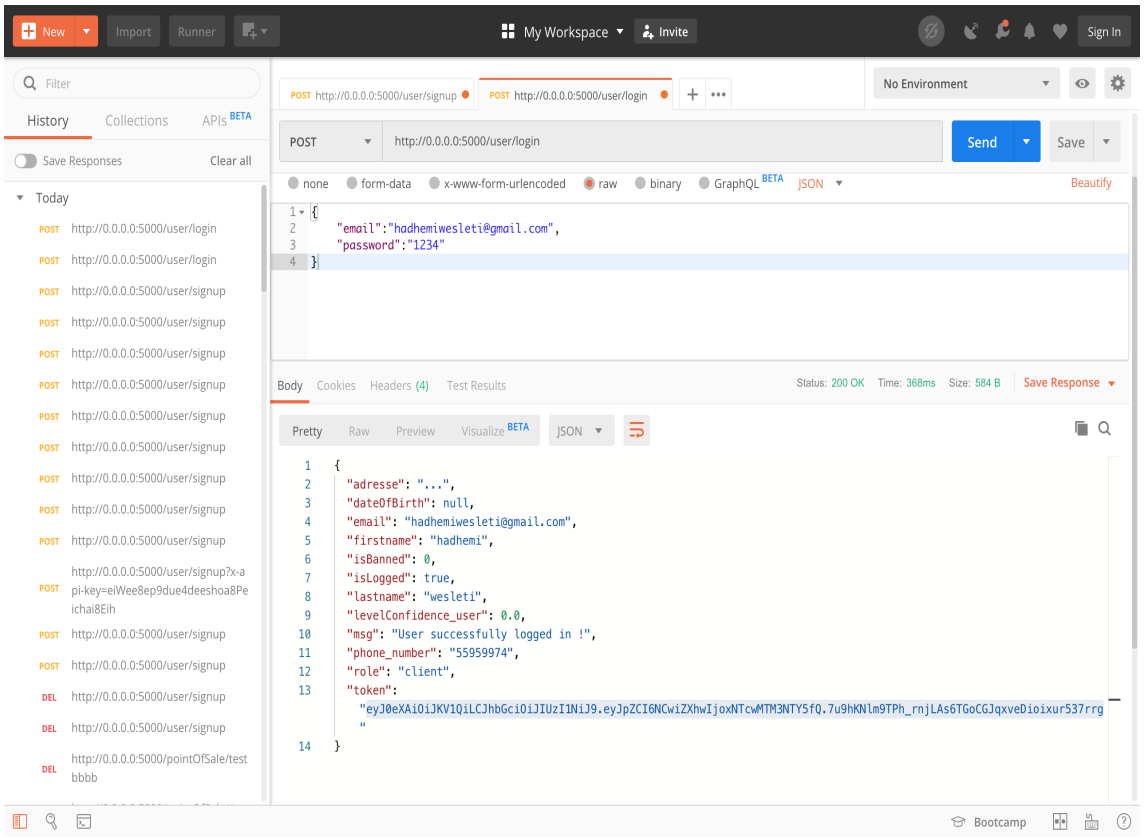


Figure 3.5 – Microservice management user: Login

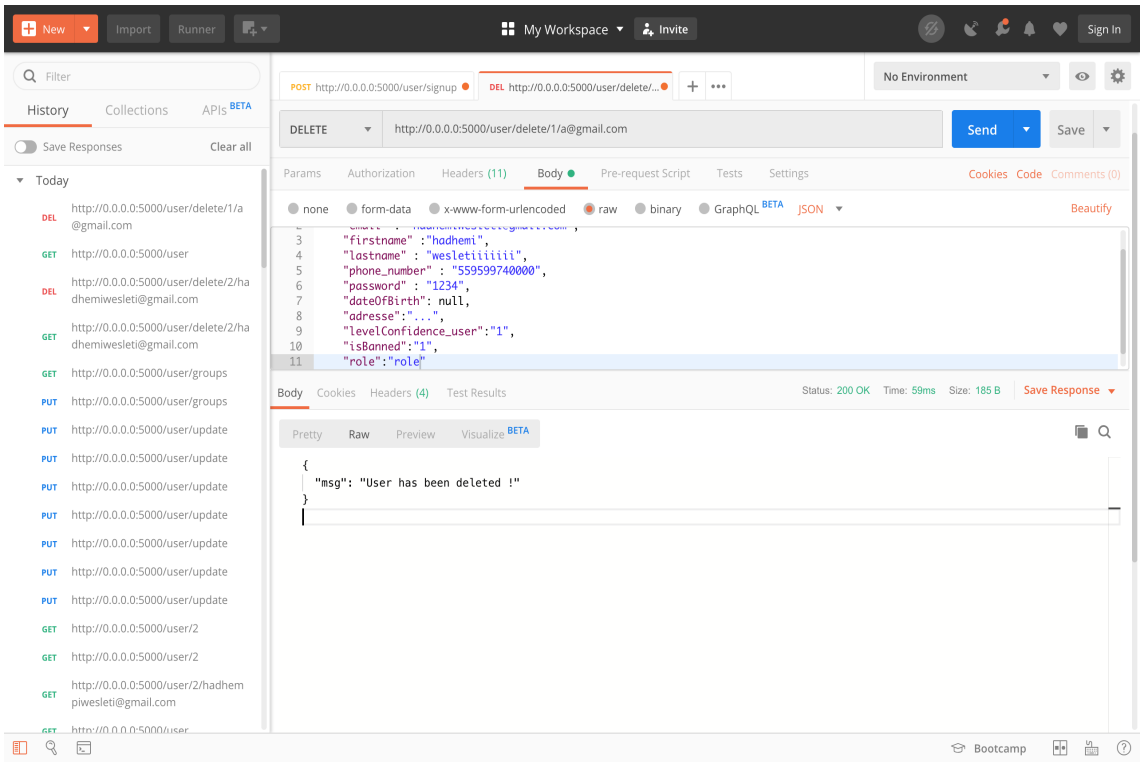


Figure 3.6 – Microservice management user: Delete user

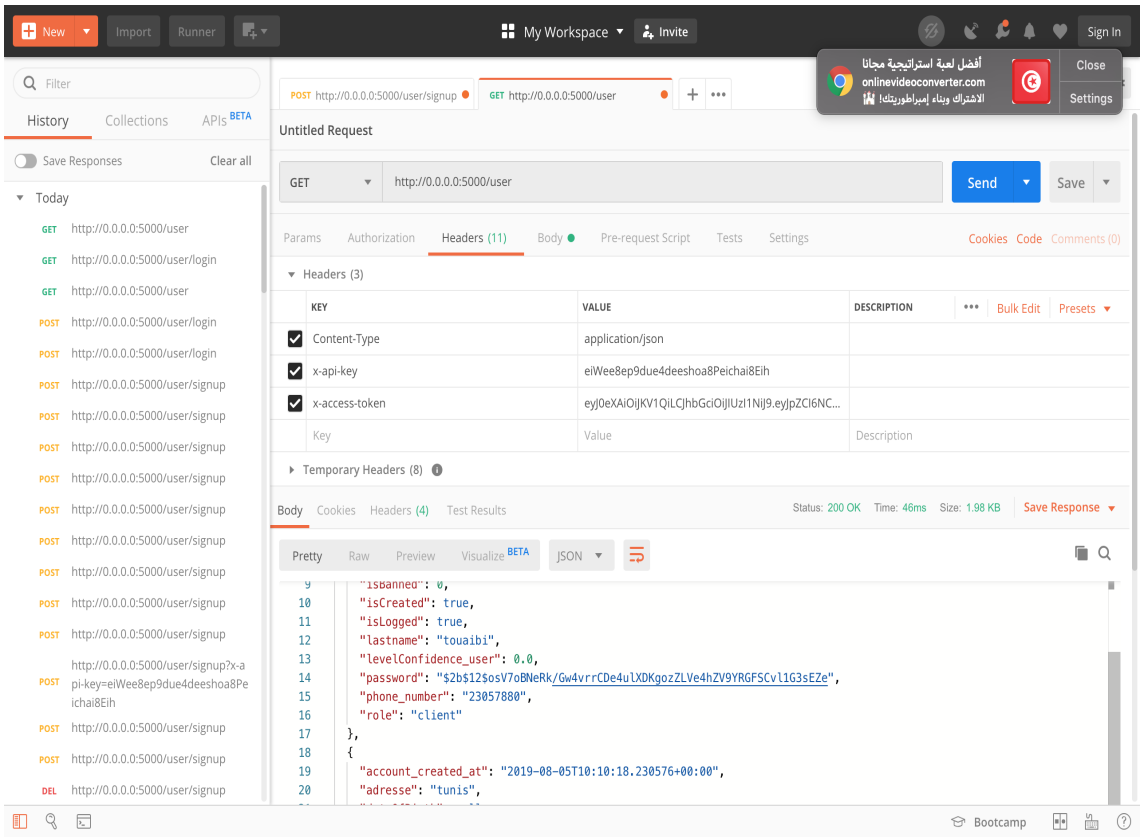


Figure 3.7 – Microservice management user: Get all users

