الجمهورية الشعبية الديموقراطية الجزائرية
République Algérienne Démocratique et Populaire
وزارة العليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

École supérieur en informatique
Sidi Bel Abbés 8 mai 1945

L'entreprise de développement
Valley Solutions

Mémoire de fin d'étude

En vue de l'obtention du diplôme : **Ingénieur**

Filière : **Informatique**

Spécialité : **Système d'information et web (SIW)**

Thème

DYNAMIC PRICING IN DOD APPLICATION

Auteur :
MOKHFI OMAR

Encadreur :
Dr. Mohammed KAZI TANI
M. Ilyas HANAGRIA

15 septembre 2020

Acknowledgement

In recognition, I would like to express my sincere thanks to everyone who contributed, from near and far, the smooth running of my end-of-study internship and the development of this modest work. My sincere gratitude to my thesis supervisor **Mr Mohammed Yassine Kazi Tani** for the quality of his teaching, his advice and indisputable interest he took in this work, I also thank him for his attention and patience.

I would like to thank my internship tutor **Mr Hanagria Ilyes** as well the entire Ouigo team at Valley Solutions, Algeria for their patience, advice, guidance and interest they have shown in my achievements. I would also like to thank "ALL" the gentlemen and ladies, my teachers who taught me during five years of training in IT for their valuable advice and guidance. My thanks also go to the members of the jury for accepting to evaluate my work. Without forgetting to thank my friends and colleagues (at ESI-SBA or in the Virtual world "Internet") which, all in different ways, contributed to what I may lead to the realization of this thesis. Finally, a warm thanks to my family (my parents and sisters, our brothers) for the support and encouragement they have brought me throughout my work.

Abstract.

Driver On Demand (DOD) platforms are a new option for passengers that took place in Algeria around 2017, making it a new field to explore. Despite the fact that they encounter a large number of users, Some features, that really make a difference from one platform to another, are still missing. One of the characteristics is the pricing strategy, and increasing availability in non crowded zones. The objective of this graduation project is to implement a full pricing system, which can improve the marketing strategies, and gain clients satisfaction, and as a result enhance the platform's income. For that purpose, we have proposed and implemented some systems. The first is the surge pricing system that calculates a multiplication factor based on drivers availability, and passengers requests in a certain zone. The second one is the prediction system that predicts duration of the ride which will be the main factor in price strategy. In addition to those systems, all configuration is controlled by the marketing team through a control panel.

We aim to achieve the following objectives during our internship :

- Implement an interactive system between marketing and pricing.
- Propose a surge pricing architecture.
- Propose and implement a duration prediction system that replaces Google maps.
- Set up a ready to deploy images of the proposed solutions.

Resume.

Les plateformes Voiture de transport avec chauffeur (VTC) sont une nouvelle option pour les passagers qui a eu lieu en Algérie vers 2017, ce qui en fait un nouveau domaine à explorer. Malgré le fait qu'ils rencontrent un grand nombre d'utilisateurs, certaines fonctionnalités, qui font vraiment la différence d'une plateforme à l'autre, sont toujours manquante. L'une des caractéristiques est la stratégie de tarification et l'augmentation de la disponibilité dans les zones non bondées. L'objectif de ce projet de fin d'études est de mettre en œuvre un système de tarification complet, qui peut améliorer les stratégies de marketing et obtenir la satisfaction des clients, et par conséquent augmenter les revenus de la plate-forme. Pour cela, nous avons proposé et implémenté quelque systèmes. Le premier est le système de tarification des surtensions qui calcule une multiplication en fonction de la disponibilité des conducteurs et les demandes des passagers dans une certaine zone. Le second est le système de prédiction qui prédit la durée du trajet qui sera le principal facteur dans la stratégie de prix. En plus de ces systèmes, toute la configuration est contrôlée par l'équipe marketing à travers un panneau de contrôle.

Nous visons à atteindre les objectifs suivants lors de notre stage:

- Mettre en place un système interactif entre marketing et tarification
- Proposer une architecture de tarification des surtensions
- Proposer et mettre en œuvre un système de prédiction de durée qui remplace Google maps
- Mettre en place des images prêtes à déployer des solutions proposées

Contents

List of Figures

List of Tables

Acronyms

DOD: Driver On Demand

DZD: Algerian Dinar

Chapter 1

General Introduction

Sommaire

1.1 Introduction

Dynamic pricing is a system that helps seller, or service provider to change price over time based on some factors set up by the seller himself such as time, distance, basic price, service request and provider availability, etc. While offering the client an exact prediction of the price [1].

Predicting a price with such factors that makes it change between a second and another was impossible at some time, and even after it became possible, predicting with such precision was quite a hard task. When it comes to market each error in prediction is money, and while the error is big and as a consequence money lost by the seller or the client is big, such system should be avoided.

Dynamic pricing is found in every large market nowadays, every take it or leave it market where clients numbers are so high and demand is so high that the only care is making money, in taxi fares, online stores, hotels, etc. An example of that is amazon store, or any similar store, where you can see clearly that each product's price changes continuously with variable discounts between clients, countries and periods, and in such a business there are always clients looking for products so the only care is variations of prices based on what satisfies clients and what brings more money to the business itself.

1.1.1 Context

Valley solutions for its new startup "Ouigo" trying to conquer the Driver On Demand (DOD) market in Algeria based on many strategies, and take the lead in concurrence with many existing services like "Yassir", "Heetch" and "Careem", etc. One of the strategies is the implementation of a dynamic pricing system that attracts drivers by providing a better income and wins over clients by providing a better service and reasonable prices while keeping the business standing.

For the system to work properly, it must be monitored by marketing team when it comes to their strategies and pricing of concurrence.

1.1.2 Problematic

For the dynamic pricing system itself, it was first implemented in the Driver On Demand (DOD) market by the first service provider "Uber" and it consists of many sub-systems, like for example surge pricing which is changing prices based on drivers availability and passengers requests in an area. Due to lack of control in the system, the fare price reached up to 50 times the normal price at disasters surge times and

it also reaches easily 5-8 times the normal price in normal surge times.

The pricing system requires months of data for the prediction part which is a problem for a new startup, and also requires the use of maps service in its inputs which is a critical cost for the company.

The precision of prediction is very important when it comes to DOD, for example, if there is an error of 20% and the fare price is about 500dzd, the result would be that either the driver loses 50dzd from its daily income, or client pays extra 50dzd for a ride. On the other hand, while there are cloud services specialized for this, the cost of the service is still critical for the startup.

1.1.3 Goal

The goal of this work is the implementation of a dynamic pricing system with a fare prediction subsystem while giving the marketing team full control. the production part of the system must be deployed to be used in the application. It is also responsible for building a working system that benefits both the startup and their clients, while lowering the flows that can be used by corruption, and also lowering the cost as much as possible by a well done bench-marking and studies of possible solutions.

The main goal is building a prediction model based on forecast algorithm.

1.2 VALLEY SOLUTIONS and NEEXIUM

1.2.1 NEEXIUM Digital

NEEXIUM[1], situated in Paris, France, is an expert company in digital engineering consulting. It assists its customers on a long term basis for the definition and implementation of digital solutions for small and medium sized companies, industries and organizations.

Figure (Fig. 3.5a) represents different stakeholders and structure of NEEXIUM Digital organization.

[1]https://www.neexium.com/

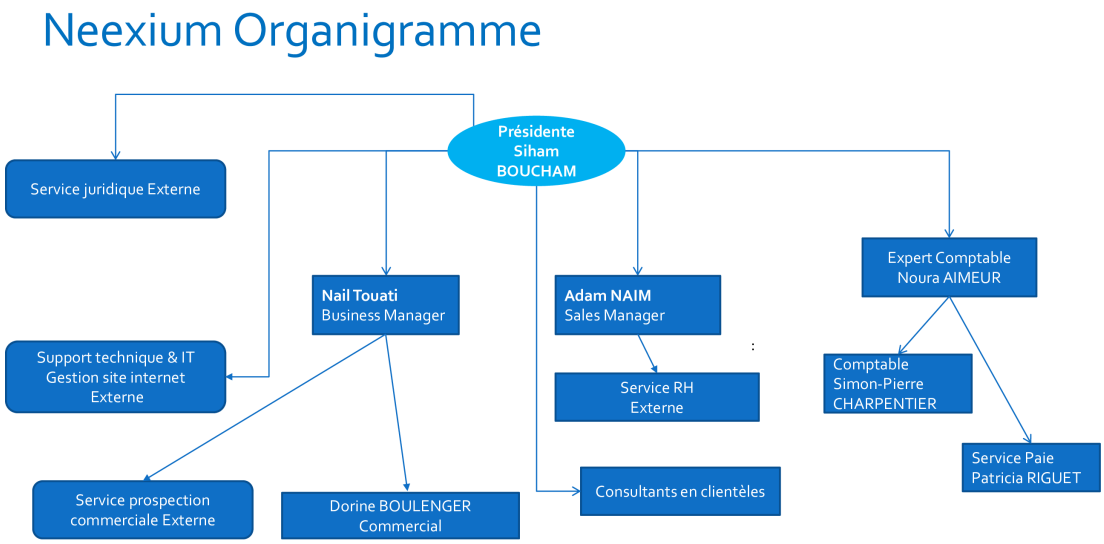


Figure 1.1: Organization chart of NEEXIUM Digital

1.2.2 VALLEY SOLUTIONS

VALLEY SOLUTIONS, situated in Algiers, Algeria, is a reference IT company, leader in the development of measurement and specificity of technological solutions. Figure (Fig. 3.5b) represents different stakeholders and structure of Valley Solutions organization.

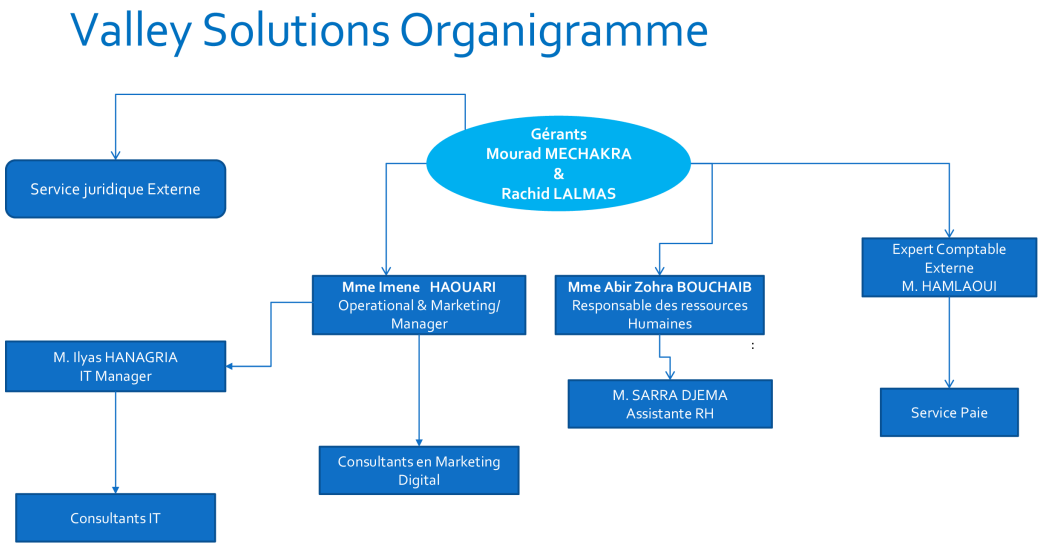


Figure 1.2: Organization chart of Valley Solutions

1.2.3 Presentation of Ouigo

OUIGO is a new upcoming startup in the DOD market developed by Valley Solutions. It is an innovative transportation service that we can use with our smartphone anywhere and anytime allowing everyone to reserve a driver and travel in complete safety.

Market situation in Algeria of Driver On Demand services

In Algeria, the concept of requesting a private driver using a smartphone and geo-localization started in 2017 and attracted a lot of citizens that lived the experience with the help of many service providers.

In 2018, this market attracted many investors and business-men that opened the doors for 10 more applications to exist.

Driver On Demand (DOD) services came with new features that transport service was lacking, and that includes the availability, providing the cost of fares, private drivers, choosing the desired pickup and destination, etc.

Existing Driver On Demand services in Algeria

Yassir, TemTem, coursa, Amir and Tymô or even Wassalni et Lahagni all seek being the Algerian "Uber" and dominate the market, and each one of these service providers offering their own features, and their own methods to distinct themselves from the others.

Having different clients, some satisfied and some not, the only winner in this is the driver having a variation of choices.

Application	Strengths	Weaknesses
YAssir	Large number of drivers Covers 12 wilayas Large amount of data	Application bugs Non competitive prices
TemTem	Referrals system	Highest prices in market
Coursa	Competitive prices Recommended by influencers	Available only in Algiers
Amir	Negative drivers feedback	Available only in Oran
Careem	Exists in many countries Daily offers	
Heetch	Exists in France and Algeria Competitive prices	

Table 1.1: Comparison between Driver On Demand services in Algeria

1.3 Organization of the report

This report consists of a total of 6 chapters. Next chapter (chapter 2) defines the basic concepts of dynamic pricing in DOD applications, and also different approaches of fare prediction: the machine learning approach based on forecast algorithms, and the amazon forecasts service.

Chapter 3 is a study on the existing solution, and the already developed features, including the android application and admin dashboard.

Chapter 4 consists of an analysis and conception of our system with different

diagrams (Use case, Sequence, Classes, Activity).

Chapter 5 introduces the implemented system with its architecture and UI/UX.

Chapter 6 presents the experimentation and the results acquired from the system and also the development environment to end it with a final chapter that consists of a conclusion and future perspectives.

Chapter 2

Dynamic pricing in Driver On Demand application

Sommaire

2.1 Introduction

Pricing has a huge importance in today's market, and a huge impact on the company itself due to the change in market flow, so clients cannot be won without offering the perfect cost of a service. The pricing strategy is a critical process because People's behaviors towards money is the same as their behaviors towards their health. As a result of them giving such importance to prices, offering a service to an audience is all based on their acceptance and satisfaction toward the cost of the service.

Due to the continues change of different factors in seconds of time, dynamic pricing became one of the most important strategies, but for Algerian users and all users around the world, they always ask for the price first before using the service or buying the product. Without a great dynamic pricing system, service providers were only able to provide a range of prices putting clients in range of confusion.

Supposing that we are providing a client with a service that has different factors that can affect its price, for example, one of the factors is uncontrollable time that varies from 10 minutes to 50 minutes, in this situation we can tell the client that the price varies from 100dzd to 500dzd because he pays 10dzd for a minute. At the end of the service, the time was 70 minutes, which has 2 results, either the client pays the unpredictable extra 200dzd and he will never use the service again, or the service provider pays it and the income of the company gets low. A good dynamic pricing system will put neither the client nor the service provider in this situation and the offer will be 700dzd from the start, take it or leave it.

For a Driver On Demand (DOD) service the challenge is even higher, because there are 3 parts in the deal, the service provider, the driver, and the passenger. Both the driver and the passenger are opposites, so if the cost is a good income for the driver, that cost will be paid by the passenger and in this scenario the service provider wins one client and loses the other, in another scenario driver has a bad income and the passenger is satisfied for paying less for the ride, and here, the service provider loses driver and wins the passenger. A middle satisfaction scenario where both clients are not very satisfied, but satisfied, that's **the dynamic pricing system**.

2.2 Basic concepts

The terms surge, discount, prediction and Competition monitoring are the basic concepts of this work, that's why we will try to present some definitions to make moving through this thesis more understandable and to simplify the comprehension of dynamic pricing.

2.2.1 Surge

A surge is defined in the Collins dictionary [1] as an increase or a sudden development in a factor like feeling or distance that was fixed on certain value, or was improving slowly at a constant small rate.

It is defined in the Oxford dictionary [2] as a sudden powerful forward or upward movement caused by a crowd or some natural force.

In certain events, numbers increase more than common rate. For example, in a zone that used to hold 1000 people, if a cultural event happens in that area number of people can reach a million.

2.2.2 Coupon

A coupon is defined as a small piece of printed paper that allows you to get a product or a service for free, or to get a discount to pay less than the usual price [3].

When we talk about coupons in the digital world, it's more of a code than a piece of paper, and it's a strategy applied by a service provider to get new clients, or to enhance the use of the service at certain periods.

2.2.3 Prediction

A prediction is defined [4] as a phrase stated by someone saying what he thinks will happen in the future.

[1] https://www.collinsdictionary.com/dictionary/english/surge
[2] https://www.lexico.com/definition/surge
[3] https://www.collinsdictionary.com/dictionary/english/coupon
[4] https://dictionary.cambridge.org/dictionary/learner-english/prediction

In our field, prediction is more of a forecast, which is deciding on a future value of a factor based on old data that have old values of that factor and that certain time.

2.2.4 Competition Monitoring

Competition monitoring is following the data and strategies of service providers in the market that offer a similar service as you.

In a Driver On Demand (DOD) service, for example, it's done by the marketing team, and it's basically noting pricing strategy, new features, etc. of other companies and these data are used to decide on the service's pricing and features.

2.3 Dynamic pricing

Dynamic pricing consists of 4 main systems, Interactive system handled by marketing and based on competitors monitoring, Coupons system for handling discounts, Surge pricing based on drivers availability and passengers requests, and prediction system that defines price using forecasts.

2.3.1 Interactive system

Interactive system is the non automatic part of dynamic pricing, it allows the marketing team to decide on a basic configuration that decides the price of a ride. The basic configuration includes the basic price which is the minimum fare, price per distance, and price for long distances.

For a better choice of this basic configuration the system uses a comparing subsystem that compares the price for 2 points A and B with prices of competitors in the market.

2.3.2 Coupons system

One successful marketing strategy to win over clients is offering discounts, especially on special occasions. Coupons system takes into consideration the discount strategy,

either it is for individual users, for a group of users and also the reason of this discount, either it's a special occasion, or for using the service daily.

2.3.3 Surge Pricing system

In Driver On Demand (DOD), drivers are free to work in the area they want, and pick the zone they want to drive in, which leads to the variation in the capacity of each zone. For the goal of attracting more drivers to work in a certain zone that has a high number of requests and low number of drivers, surge pricing takes lead by augmenting the fare cost in the area which will be a goal for all drivers. The system works by setting a surge multiplier for that area, the only inconvenience is that it must be controlled so that the multiplication factor doesn't go overboard [3].

2.3.4 Prediction system

The main system in our work is the prediction system that allows giving an exact estimation of the price of the ride which affects the client's decision and make it quicker. The factor that decides the price is the time of the ride that is affected by many other factors like traffic, weather, driver speed, and also the path taken by the driver. Since other factors like the basic price, price per kilometer, etc. can change over time, our system predicts the time and not the price.

Time of the ride can be estimated using Google Maps API, but the price of the service which is 8$ per 1000 request makes it an avoidable solution although it gives a good estimation.

2.4 Conclusion

In this chapter, we have presented the basic concepts of pricing and the main subsystems of a dynamic pricing system. These basics will give us a good understanding of the next steps in implementing our solution.

Chapter 3

Analysis of the current situation

Sommaire

3.1 Introduction

One of the challenges of our internship was working on an existing project. Valley solutions decided at first to go for a freelance project, and although the freelancers ended by building a functional system that has basic functionalities of a DOD application, it was lacking some functionalities and contained some problems that will be

discussed in this chapter. Figure (Fig. 3.1) shows the final deployed Ouigo project of the IT team.

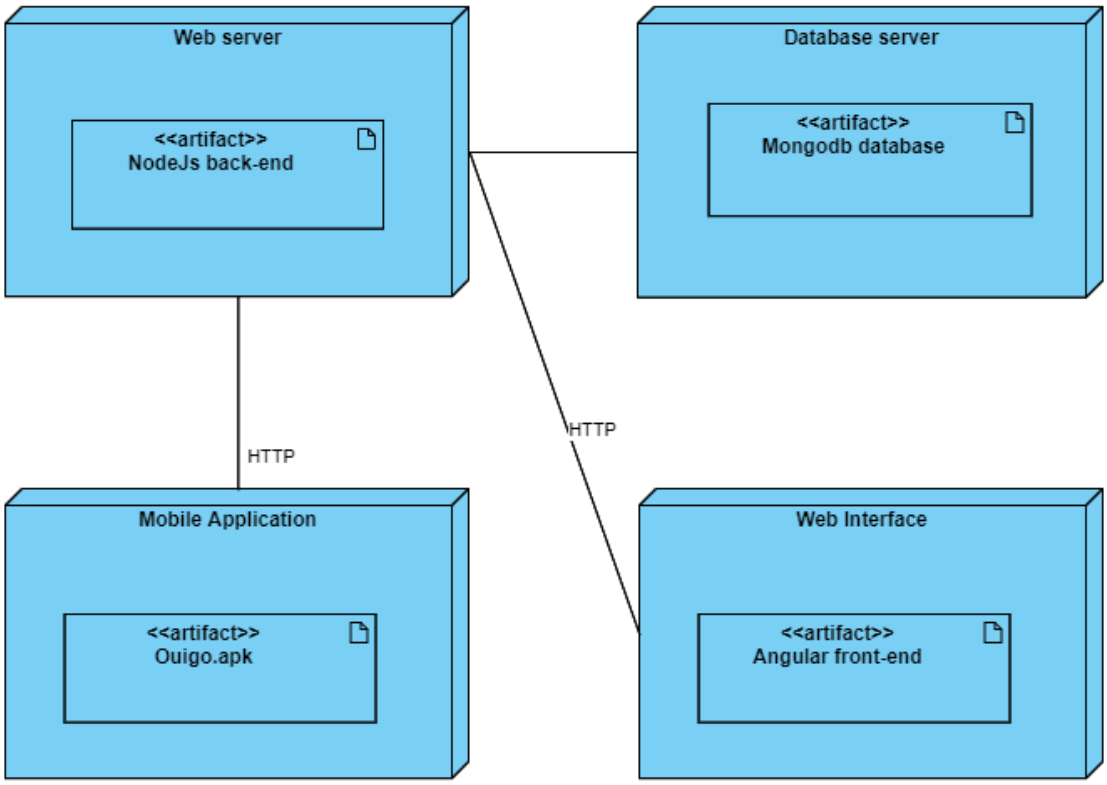


Figure 3.1: Deployment diagram of the existing Ouigo system

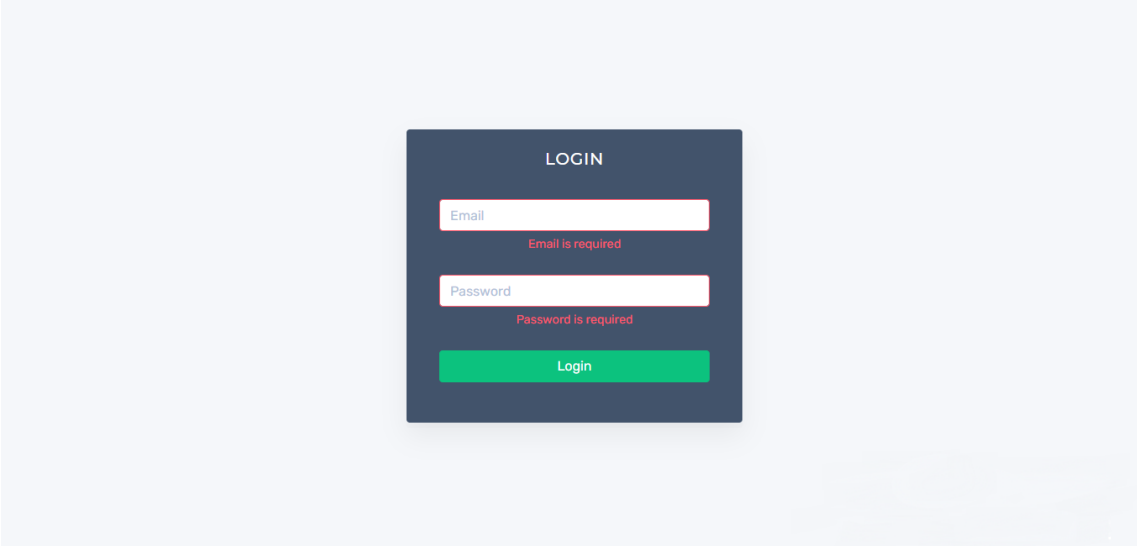
3.2 Existing project

Ouigo had the very basic functionalities developed with MEAN[1] Stack, and it's consisted of 2 applications, a web application for the admin to control the system, and an android application for both riders and drivers.

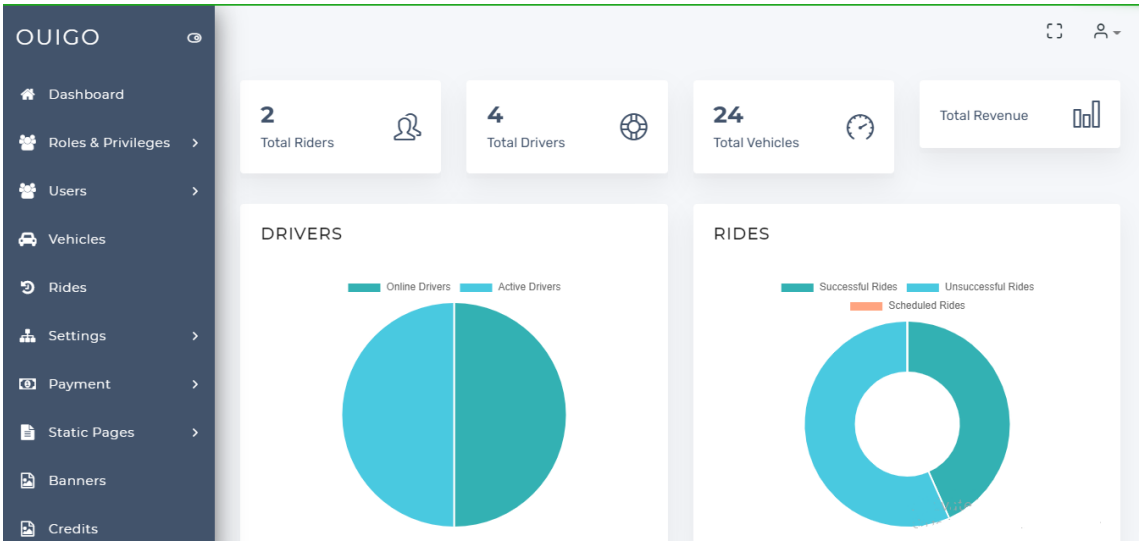
3.2.1 Web application

Figure (Fig. 3.2) shows the web interfaces of both login page and the dashboard that has many pages, each page with its functionalities.

[1]MEAN : MongoDb - Express - Angular - NodeJs



(a) Login Page



(b) Dashboard Page

Figure 3.2: Web interfaces of the existing Ouigo application

Use case diagram

Figure (Fig. 3.3) shows the functionalities of the web application of Ouigo. Many functionalities were grouped together in some use cases to simplify the diagram but will be detailed in this part.