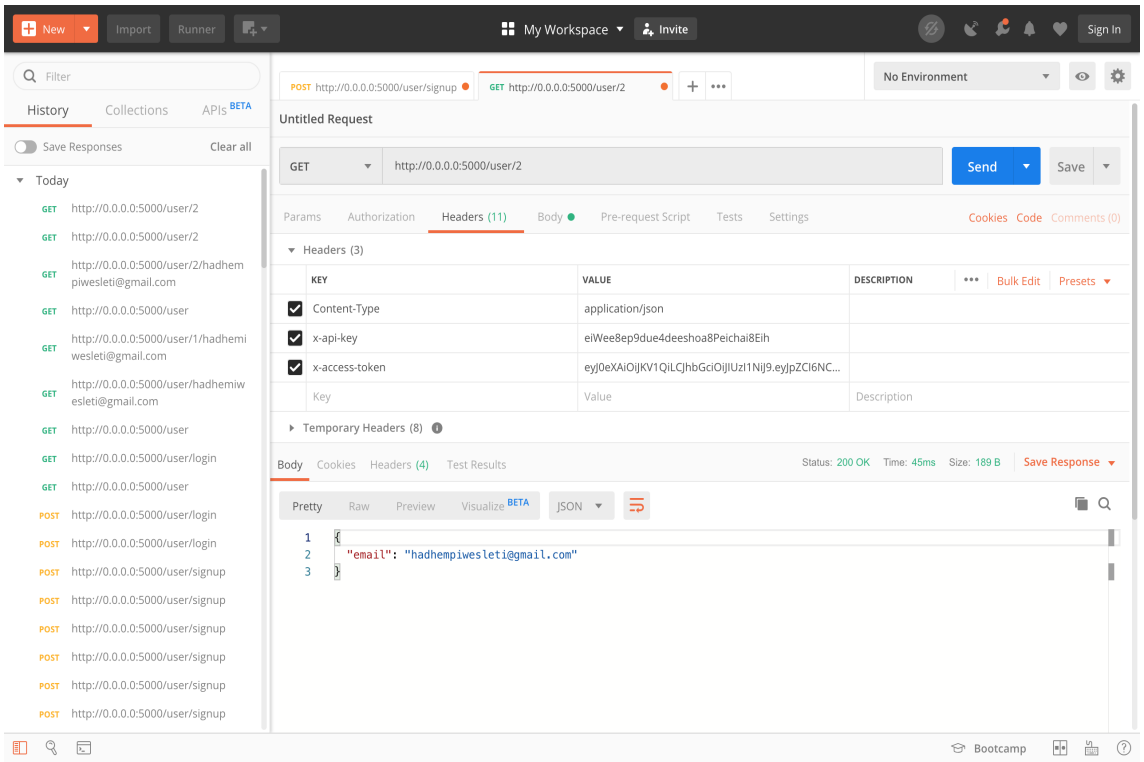


Figure 3.8 – Microservice management user: Get single user by email and id

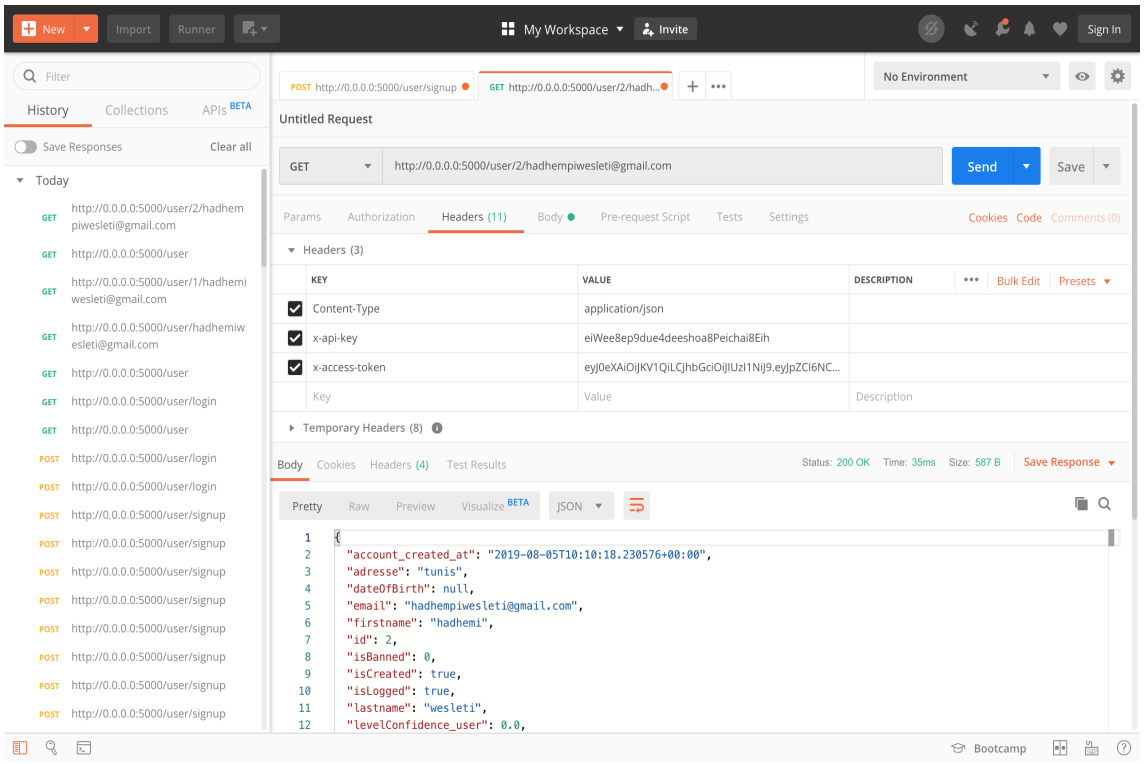


Figure 3.9 – Microservice management user: Get single user by id

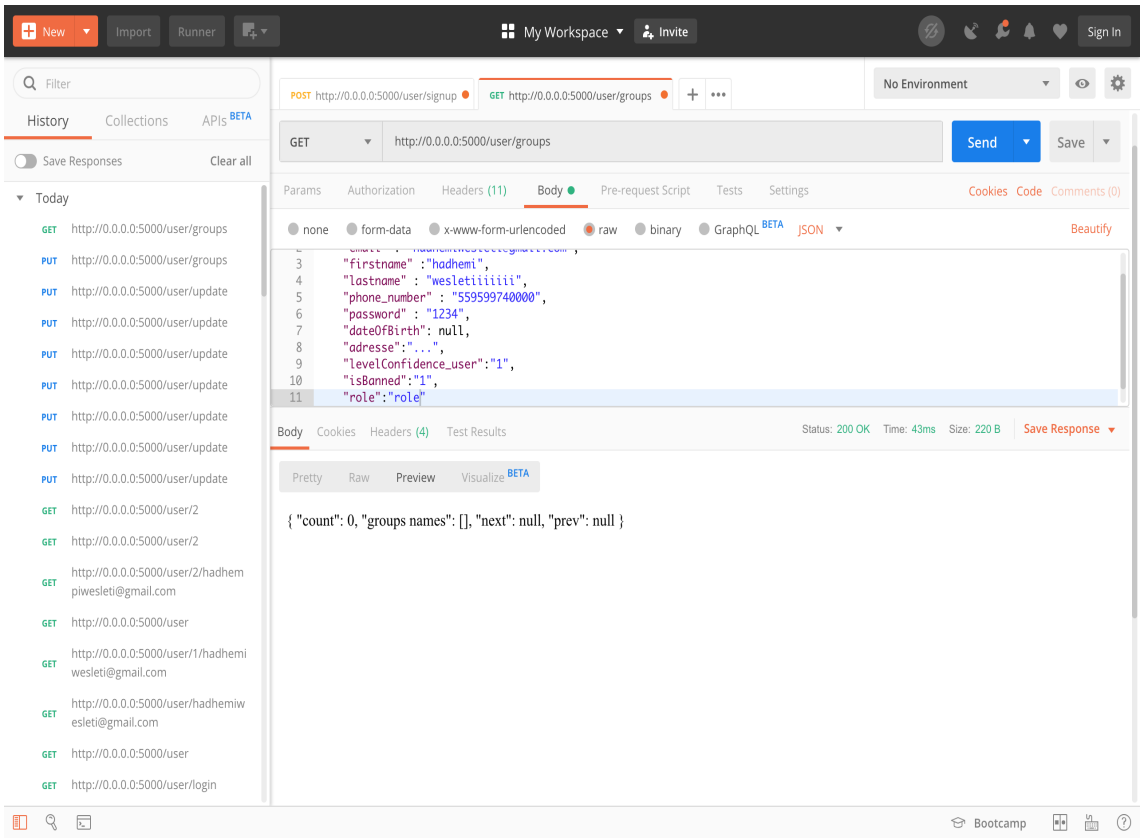


Figure 3.10 – Microservice management user: Get user who crete a group