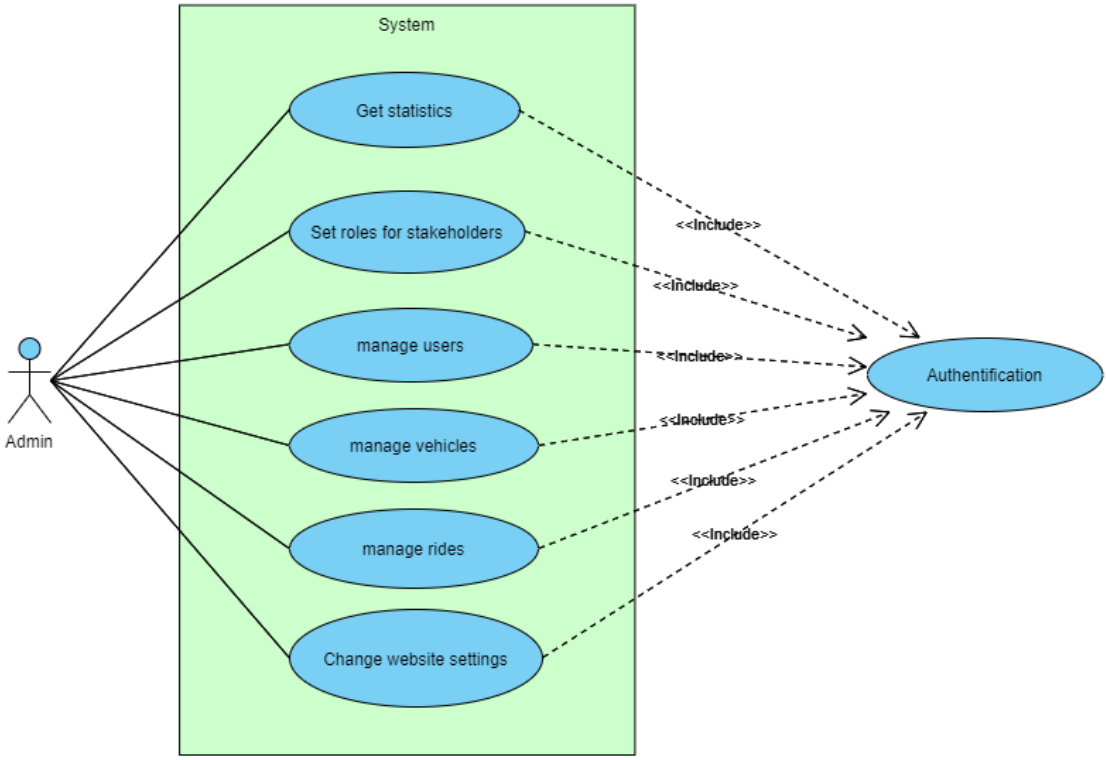


Figure 3.3: Use case diagram of the existing Ouigo web application

* **Get statistics:** The admin can get information about number of rides, drivers and vehicles, he can also see visualization of online/offline drivers and rides with different states.

* **Set roles for stakeholders:** Each user who is also a stakeholder has access to certain and precise functionalities of the system, the admin here can set roles for users and also give specific privileges to each role.

* **Manage users:** The admin can add and manage riders, and drivers by controlling their active state and their juridic information.

* **Manage vehicles:** Where each driver sign up with his own vehicle, the admin can manage and keep on track those vehicles for juridic purposes.

* **Manage rides:** This functionality gives the admin the ability to get all completed and uncompleted rides where each ride has its fare and tips.

* **Change website settings:** Where the admin controls all static pages with their details, like contact information, about us section, header, footer, etc.

MongoDB Database

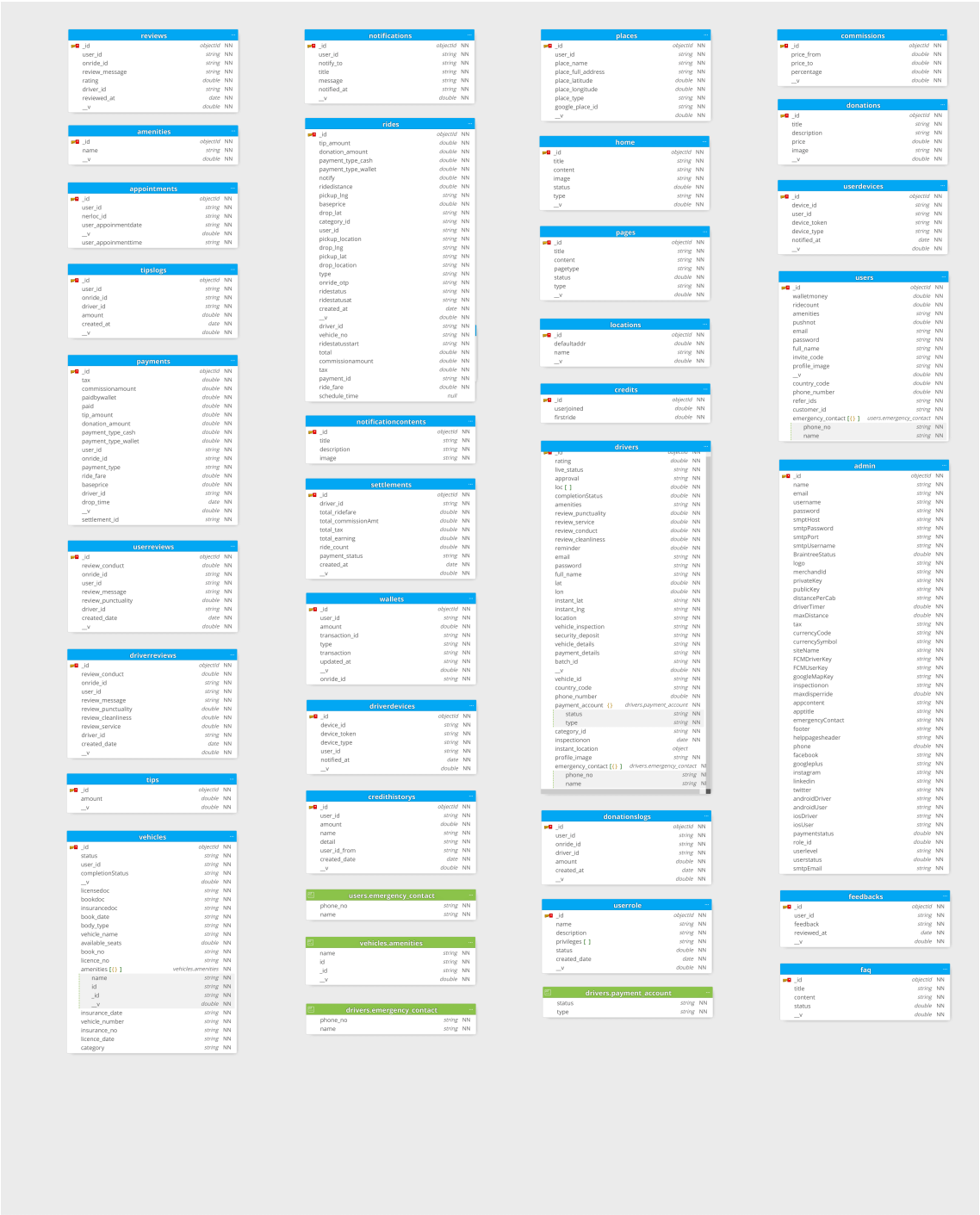
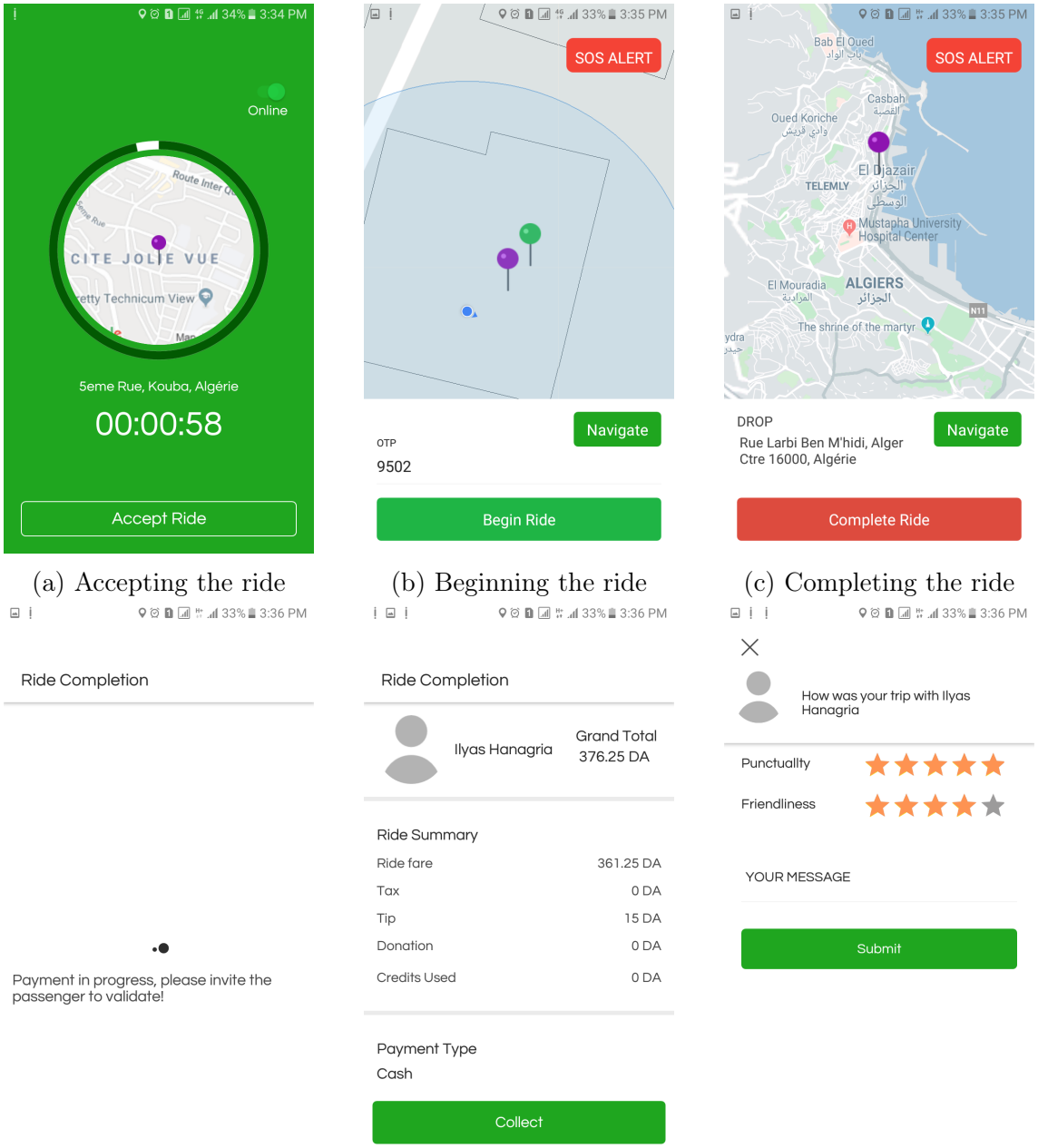


Figure 3.4: MongoDb documents of the existing Ouigo application

3.2.2 Mobile application



(a) Accepting the ride (b) Beginning the ride (c) Completing the ride

(d) Confirming the ride (e) Payment report (f) Reviewing the driver

Figure 3.5: Existing mobile application functionalities

For the mobile application that is only available for android, the basic functionalities are introduced in Figure (Fig. 3.5) and it shows the process of a ride for the driver application, and the passenger application.

When the passenger choose pickup and destination location a ride is requested then the driver is prompted with figure (Fig.3.5a) where he can accept or skip the

ride request. The driver then arrives at the location and picks up the passenger to begin the ride as seen in figure (Fig. 3.5b), the moment he arrives at the destination location, the driver needs to complete the ride and wait for passenger's validation as seen in figure (Fig. 3.5d), while this step is not necessary in terms of user experience, it is very necessary for user's security.

When the ride is completed and confirmed the driver collects the money as seen in figure (Fig. 3.5e) and the passenger is prompted with the final step (Fig. 3.5f) where he reviews the driver for punctuality and friendliness.

3.3 Problems with the existing project

3.3.1 Architecture

The monolithic architecture was used to build the full MEAN stack, and while this architecture is functional, it has many drawbacks, especially for the desired project.

The monolithic approach is simple and not preferable for big projects, because the bigger the project is, the more complex and the hardest is to make changes or to scale the application.Another drawback is the need to redeploy the whole application when making any change, and also a bug in a module can bring off all other modules.Without forgetting an important issue, which is the need of developers working with the same technologies.

3.3.2 Code structure

A challenge in the application was that the code is not very well structured, and also not documented, which means that adding functionalities would take more time and requires understanding the conception before executing.

3.3.3 Maps service

Since the application is monolithic, Everything used by the android application was hard-coded including the map functionalities. Since the application uses Google Maps API, another issue was the need to secure the premium key.

3.4 Tasks that should be done

Before working on the current project, different issues must be resolved to start implementing the dynamic pricing solution, which leads to the following tasks:

Code refactoring

The first task to do is refactoring the web application code, for both front-end and back-end, to add dashboard functionalities for the marketing team.

Migrating to micro-services architecture

Working with the IT lead developer, the next task is dividing the application into sub-functionalities and isolate each functionality into a proper micro-service, in its own ready to deploy the image.

Implementing maps micro-service

Choosing to work with Google Maps API wasn't the final decision, this task is basically doing a benchmarking on different maps services and finishing with an implementation of a map micro-service that can be used by the android application, the dashboard and the pricing micro-service.

Implementing pricing micro-service

The last task is doing a study on the dynamic pricing to decide on the functionalities of the service since the title was a little bit noisy, and finish by implementing a micro-service ready for deployment that works with both the mobile application and the control dashboard.

3.5 Conclusion

In this chapter, we have presented a study on what already exist in the project by defining the drawbacks of the implemented solution and the tasks that should be implemented. As it was clear the existing application was not scalable in terms of functionalities and a basic understanding of it is a must before jumping to the conception part.

Chapter 4

Analysis and conception of the system

Sommaire

4.1 Introduction

In this chapter, we begin by identifying the needs that must be addressed and satisfied in the development phase. We use both the use case and sequence diagrams to express the desired functionality of the system, then we model the data of the system through a class diagram and the business aspect through an activity diagram.

4.2 Analysis of needs

The expression of needs is a phase that consists of understanding and determining the different functionalities and the needs of the system. Before starting the conception, we list the functional and non-functional (technical) specifications that will be used to establish a complete conception of the solution.

In order to structure the specifications and to properly manage development activities, an organization of the specifications in modules is necessary.