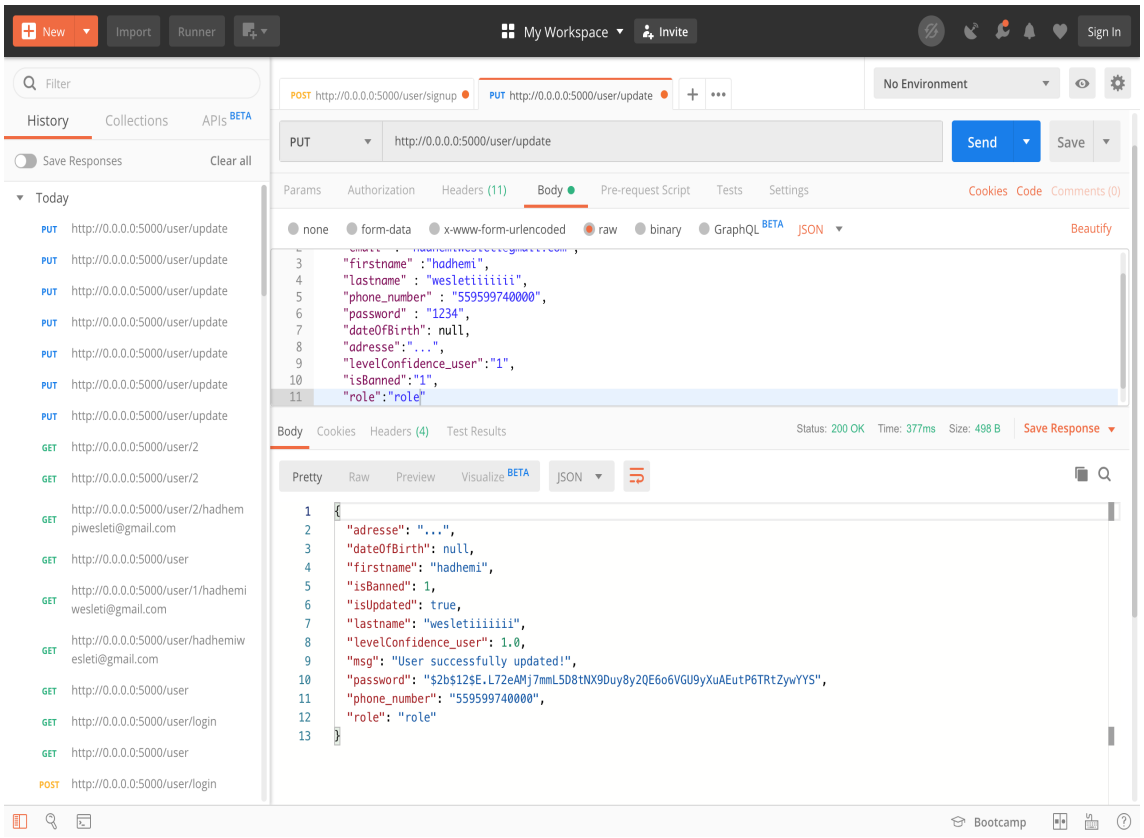


Figure 3.11 – Microservice management list: Update user info

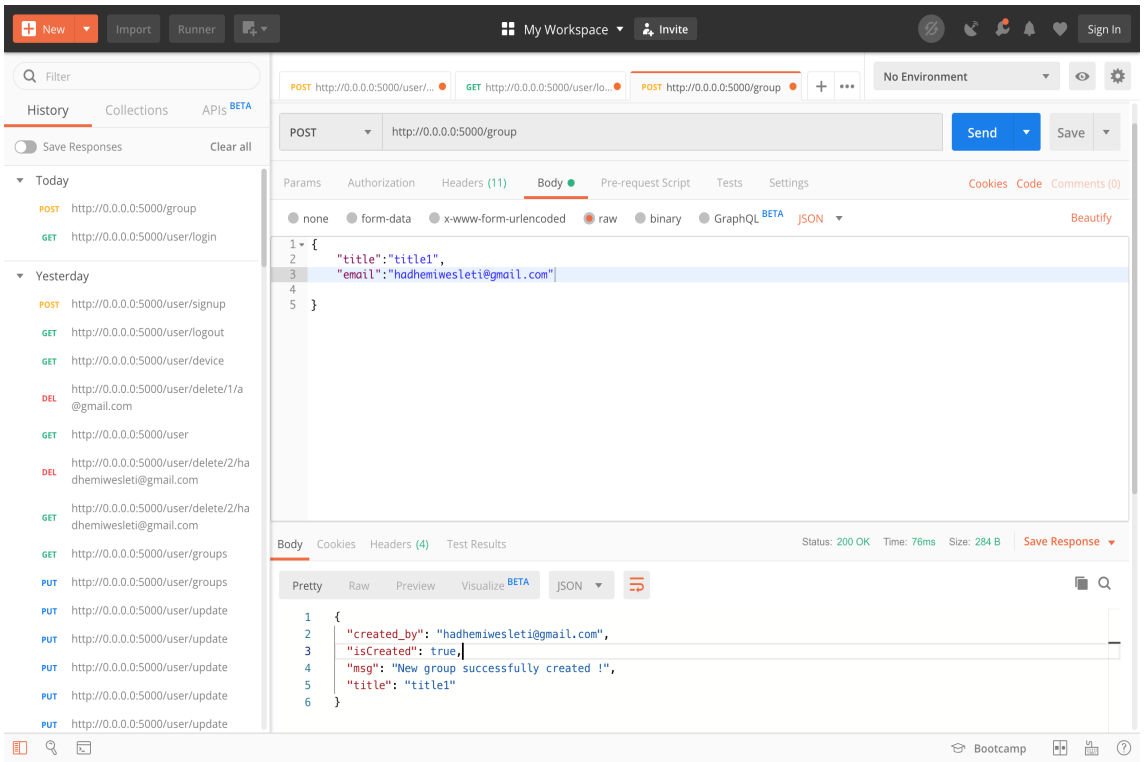


Figure 3.12 – Microservice management Group: Create Group

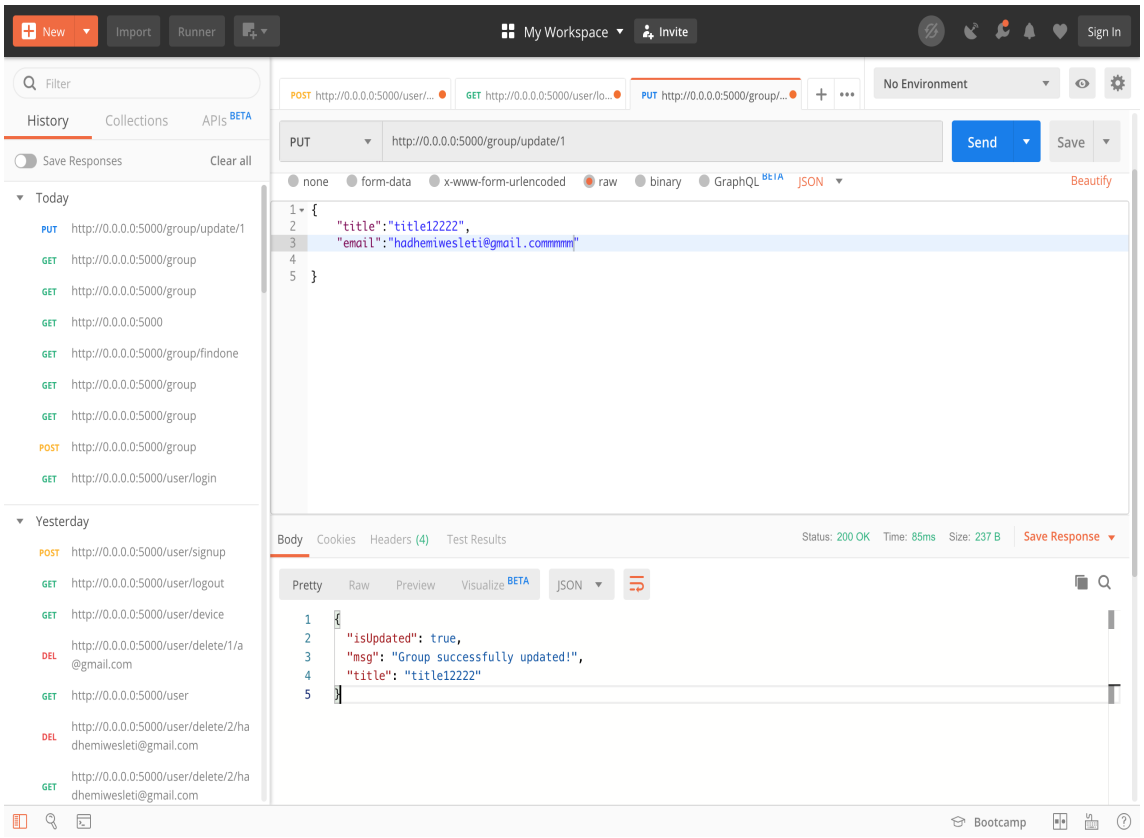


Figure 3.13 – Microservice management Group: Update group information

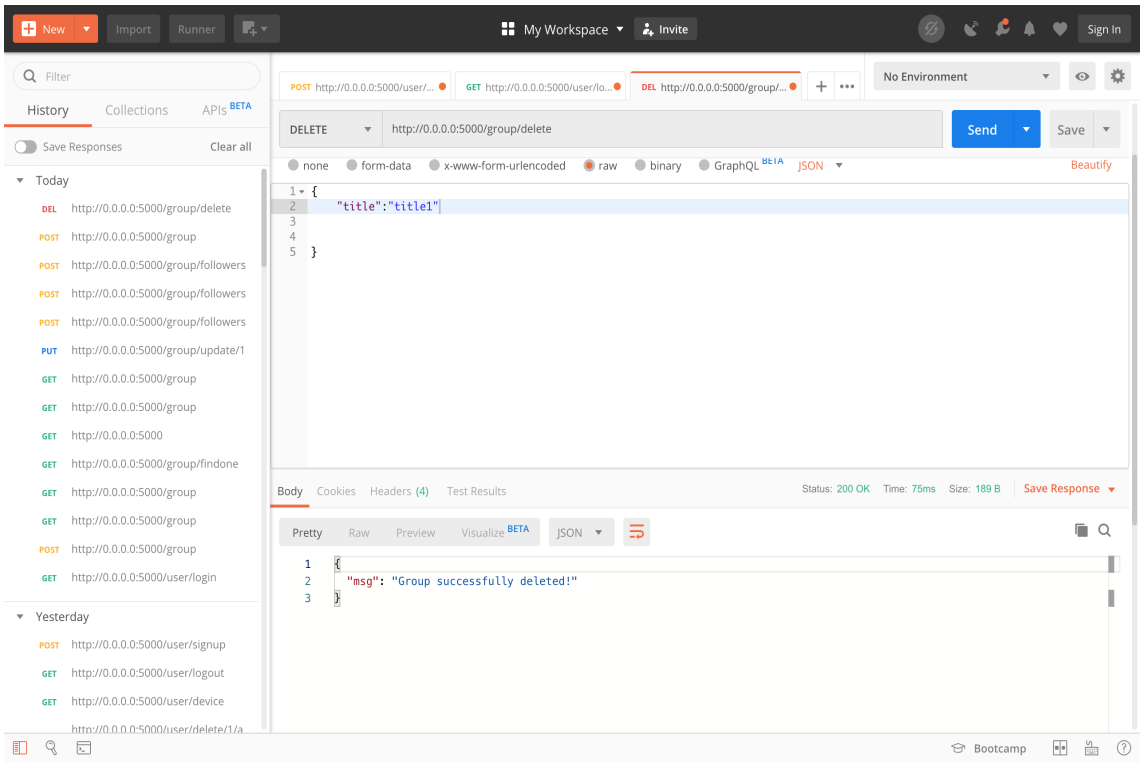


Figure 3.14 – Microservice management Group: Delete Group

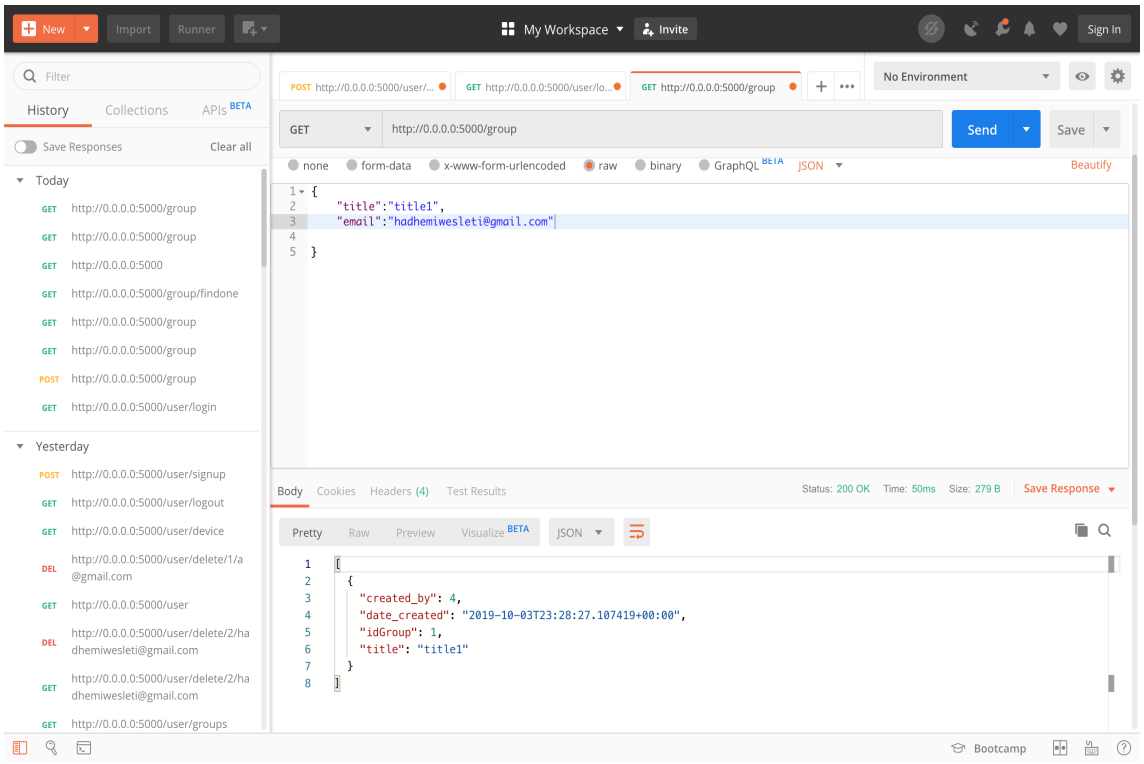


Figure 3.15 – Microservice management user: Get all groups

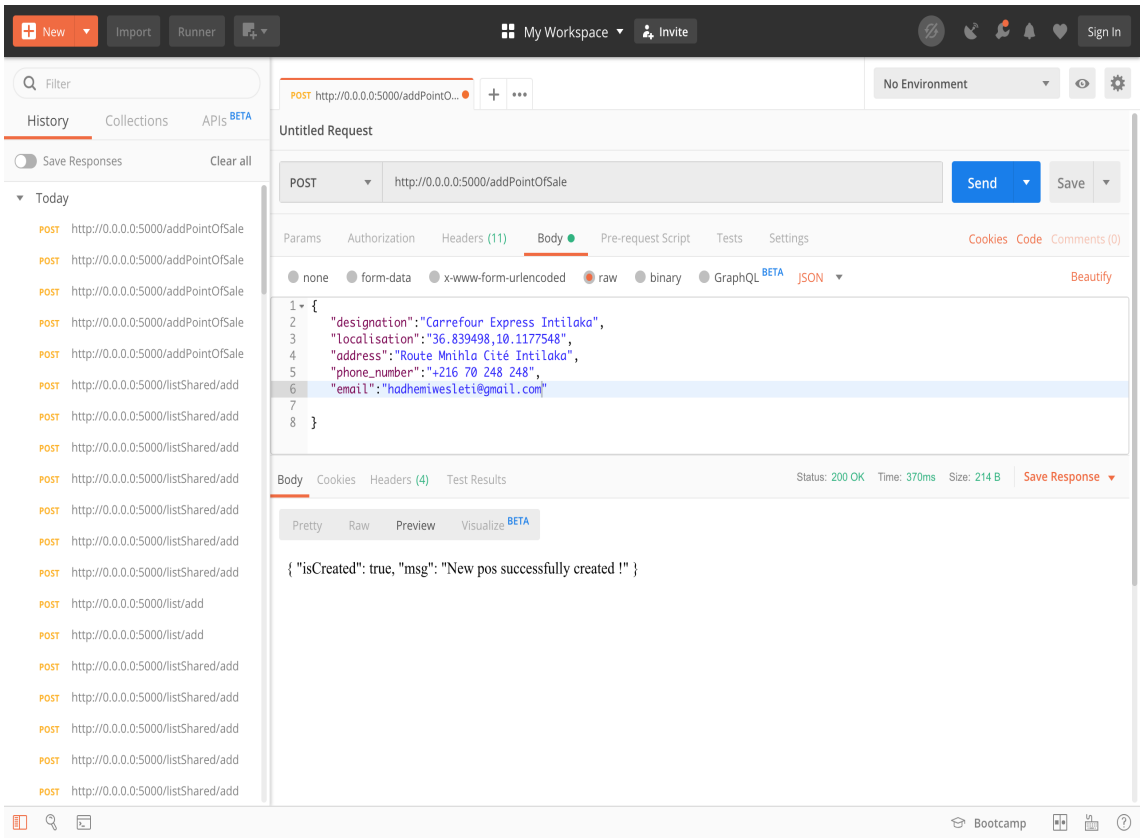