Our system consists of several micro-services, where each service has it's own functionalities. We can group these functionalities into 4 sub-systems, the basic pricing system, surge pricing system, discounts system and price prediction system.

4.2.1 Functionalities

To simplify the development of the functional specifications of the system, we have defined a group of sub-systems (pricing, surge, discounts and prediction). The following table describes the priorities assigned to the specifications.

System	Priority
Basic Pricing Configuration	P (Must have for production build)
Discounts System	W (want to have but can wait)
Surge Pricing	F (Future implementation when there is enough data)
Price Prediction	F (Future implementation when there is enough data)

Table 4.1: Priorities of the functional specifications

4.2.2 Use-case diagram

Use-case of the basic pricing configuration system

Figure (Fig. 4.7) presents a Use Case diagram for the basic pricing configuration. When the admin choose a pickup and destination location, he is able to estimate the price of Ouigo ride, and also get prices of competitors in the market, so he can then configure the appropriate pricing settings.

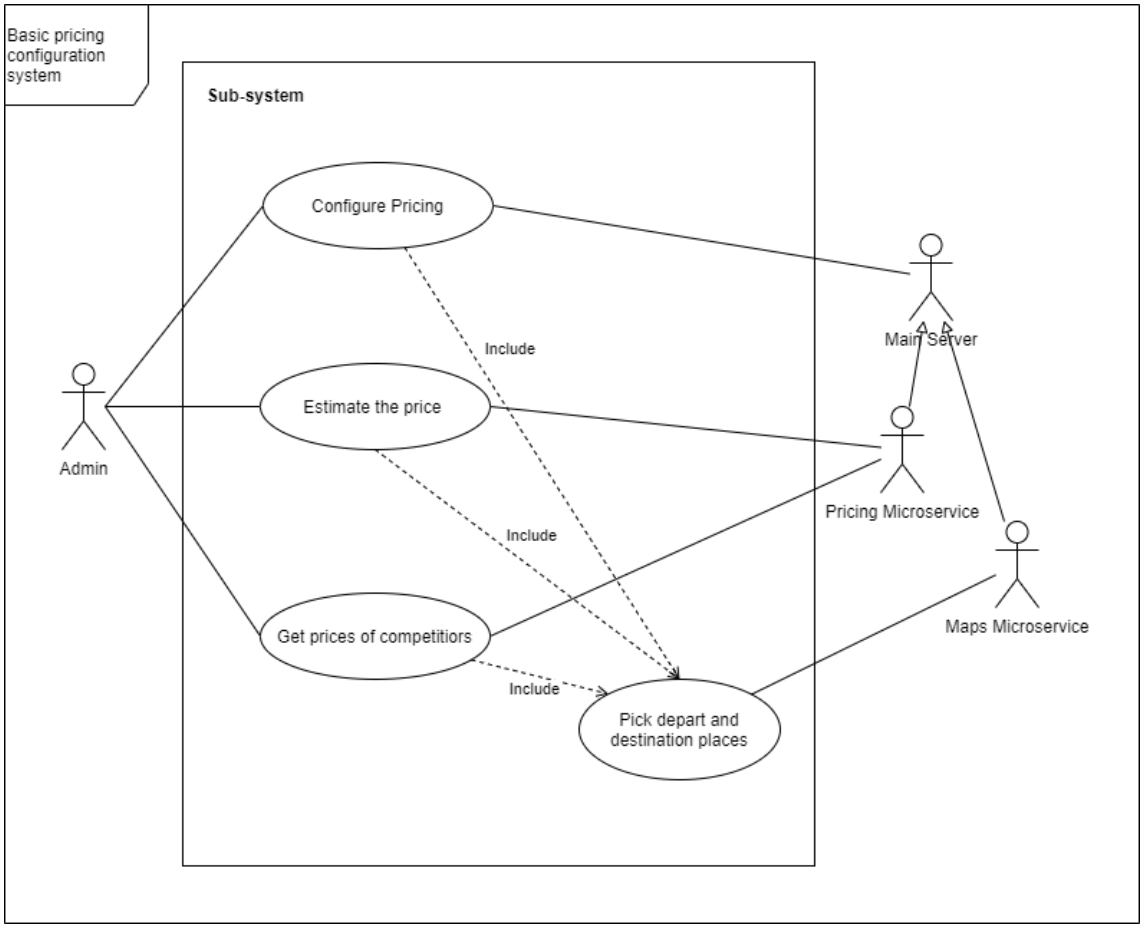


Figure 4.1: Use case diagram of the basic pricing configuration system

The following tables elaborates the use cases "Configure Pricing" and "Get prices of competitors", the use case "authentication" is removed from the diagram to simplify the diagram, but all the authentication operations require authentication.

Use case	Configure Pricing
Objectif	Allows the admin to update pricing calculation settings
Pre conditions	- Login to the dashboard - Choose a pickup and destination points - Get the old configuration from the database
Post conditions	- The system updates the configuration
Scenario	1 - The system shows the update form with old configuration 2 - The admin fills the form with new values 3 - New price gets estimated with new values 4 - The admin compares new price with competitors 5 - The admin saves the new configuration

Table 4.2: Description of the use case "Configure Pricing"

Use case	Get prices of competitors
Objectif	Gives the admin an overview of competitors prices
Pre conditions	- Login to the dashboard - Choose a pickup and destination points
Post conditions	- The system shows prices of some competitors
Scenario	1 - The admin chooses the depart and destination points 2 - The system gets all prices of different apps 3 - Prices are shown in cards

Table 4.3: Description of the use case "Get prices of competitors"

Use-case of the discounts system

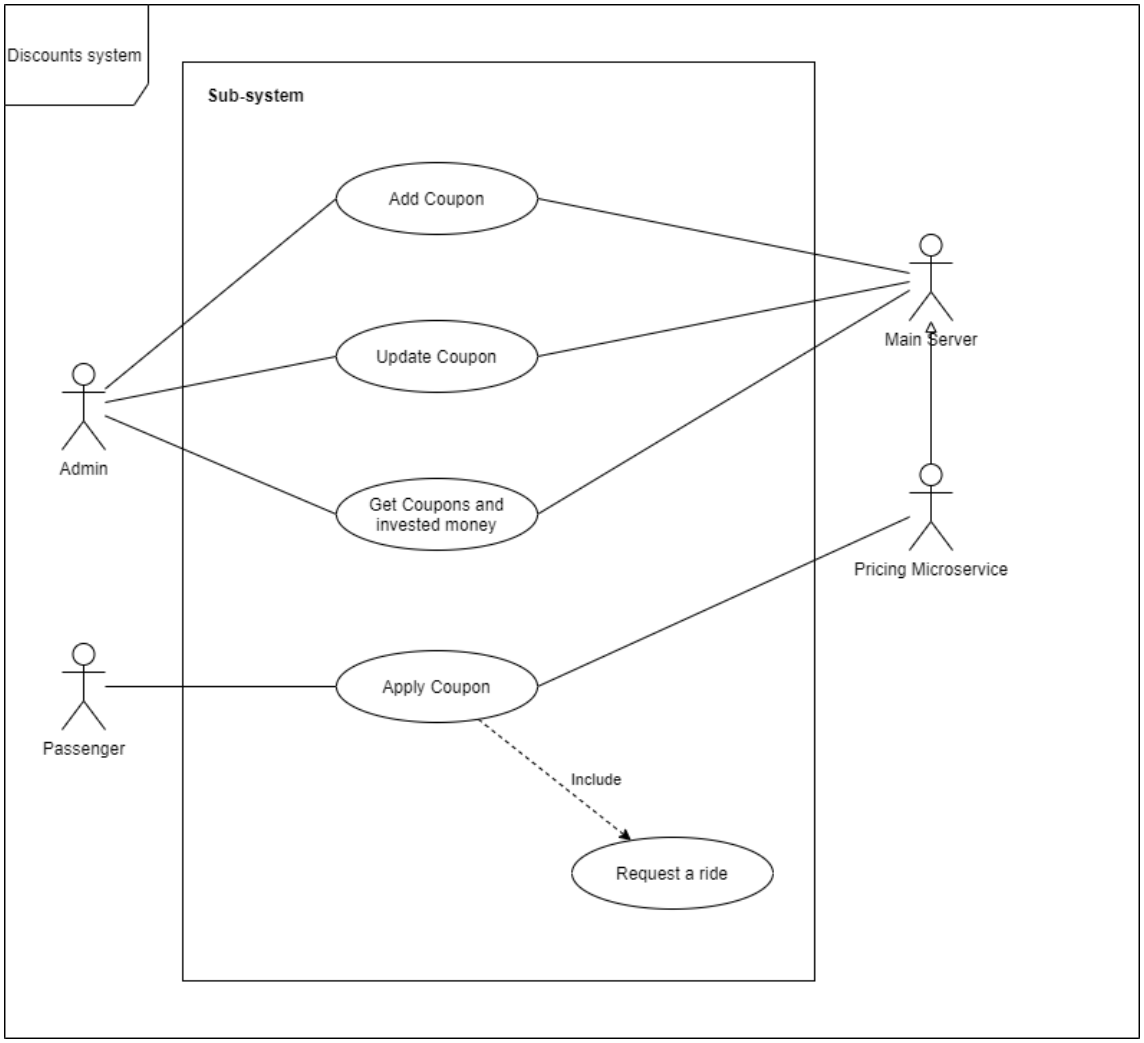


Figure 4.2: Use case diagram of the discounts system

Figure (Fig. 4.2) represents a Use Case diagram for the discounts system. The admin can do basic managing functionalities like adding a new coupon and updating a coupon. He can also get an overview on how much money is invested in those discounts and users who are claiming the coupons.

For a passenger, he can apply a coupon when he requests a ride.

The following tables elaborates the use cases "Apply Coupon" and "Get Coupons and invested money", the use case "authentication" is removed from the diagram to simplify the diagram, but all the authentication operations require authentication.

Use case	Apply Coupon
Objectif	- Apply a reduction in the ride price - Saves money won by the client
Pre conditions	- Request a ride
Post conditions	- The ride fare is updated - The money investment is saved
Scenario	1 - The passenger requests a ride 2 - The system checks the coupon availability 3 - The system checks if coupon is used 4 - new price is sent to the app 5 - Invested money is saved

Table 4.4: Description of the use case "Apply Coupon"

Use case	Get Coupons and invested money
Objectif	- Show coupons - Get redeemers and money saved for each coupon
Pre conditions	- Login to the dashboard
Post conditions	- A table with all coupons is shown
Scenario	1 - Admin login to the dashboard 2 - Coupons with details of each coupon are shown

Table 4.5: Description of the use case "Get Coupons and invested money"

Use-case of the surge pricing system

Figure (Fig. 4.3) shows a Use Case diagram for the surge pricing system. The admin is responsible for activating or dis-activating the service, he also sets up a range for the surge score so it doesn't go beyond certain values. When the service is in the active state, the price estimated by the app puts in consideration the availability of drivers and rides demand in the appropriate zone.

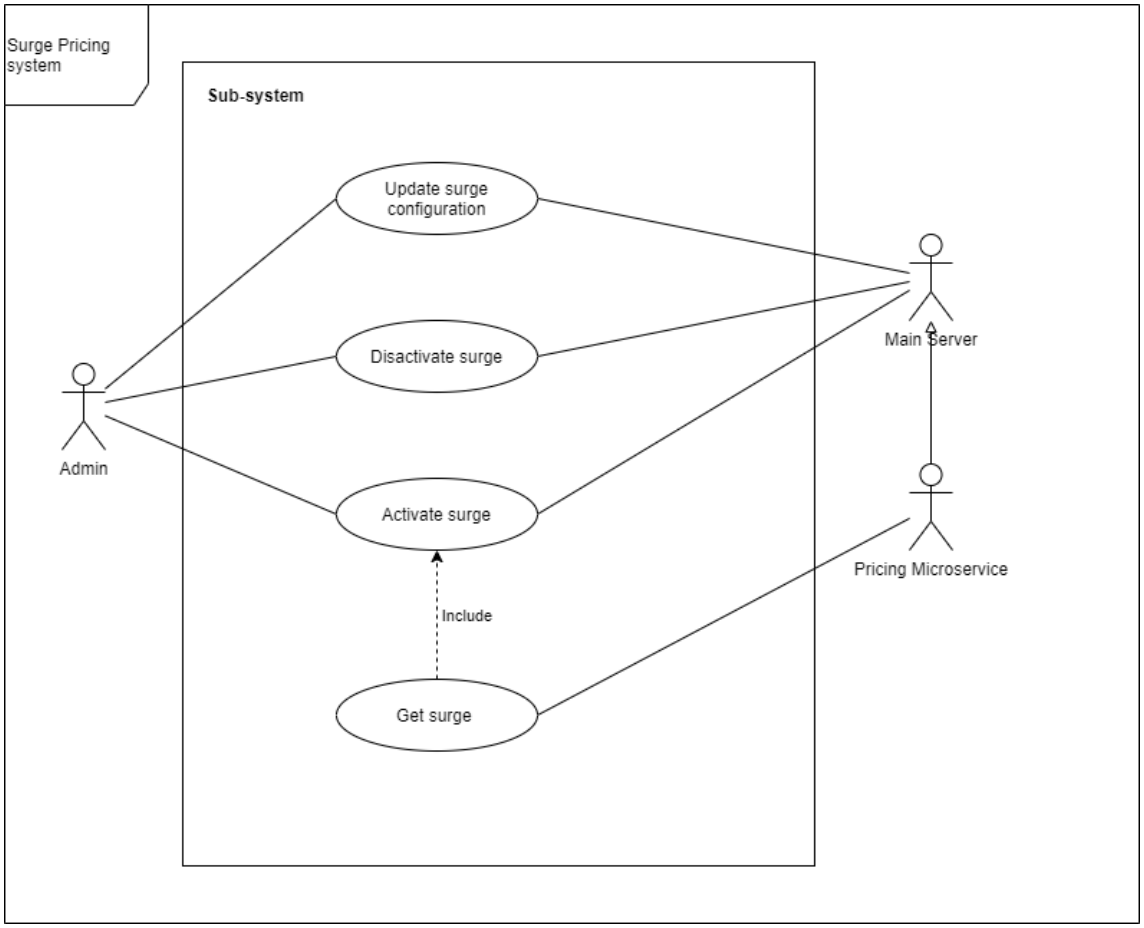


Figure 4.3: Use case diagram of the surge pricing system

The following tables elaborates the use cases "Update surge configuration" and "Get surge", the use case "authentication" is removed from the diagram to simplify the diagram, but all the authentication operations require authentication.

Use case	Update surge configuration
Objectif	- Setup a range that controls the surge pricing
Pre conditions	- Login to the dashboard
Post conditions	- Surge pricing settings are updated
Scenario	1 - Admin login to the dashboard 2 - Admin sets a minimum multiplication score 3 - Admin sets a maximum multiplication score

Table 4.6: Description of the use case "Update surge configuration"

Use case	Get surge
Objectif	- Get a score multiplication for the current zone
Pre conditions	- Activated surge pricing
Post conditions	- multiplication factor is recovered
Scenario	1 - Passenger requests a ride 2 - Pricing system checks the state of the surge system 3 - Multiplication factor is recovered

Table 4.7: Description of the use case "Get surge"

Use-case of the price prediction system

Figure (Fig. 4.4) shows a Use Case diagram for the price prediction system where the admin is responsible for activating or dis-activating the service. As for the price estimation, when the service is in active state the server uses the prediction system to get the duration of a ride and uses it to calculate the price, and when the service is inactive, the server uses google maps to get the duration.

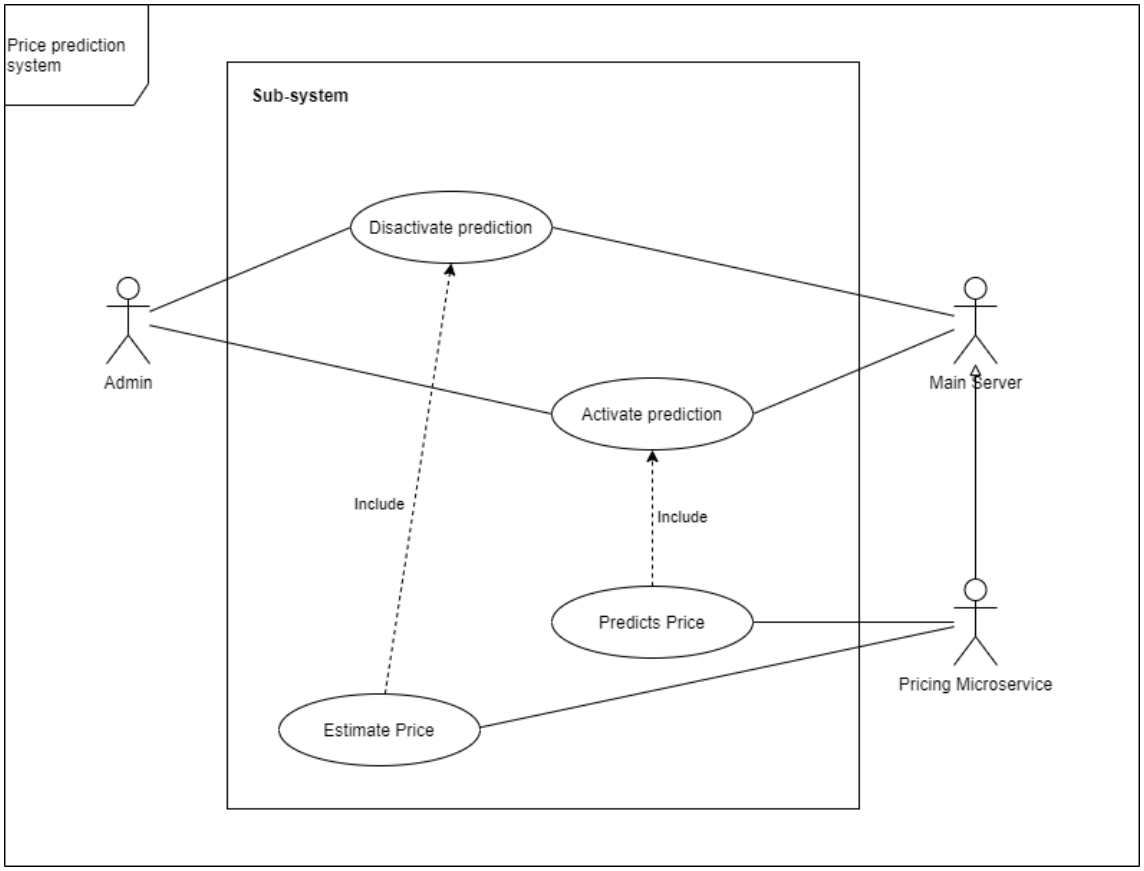


Figure 4.4: Use case diagram of the price prediction system

4.2.3 Sequence diagram

Dynamic pricing groups many tasks for both passenger side, and admin side. In this section we present diagrams of processes that we judge are important.

Sequence diagram for pricing configuration

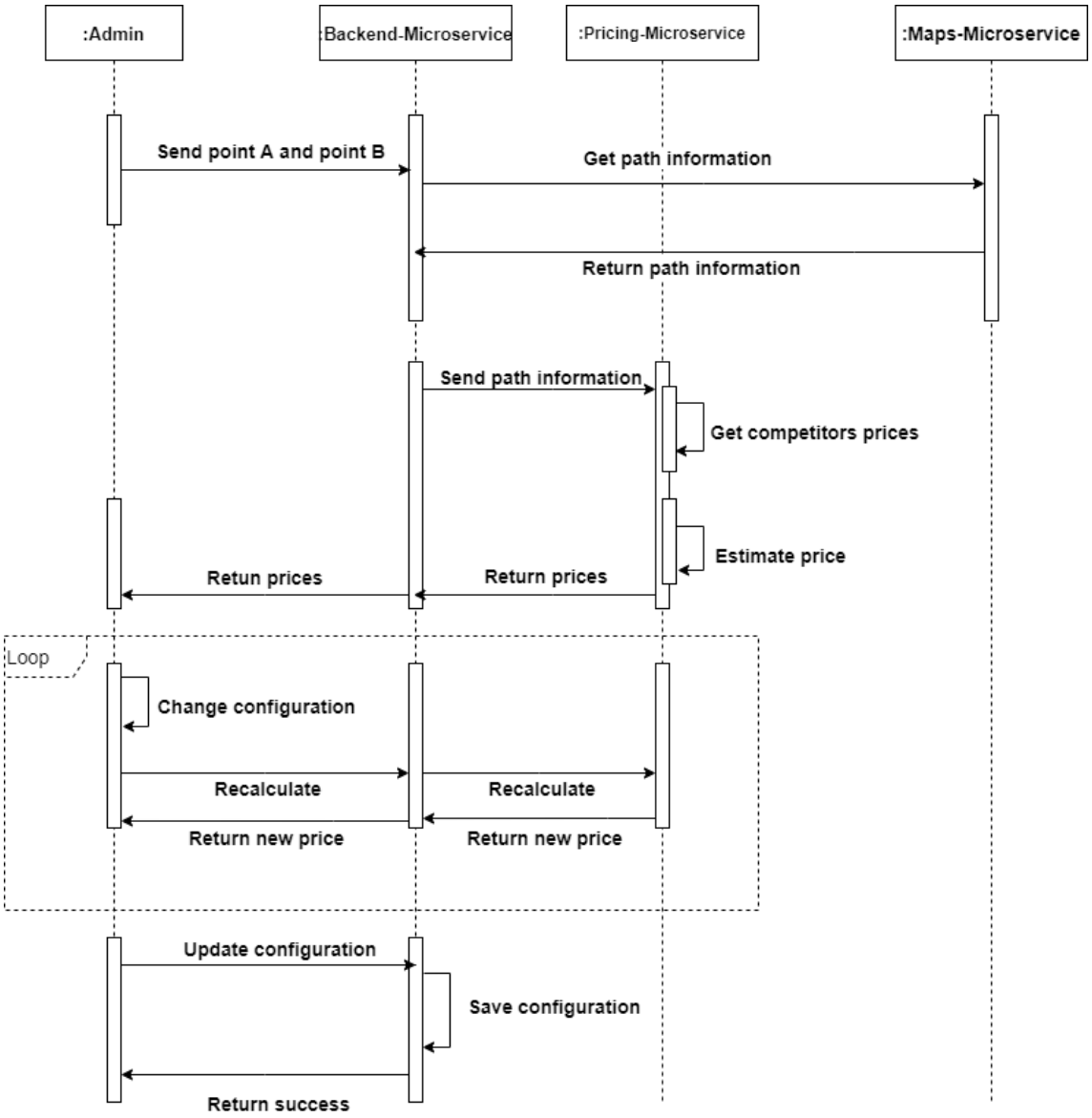


Figure 4.5: Sequence diagram of pricing configuration process

Pricing configuration consists of many tasks including managing categories and managing periods, but the important part is the basic configuration. The admin chooses

two points, depart and destination, then the system shows prices of both our application and other competitors. An overview of these prices allows the admin to decide on his own configuration, and at this point he starts tweaking the settings to get to the appropriate configuration.

Sequence diagram of ride confirmation

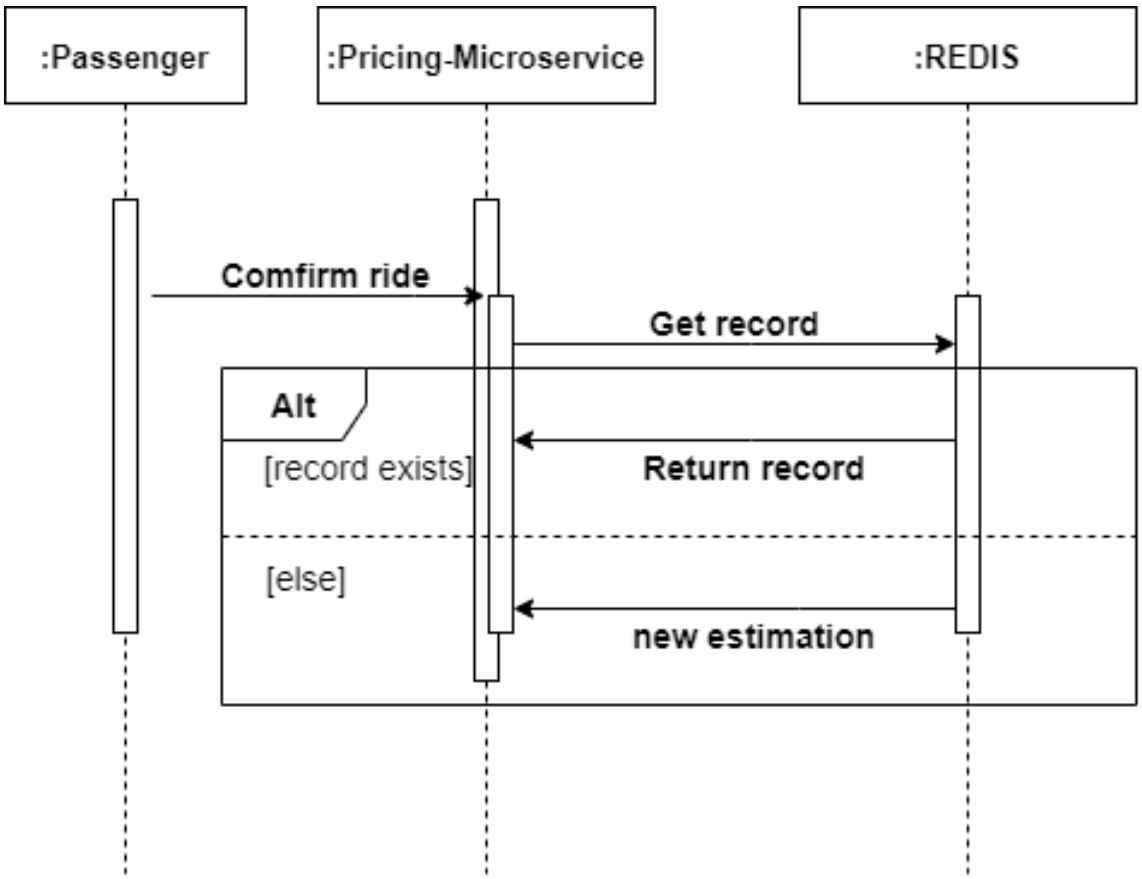


Figure 4.6: Sequence diagram of the ride confirmation

The price is estimated after requesting a ride, which means before confirming the ride. One of the flows of DOD applications is that the user can request a ride in the morning when the price is low and keep the application running, then at night, he can confirm the ride with that price.

To solve this problem we though about saving a record when a ride is requested and generating an id for that request, then when the ride is confirmed, the system checks the record that has a lifetime of some seconds, when record isn't found the

system runs a new estimation.