Figure 3.16 – Microservice management point of sale: Add a point of sale

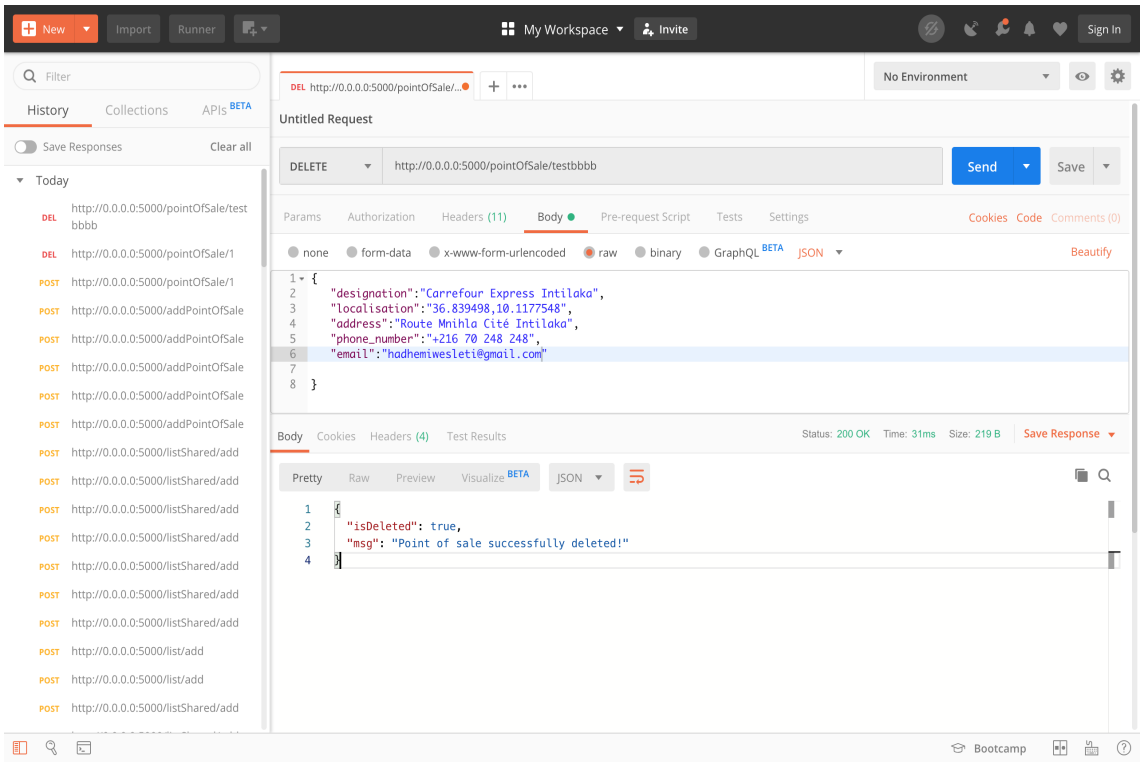


Figure 3.17 – Microservice management point of sale: Delete point of sale

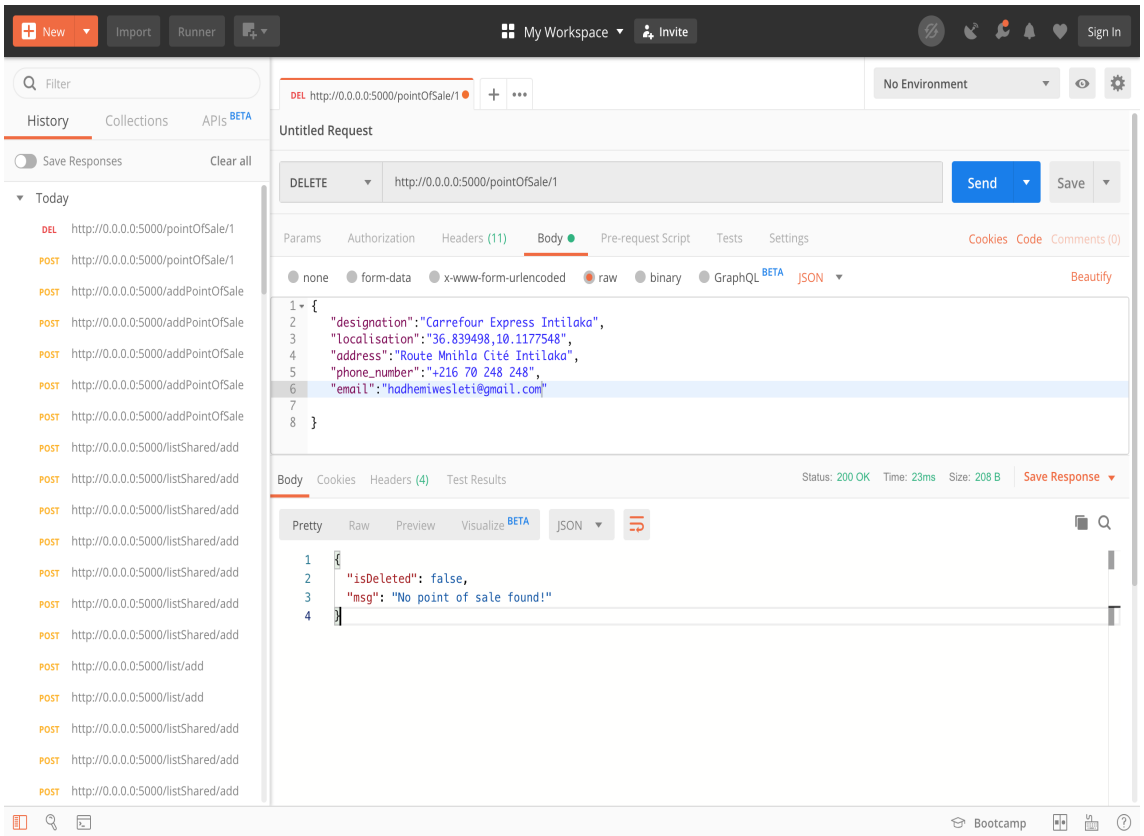


Figure 3.18 – Microservice management point of sale:Errors when delete a point of sale with wrong condition