Sequence diagram for pricing estimation

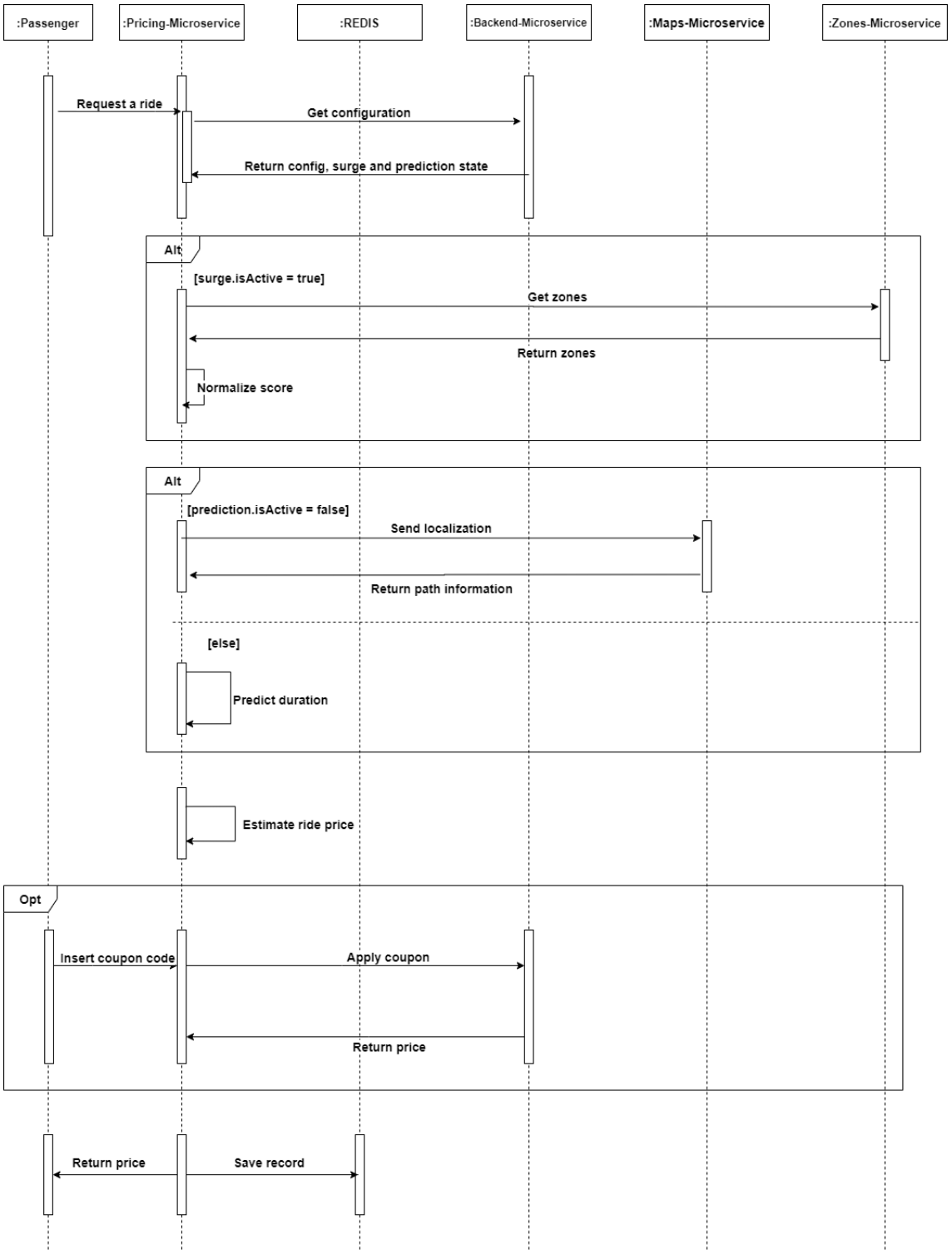


Figure 4.7: Sequence diagram of the price estimation process

The most important process of our solution is the price estimation. When the user requests a ride the pricing system gets the pricing configuration, then checks for surge and prediction state.

When surge pricing is active, the service communicates with zones service that returns 3 zones, the zone where the user is located and 2 near zones that have high demand, then the system detects the current zone and its score. The score is then normalized to be used in the price estimation.

When the prediction system is active, the pricing service uses it to get details about the requested path, in case it is not active it uses maps service.

The user can apply a coupon code that applies a discount on the price, the system then returns the new price and saves how much money will the company pay for the client.

4.3 Conception

Before giving a general overview of the system conception, we are going to define some notions on which the conception is based.

4.3.1 Architectural conception

Architecture based on Micro-services model

Coding gets harder when adding more functionalities. It can be difficult to know where a change needs to be made because the code base is so large, and although the base of monolithic codes is clear and modular, its arbitrary limits during manufacture break too often. The code linked to similar functions starts to spread everywhere, making bug fixes or the implementation of new features more difficult.

Within a monolithic system, we fight against these forces by trying to ensure making our code more consistent, often by creating abstractions or modules. Cohesion [1] is an important concept when we think of micro-services.

Micro-services take the same approach as independent services. We delineate

[1] Cohesion : The willingness to group associated code together

the service by the boundaries of the trade, which makes it obvious where the code is for a given functionality. By keeping this service focused on an explicit limit, we avoid the temptation for him to grow too much, with all the associated difficulties that this can introduce, generally micro-services should be designed in such a way that they are developed from zero in a single two-weeks sprint [4].

REST

REST is the acronym for "REpresentational State Transfer", it is an architectural style for distributed hypermedia and it was first featured by Roy Fielding in 2000 in his famous thesis, known for its decoupling and light communication between the producer of a resource (Server) and its consumer (Client) [2]. Generally used with the HTTP protocol, REST is the preferred mode for creating interfaces for applications (API). REST has 6 main principals that are listed below:

* **Client–server :** By separating the user interfaces from the data storage and the back-end, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying server components.

* **Stateless :** Each request from the client to the server must contain all the information necessary to understand the query, and cannot take advantage of any stored context on the server. The session state is therefore kept entirely on the client side.

* **Cacheable :** Cache constraints require that the data in a response to a query are implicitly or explicitly labeled as cacheable or non-cacheable. If a response can be cached, a client cache has the right to reuse that data for subsequent equivalent queries.

* **Layered system :** The layered system style allows an architecture to be made up of hierarchical layers by constraining the behavior of components so that components cannot "see" beyond the immediate layer with which they interact.

* **Code on demand :** REST allows the functionality of the client to be extended by downloading and by executing code in the form of applets or scripts. This simplifies customers by reducing the number of features to pre-implement.

* **Uniform interface :** In order to achieve a uniform interface, several architectural constraints are necessary to guide the behavior of the components. REST is

defined by four interface constraints: identification of resources; handling resources through representations; self-descriptive messages; and hypermedia as the engine of the application state.

4.3.2 Classes diagram

The figure (Fig. 4.8) presents the class diagram of our system conception. In this diagram, we present the most important classes, where each service groups together some of them.

The 2 main classes of the system are competitors class that holds the prices of competing DOD services, and estimator class responsible for price estimation. They both use the maps classes to get path details.

Estimator class This class has 4 main functions:

*** Price calculation function** that sets an equation of the price decided by marketing team.

*** Estimation function** runs when requesting a ride and estimates the price based on the configuration, surge service, prediction service and coupons.

*** Marketing getter** is a function similar to the estimation function that handles pricing for marketing team.

*** Record function** runs when confirming a ride and checks for estimation record existence, if it doesn't exist, it runs a new estimation.

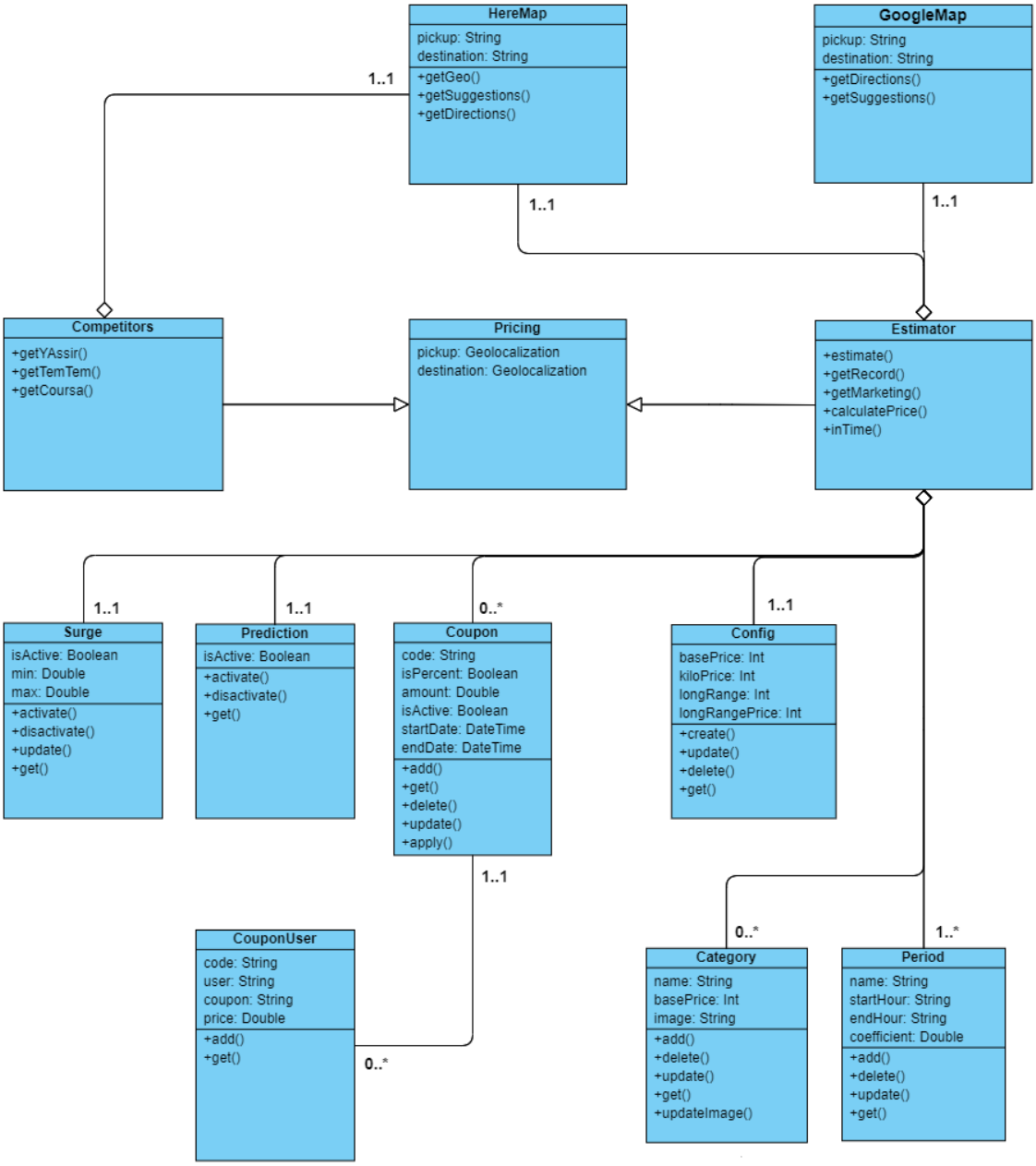


Figure 4.8: Classes diagram of the dynamic pricing system

4.3.3 Activity diagram

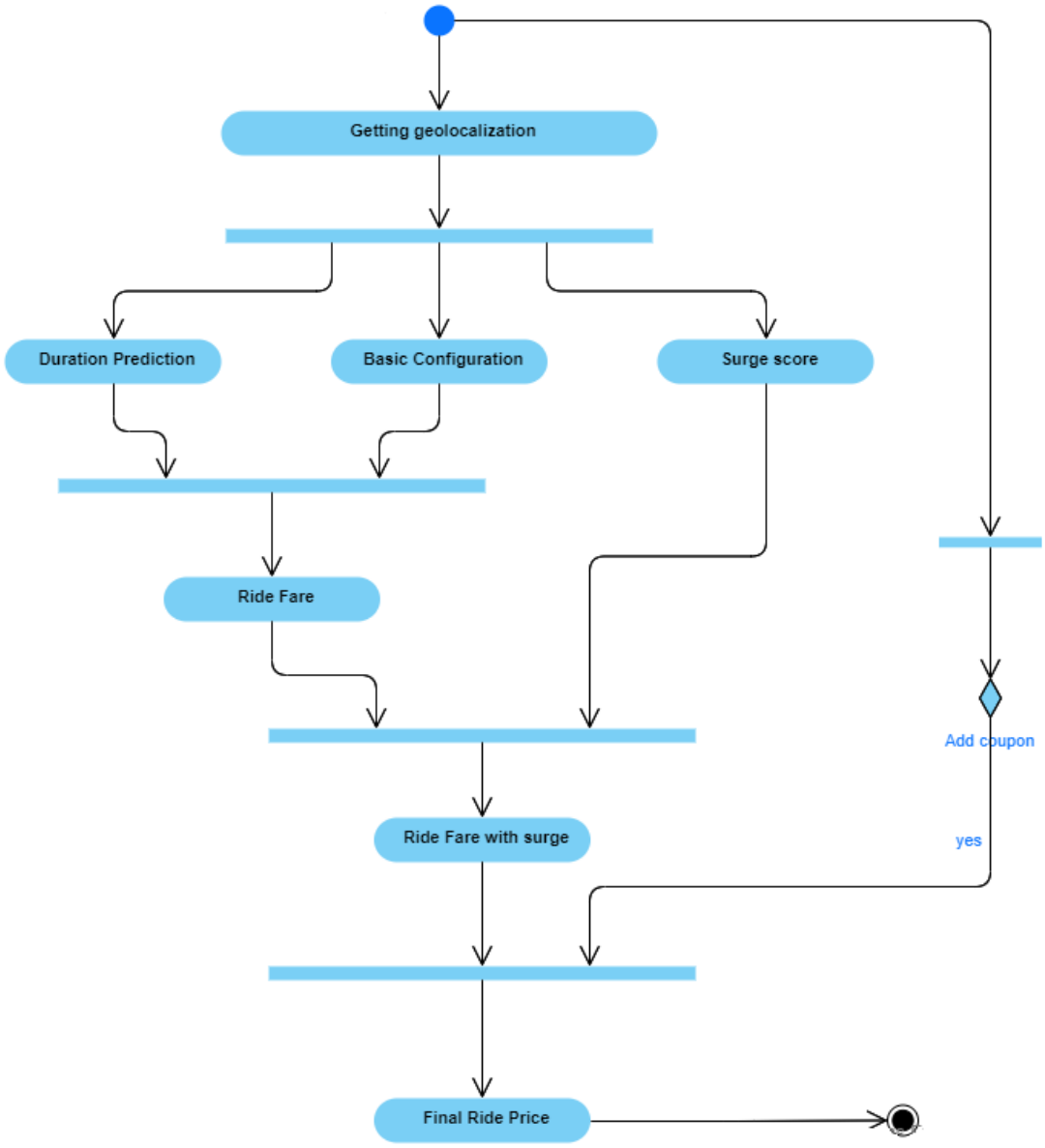


Figure 4.9: Activity diagram of the dynamic pricing system

The following activity diagram gives a vision of the sequence of activities specific to an operation or use case, in figure (Fig. 4.9) it shows price estimation activities in

the ideal form where surge and prediction services are active.