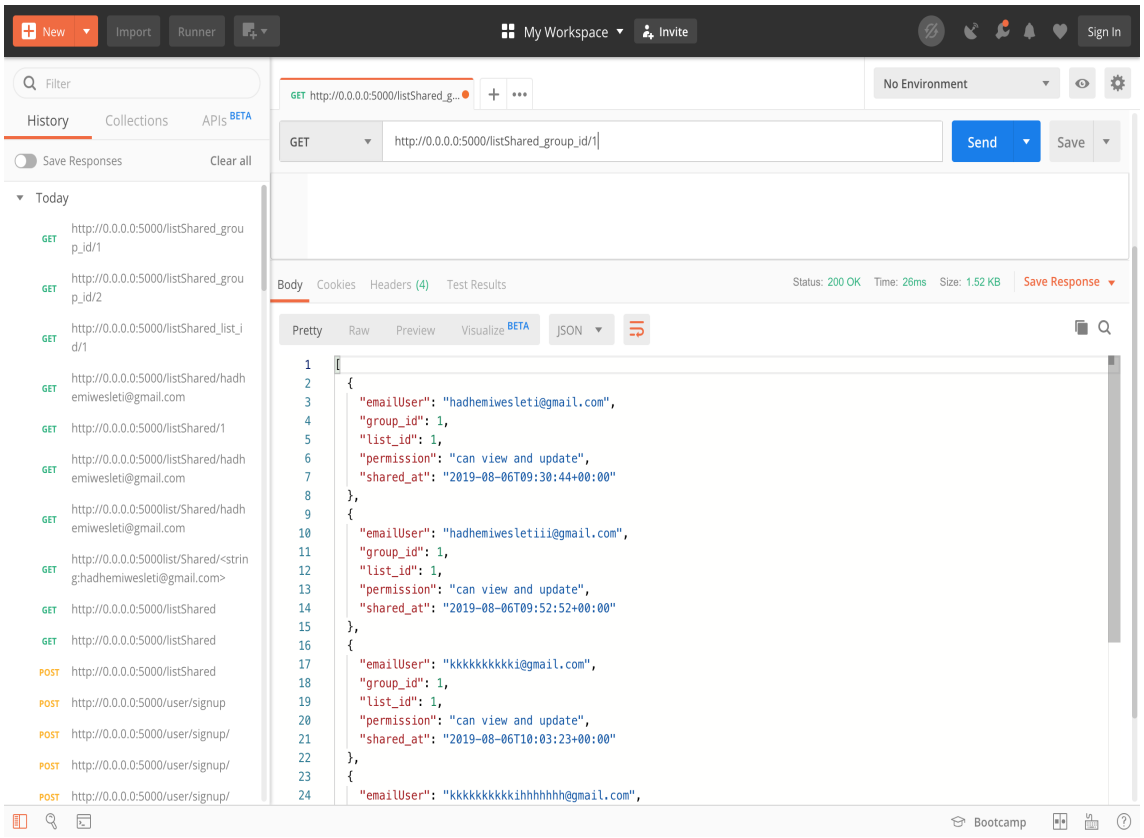


Figure 3.19 – Microservice management lis: Get all list shared by group id

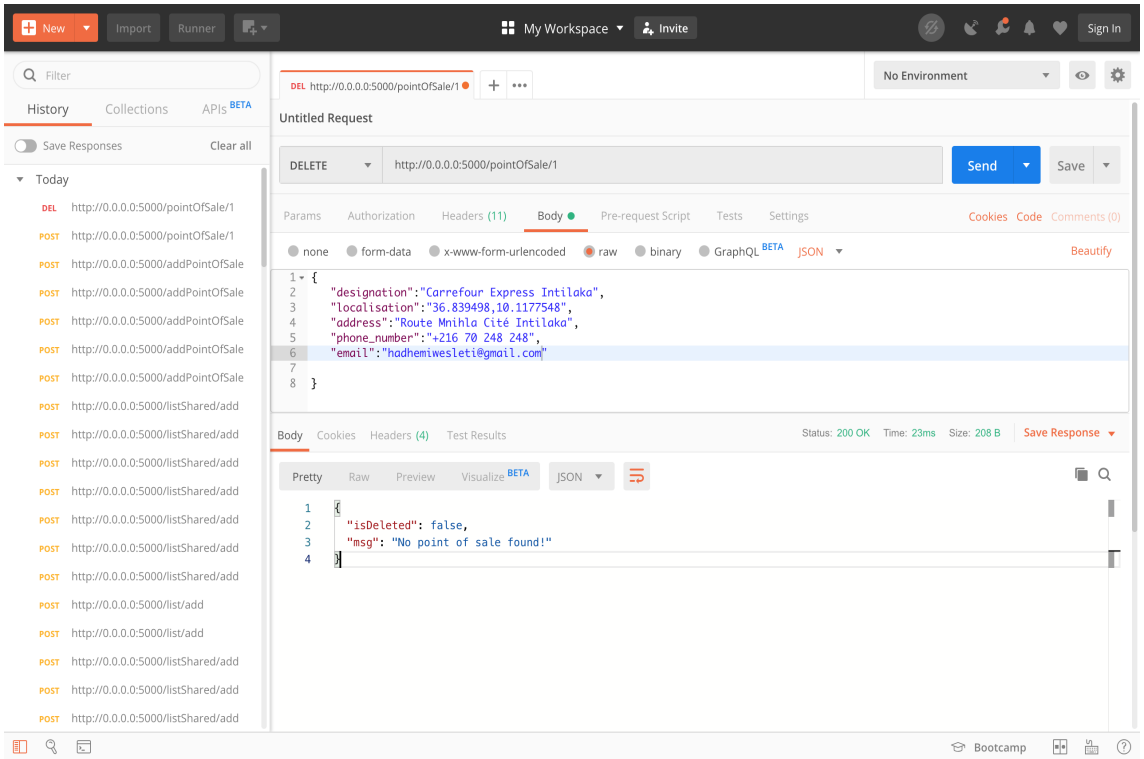


Figure 3.20 – Microservice management point of sale:Errors when delete a point of sale with wrong condition