The activity diagram is attached to a class category and describes the course of activities in this category. It indicates the share taken by every object in the execution of a job. It will be enriched by the conditions sequence. The activity diagram allows us to see the internal behaviors of the system.

4.4 Conclusion

In this chapter, we have identified the needs that we will try to meet and satisfy in the development phase. We used the use case and sequence diagrams to express the functionality desired by the system. Subsequently, we modeled the system data through a class diagram and the business aspect through an activity diagram.

Chapter 5

Ouigo Pricing: A system that manages the dynamic pricing for Ouigo DOD application

Sommaire

5.1 Introduction: Presentation of the system

The system is called "Ouigo Pricing". It helps the marketing team decide on their pricing strategy, control the pricing of rides, and also calculates the price based on surge zones, and the prediction system that decides the duration of the ride based

on previous rides using machine learning. This system is developed in NodeJS and Python.

5.2 Architecture of the system

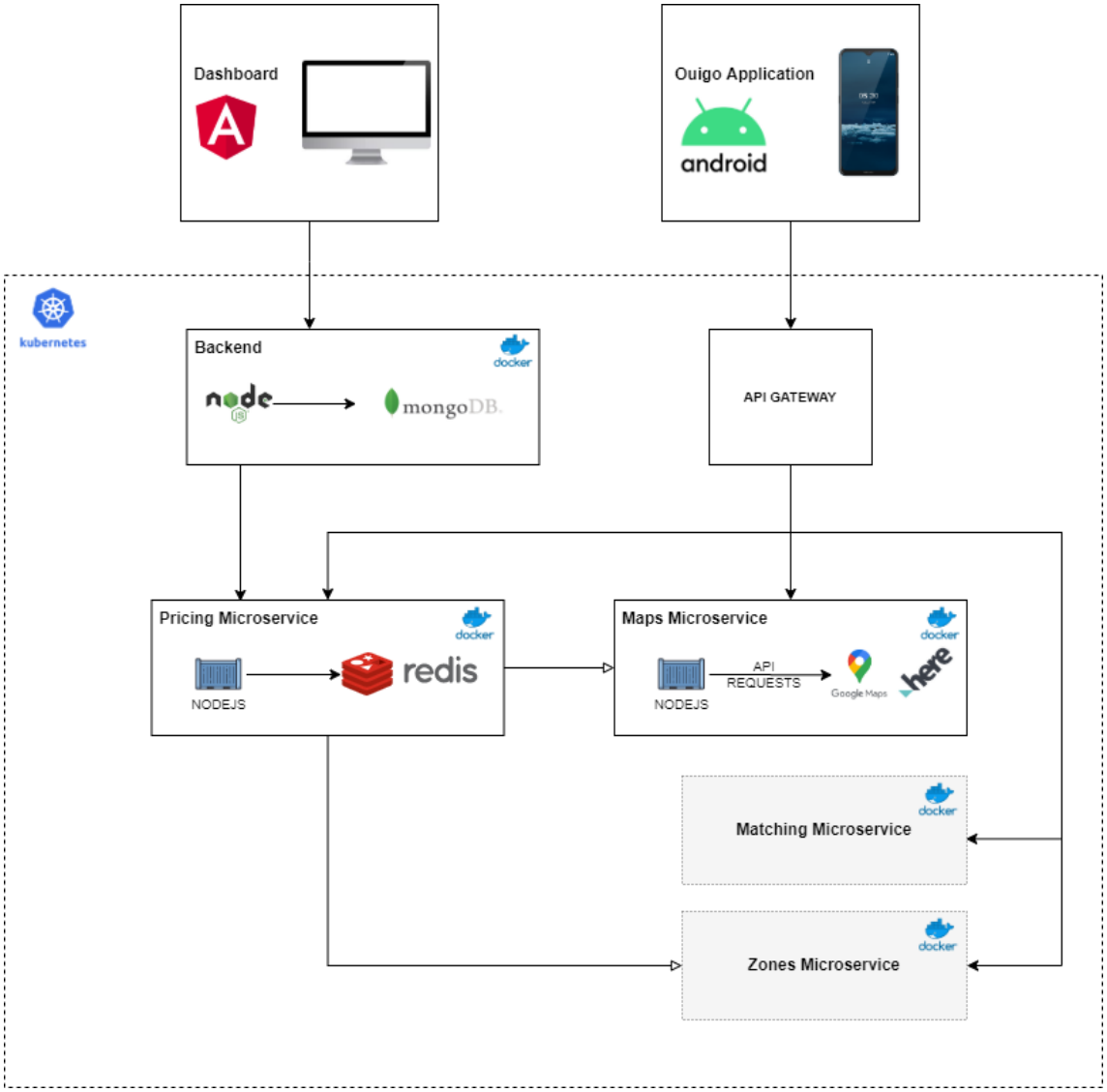


Figure 5.1: Architecture of dynamic pricing system of Ouigo

The pricing of DOD application requires both the pricing strategy, pricing factors, and the control of the marketing team, for that purpose it needs the implementation of different systems. Our prototype uses the approach of micro-services, which are the maps, micro-service that handles everything related to Geo-localization, and

the pricing micro-service that handles everything related to ride fare estimation. In addition to that, the system includes the control dashboard.

Figure (Fig. 5.1) shows the architecture of our system, that we will define with details in this following parts.

5.2.1 Maps micro-service

Maps service groups together all the functionalities used by the pricing system, and other micro-services, these functionalities are getting suggestions that give places list when typing first letters, getting Geo-localization latitude and longitude from text, and getting directions including duration, distance and polygon.

The implementation of this micro-service requires doing a study on different maps APIs to decide on what fits the company.

APIs Benchmark

Google Maps API has all the functionalities needed, on the other hand the cost of those functionalities which is 8$ per 1000 requests is quite challenging. Because of that we've put the service in comparison with other services like HereAPI and BingMaps in terms of features, data and styling, and cost.

5.2.2 Pricing micro-service

Pricing service is the main part of our system, developed using NodeJs, it gets depart and destination latitude and longitude as input from the mobile application and returns final exact prices for different categories. The service has 4 functionalities:

Basic pricing

Before selecting a ride, the user chooses the car type, and price changes in different periods of the day, the service uses MongoDB to get this data, and uses it to calculate the basic price.

The price also depends on different factors like weather, traffic, etc., which can be grouped together in duration factor, and for this, the service has the prediction

functionality.

Price prediction

The changing factor that groups many aspects is the duration of the ride. Ouigo pricing uses Maps service, as a default duration predictor, but since the Google Maps API costs much, the pricing service uses it's own prediction system using machine learning and Python language. This sub-system checks if the prediction is active or not, in case it's active, we use our own model, and in case it's it's inactive, we use Google's model.

Surge pricing

Surge pricing sub-system augments the price based on the drivers availability and rides request. To retrieve this information the pricing service communicates the depart location to the zones micro-service, which was developed by our colleagues **yacine BENKAIDALI** and **Akram BENRANDJA**, and it basically divides the region into zones and returns 3 zones where each zone has a demand score and one of those zones is the depart zone. The sub-system then retrieves the depart zone and normalize the score based on marketing configuration, so we can use it as multiplication factor.

Applying coupon

When the price is estimated, the passenger can redeem a coupon code, this sub-system takes that code and checks if it's active. When the coupon is active, the sub-system return the new price with discount to the mobile application, and saves the reduction rate of the price, and the number of users who redeemed that code.

Price estimation is recorded in Redis database to create the future data for the pricing prediction.

5.2.3 Web application

After refactoring the code of the existing web application, we implemented the pricing control functionalities, which allows full control over the pricing service, those functionalities are already discussed in the previous chapter.

Back-end

The back-end was developed with NodeJs, one of the reasons we had to stick with the technology, and it has its own MongoDB database.

Dashboard

Since the dashboard was built with an old Angular version, we had to stick with the technology to not invest lots of time in the upgrade process. We implemented the pricing interfaces ready to be assigned to the marketing role.

5.3 UI/UX

These next figures show different interfaces for controlling the pricing service.

In figure (Fig. 5.2) admin adds filters that user will use later to get the perfect matching driver.

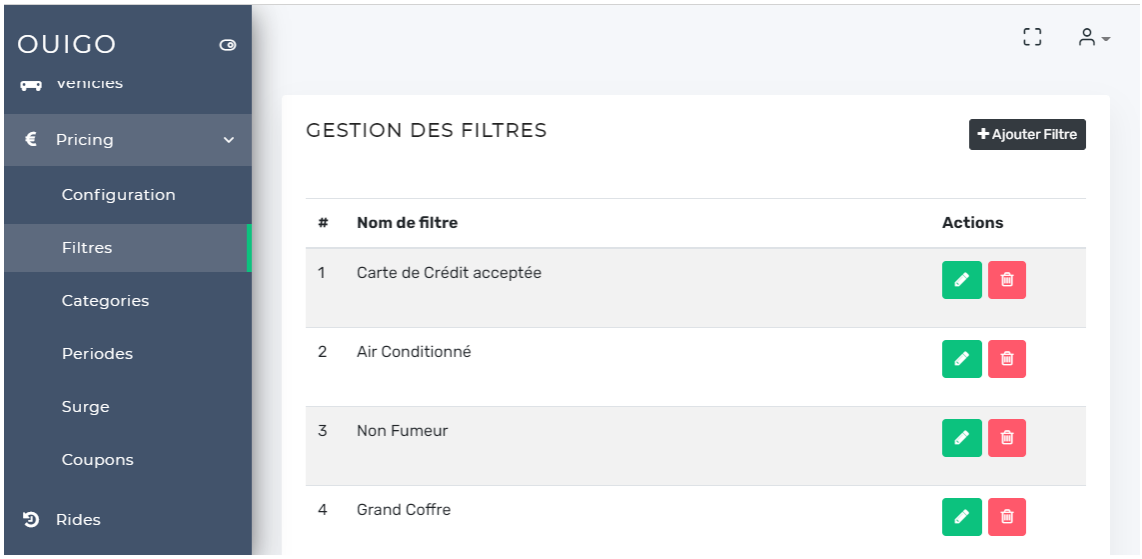
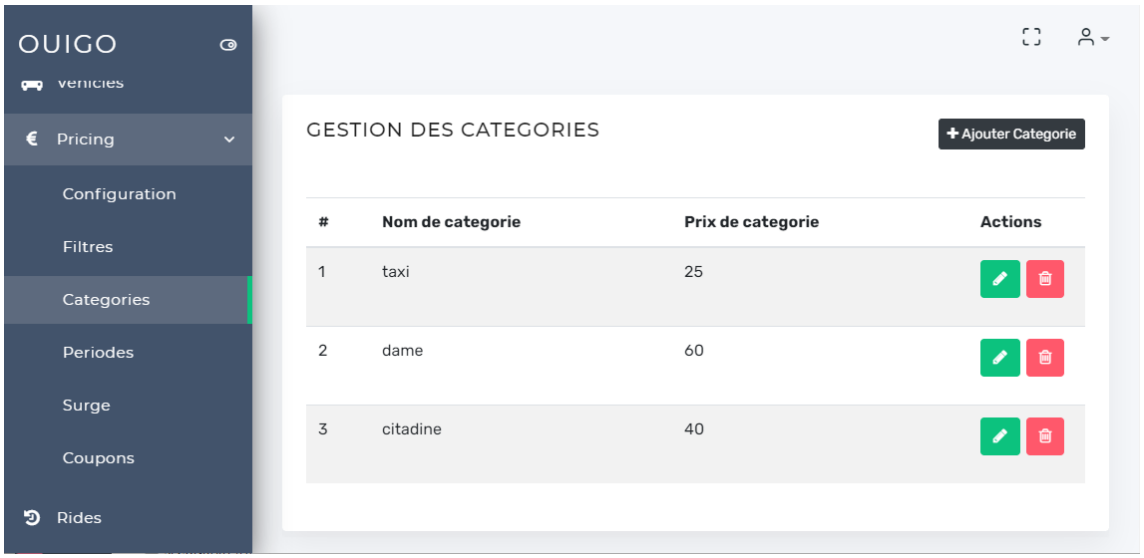
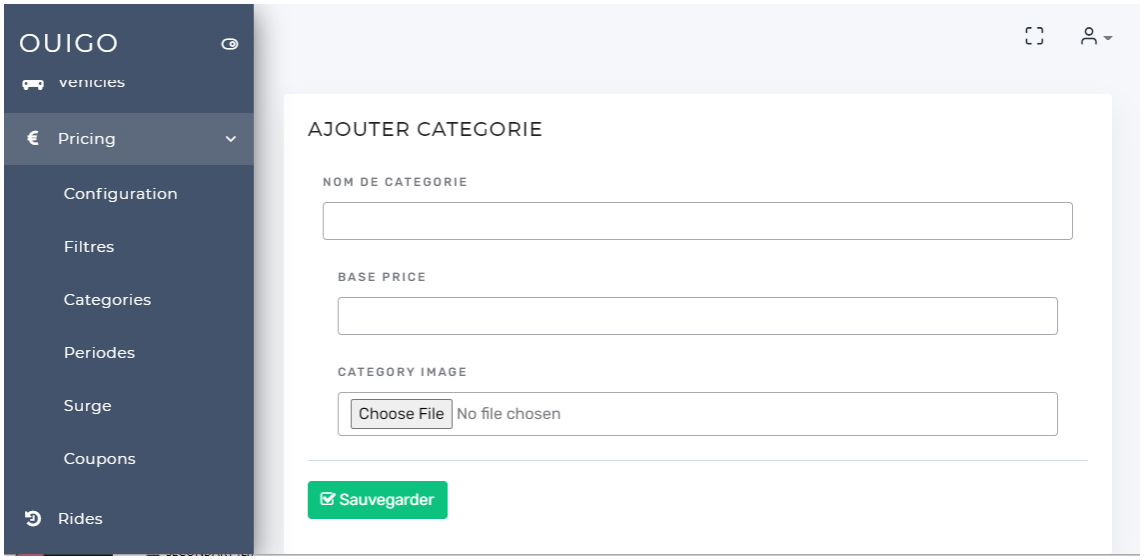


Figure 5.2: Web UI - Managing filters



(a) Viewing Categories



(b) Adding Categories

Figure 5.3: Web UI - Managing categories

Figure (Fig. 5.3) presents the basic CRUD[1] functionalities for managing categories and setting their prices, where each type of car has its own extra price.