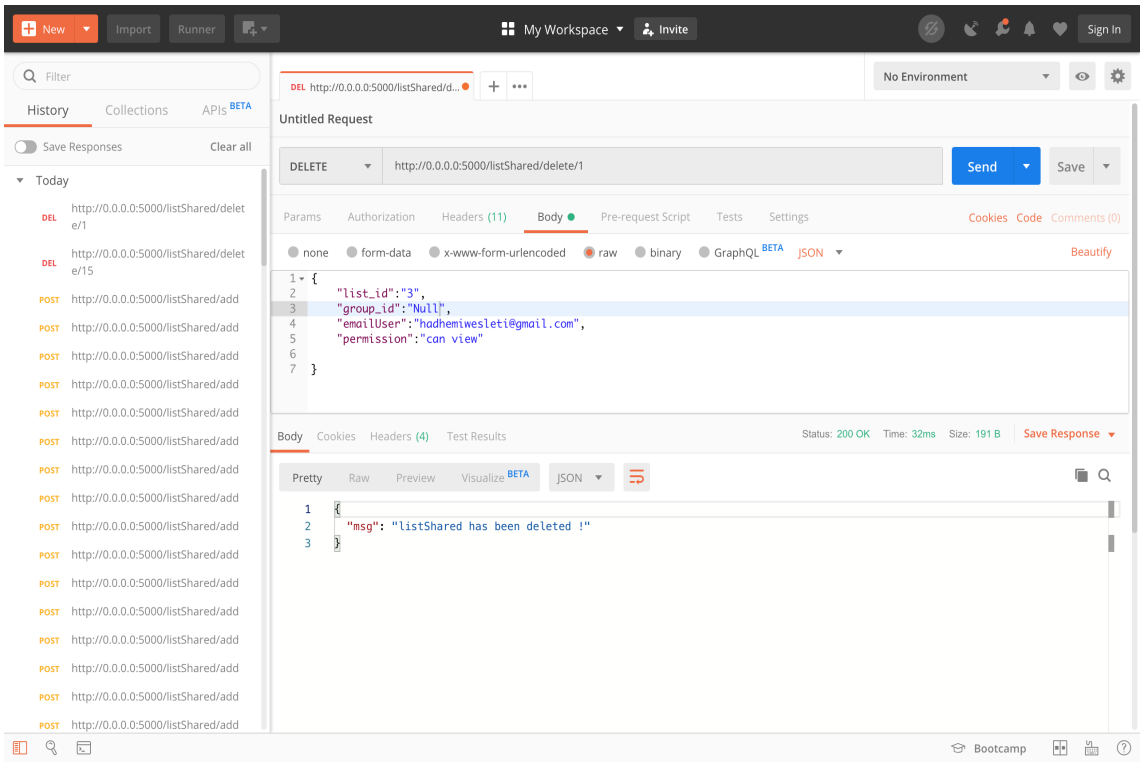


Figure 3.21 – Microservice management listShared : Delete list shared

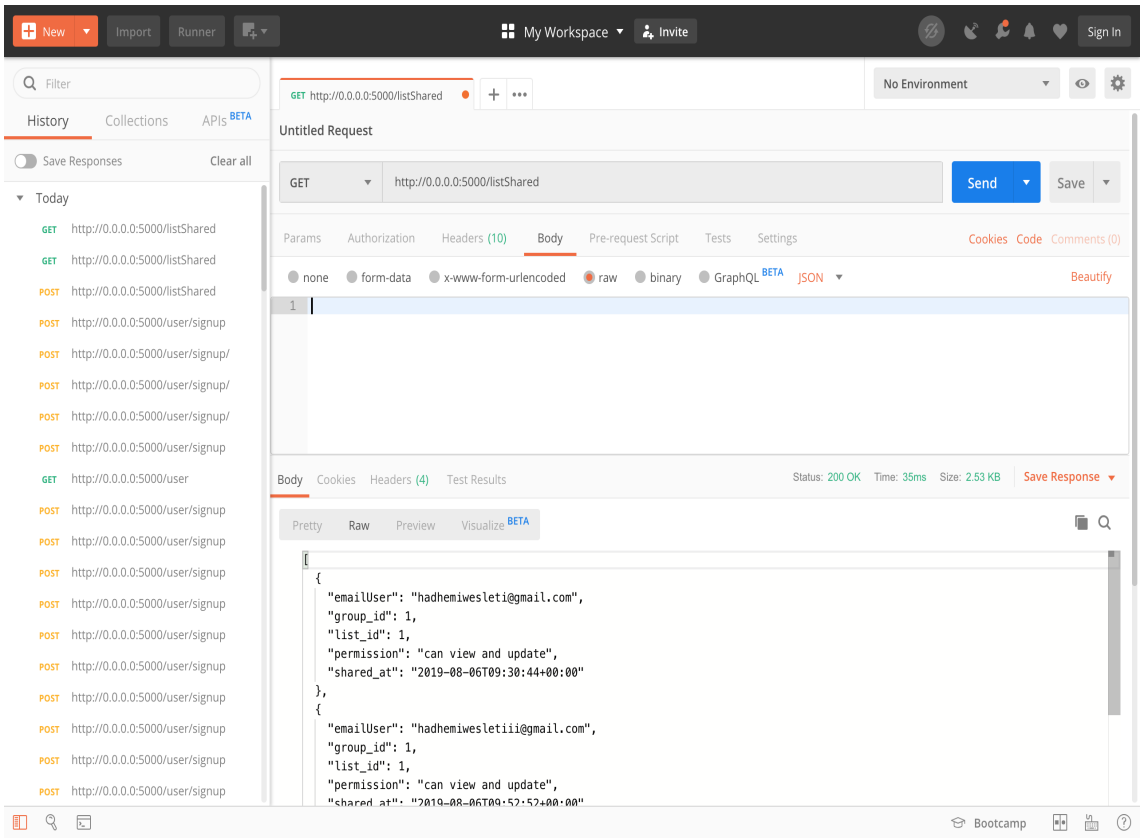


Figure 3.22 – Microservice management listShared : Get all list shared

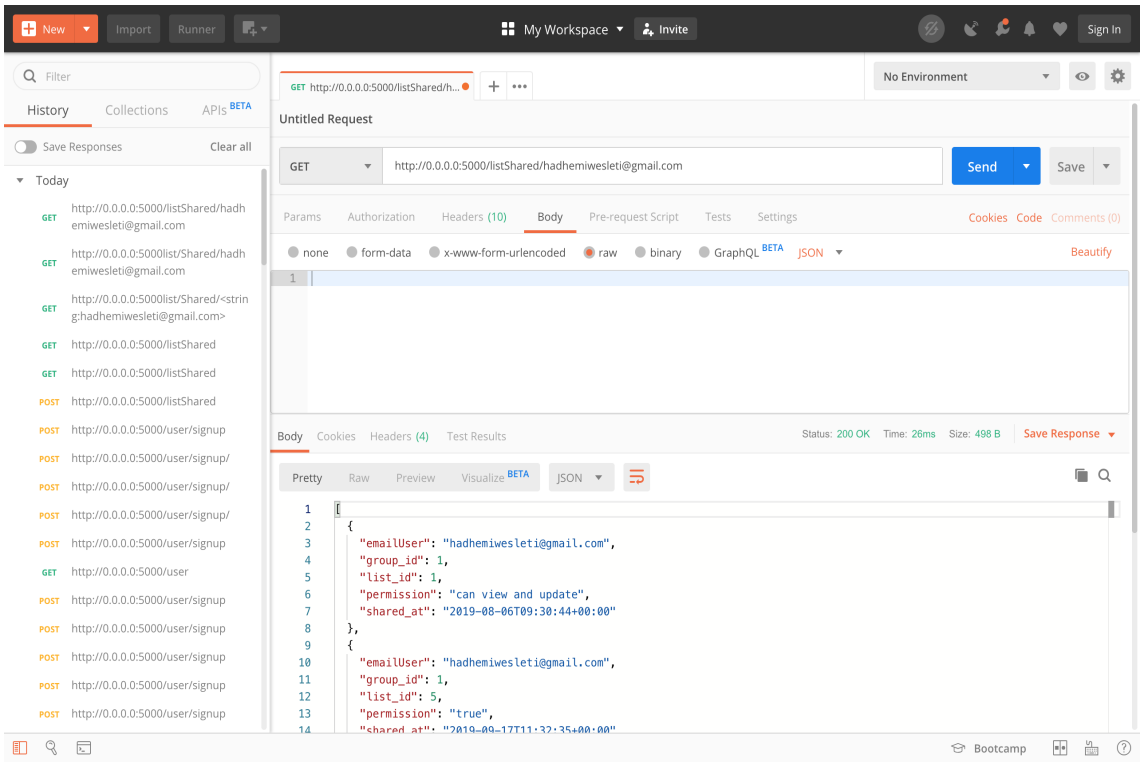


Figure 3.23 – Microservice management listShared : Get all list shared by email

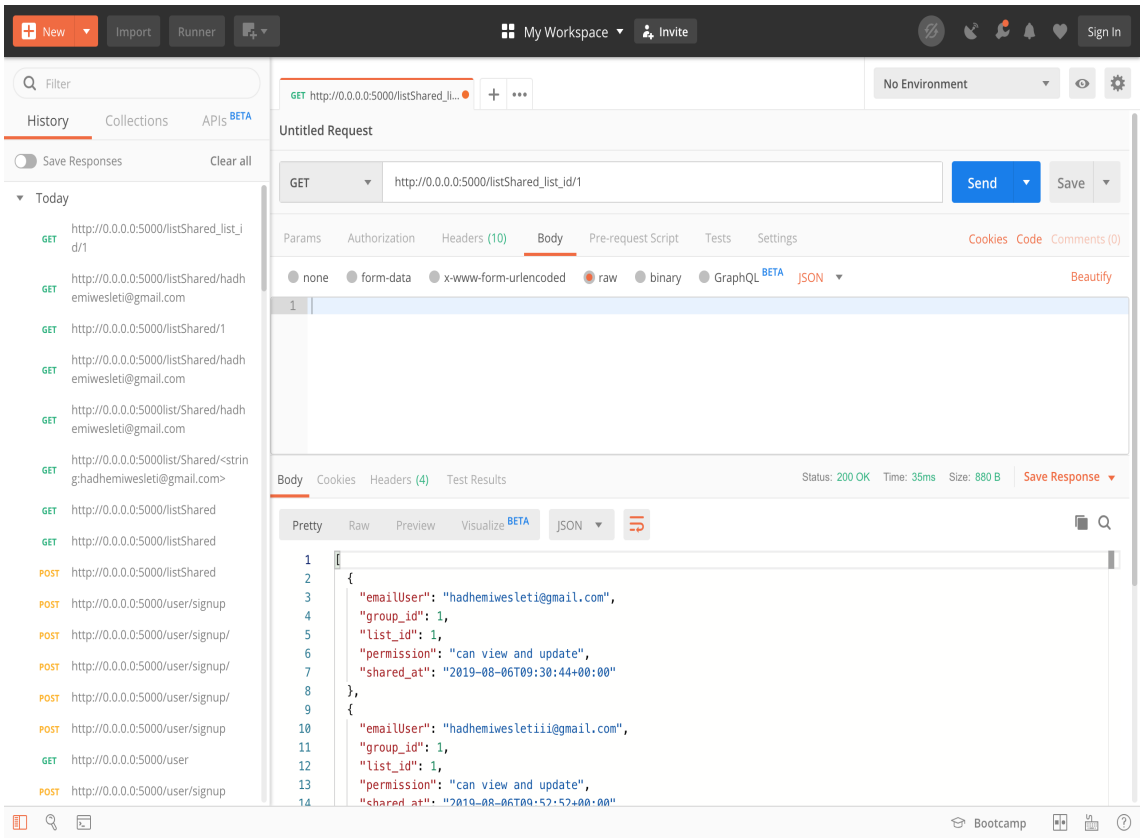


Figure 3.24 – Microservice management listShared : Get all list shared by id

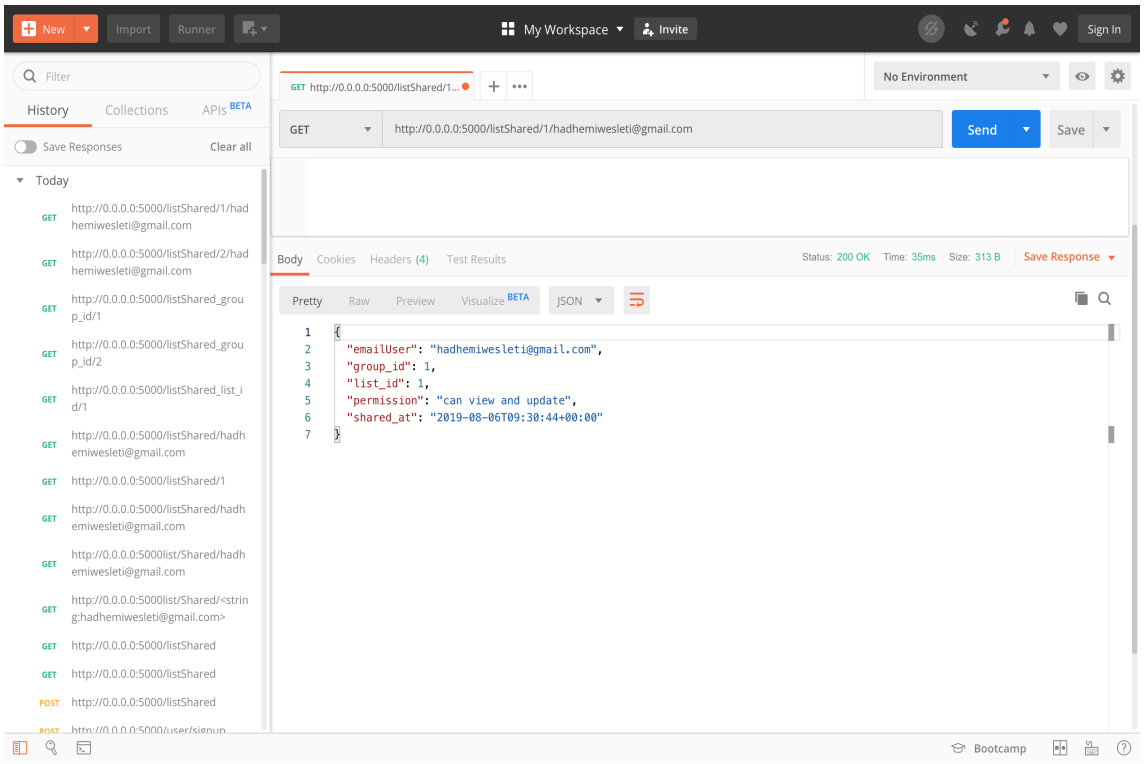


Figure 3.25 – Microservice management listShared : Get single list shared by email and id

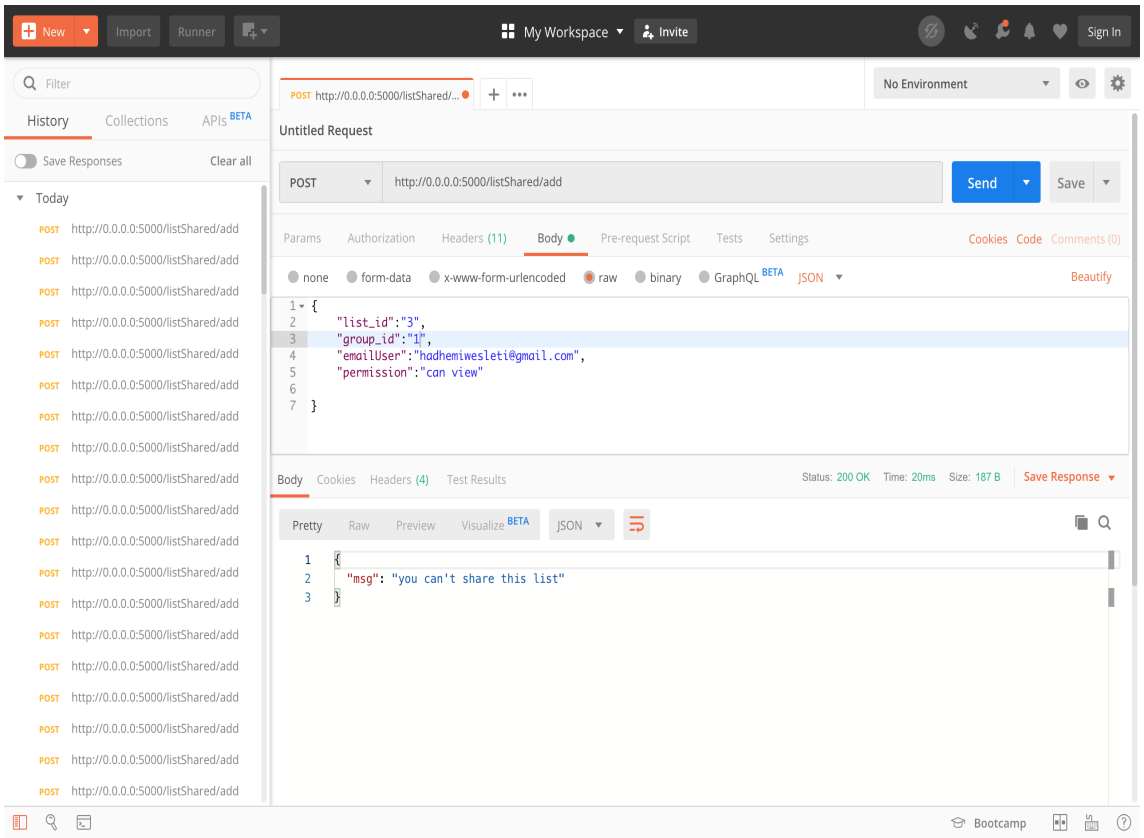


Figure 3.26 – Microservice management listShared : Errors when a user enter wrong information

3.4 Microservice communication

Any operation conducted on our system must keep a log in. This requirement is being translated into communication through REST xommunication betwwen the different microservices. However, remote calls may sometimes fail so in order to ensure a tolerance to this problem, we have opted for the solution of Kapache Kafka circuit braker, a dashboard is offered to track the execution of ongoing microservices. In case of failure of all instances of microservices, there is a michanism for fallback method that offers default result. the breaker circuit made with kafka wraps this method and monitors failures.

Conclusion

In the end of this part we have presented the work environment, unittests, some functionnal smart list appication and finaaly how to communicate our microservices.

Conclusion

Companies like Amazon, Netflix, Linkedin, and Pandora leverage recommender systems to help users discover new and relevant items (products, videos, jobs, music), creating a delightful user experience while driving incremental revenue.
Here we provide a practical overview of recommender systems named Smart List. First, its major reviewed by three major systems: content-based, collaborative filtering, and hybrid, followed by discussions on cold start, scalability, interpretability, and exploitation or exploration.

In this report we presented an explanation of different process that have led to implement a recommendation system which help users to manage their shopping list by promoting the social aspect and offer assistance to cutomers. We have identified both functional and non-functional requirements, described several applications in the field of recommender systems and organized our product backlog. Based on microservices architecture we have established our conceptual design approach. Then we have explored and tested some of the core microservices. We have release the implementation of the administration module which supervise and monitor all operations in real-time on a single platform.

Due to the lack of time we only worked on the client, list and point of sale microservice. In the next release we will focus on implementation of product and shopping microservices where we will manipulate Big Data and Machine Learning.