In figure (Fig. 5.4) the admin can manage periods of the day, for DOD applications the price gets high when the risk for the driver gets high and when the availability is low, so every period of the day has its own multiplication factor.

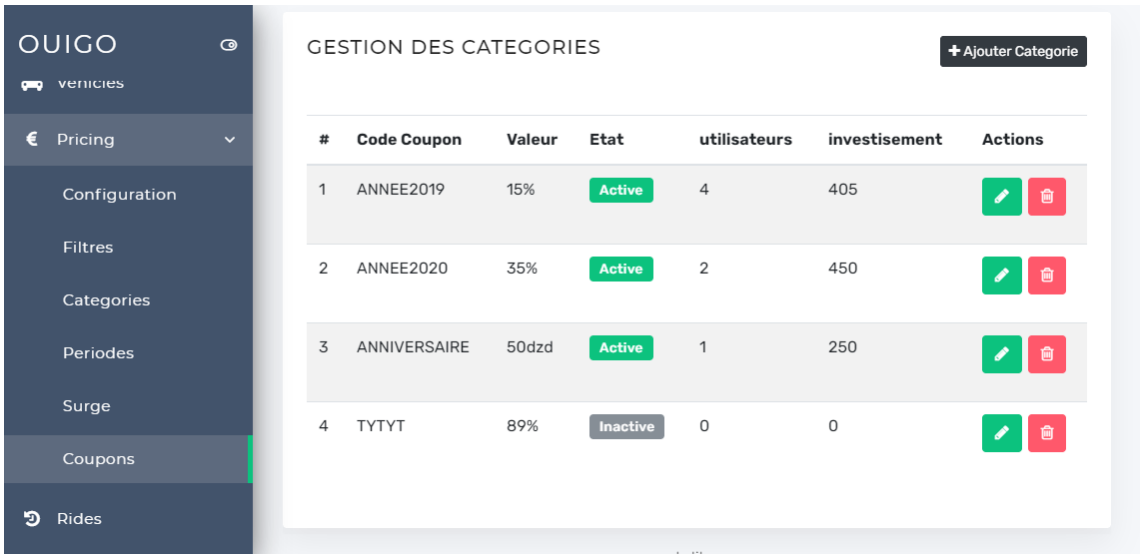
[1]CRUD: Create, Read, Update, Delete

(a) Viewing periods

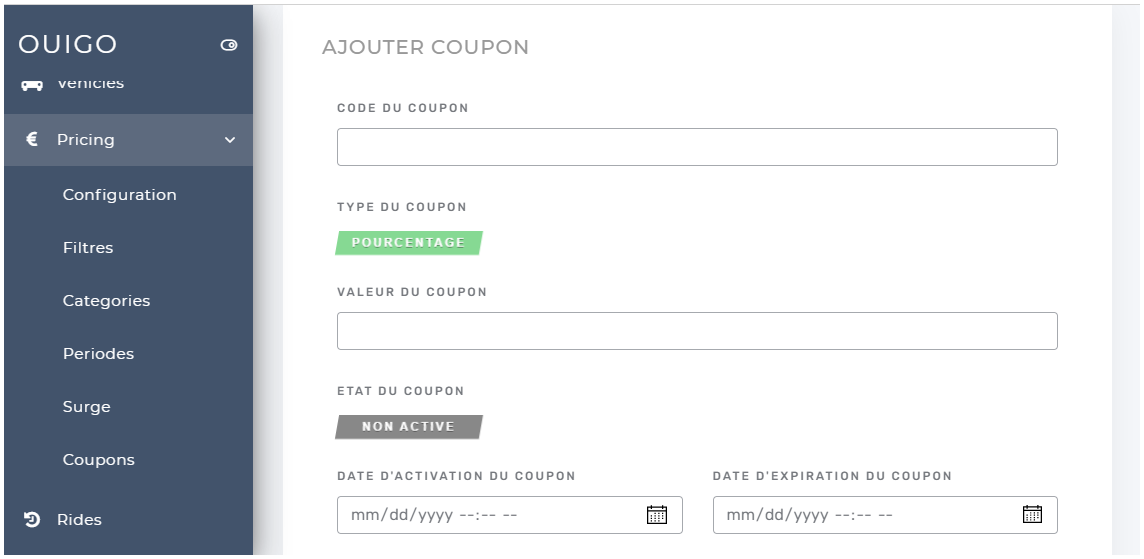
(b) Adding period

Figure 5.4: Web UI - Managing periods

Figure (Fig. 5.5) shows the interfaces for the coupon system, when viewing the list of coupons in figure (Fig. 5.5a), the admin gets extra information about the users who redeemed the code, and how much money the company invested in that coupon.



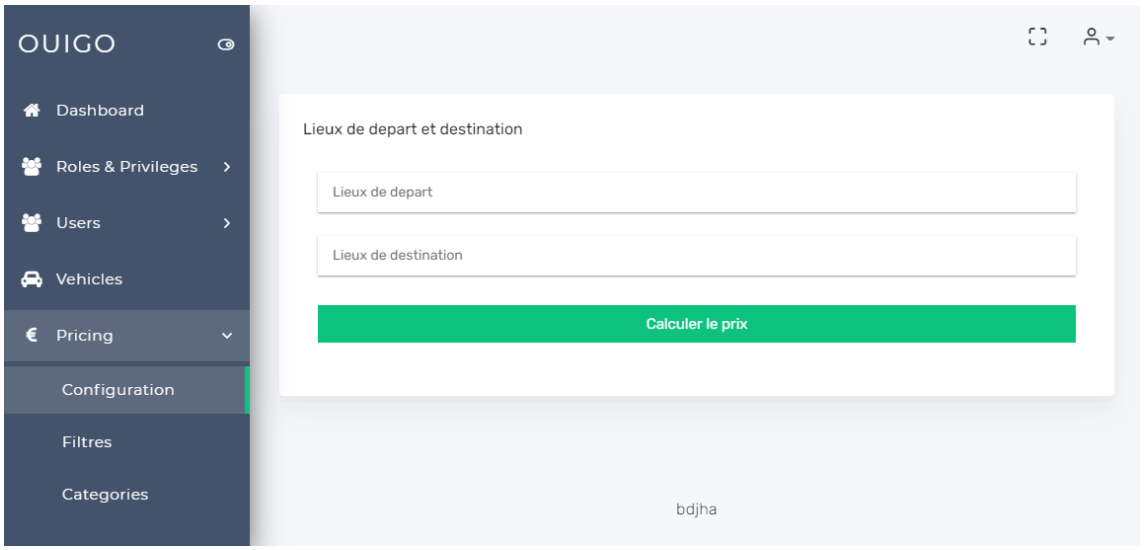
(a) Viewing coupons



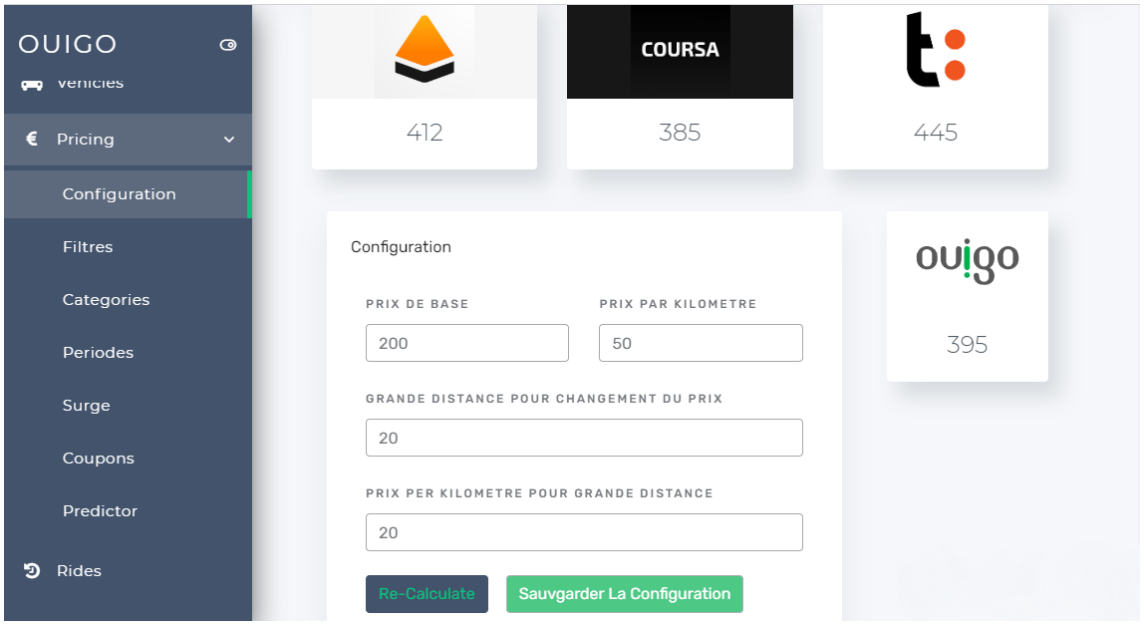
(b) Adding coupon

Figure 5.5: Web UI - Managing discounts

As for configuring the pricing settings, the user starts by choosing a pickup and destination location as seen in figure (Fig. 5.6a) then click the calculate button. The admin then is prompted with figure (Fig. 5.6b) where he can find the prices of competitors, and the price of our solution, then he starts tweaking the settings and re-calculate the price until he is satisfied, the configuration is then saved.



(a) Choosing depart and destination



(b) Showing competition and configuring prices

Figure 5.6: Web UI - Basic pricing configuration

Figures (Fig. 5.7) and (Fig. 5.8) shows the settings for both surge pricing and price prediction system.

Figure 5.7: Web UI - Surge configuration

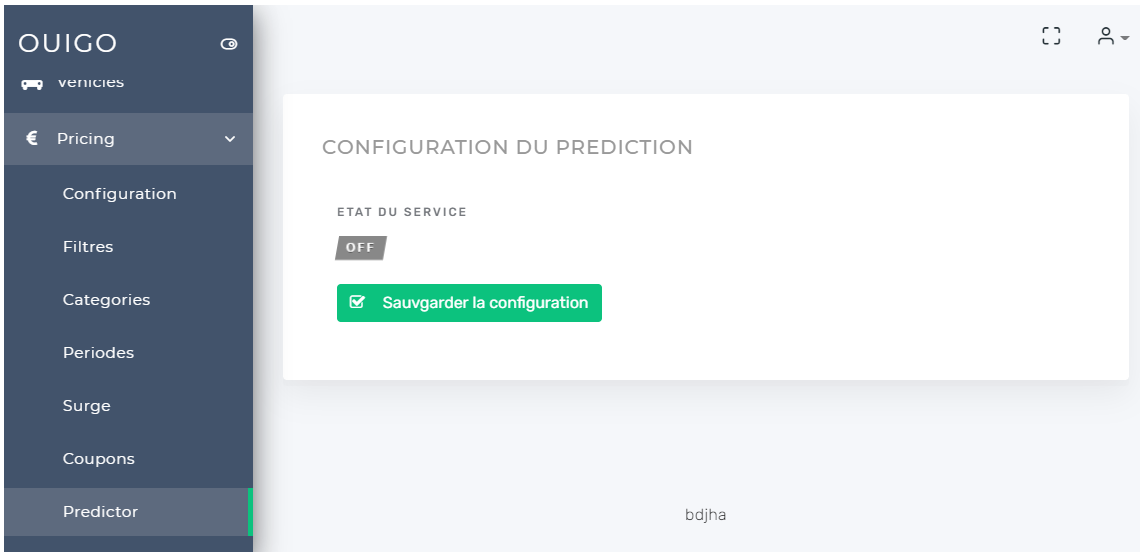


Figure 5.8: Web UI - Price prediction configuration

5.4 Conclusion

In this chapter we have presented our system "Ouigo Pricing" with its general architecture and UI. The following chapter shows the development and deployment environment of this system, the experimentation and analysis of results obtained.

Chapter 6

Experimentation

Sommaire

6.1 Introduction

After having presented the theoretical and technical conception of our solution, we begin the stage of carrying out in development, collaborating and building our ready to deploy web application and the micro-services for maps and pricing. In this part, we will present the technical solution and justify the chosen development environment, the technical choices used and the languages adopted.

6.2 Development environment

6.2.1 Languages and frameworks

In this part we present the different technologies we used and justify each one of them.

NodeJS

NodeJS is a server-side technology based on Google's V8 JavaScript engine (v8 engine). It is a highly scalable system that uses asynchronous calls rather as separate threads or processes. It is ideal for frequently consulted, but easy to calculate web applications.

Since most modern kernels are multi-threaded, they can handle multiple operations running in the background. When one of these operations is completed, the kernel notifies NodeJS that the appropriate callback can be added to the queue to be possibly executed. This advantage makes the framework perfect for real-time systems such as the pricing service. Here are some major advantages of this language:

* Node.js offers easy scalability
* Easy to learn
* Known to provide high performance due to its asynchronous nature.
* Perfect for real time systems.
* Support from a great community.
* Its rich NPM ecosystem

Angular

Angular[1] is a client-side, open source, TypeScript-based framework, and co-directed by the "Angular" project team at Google and a community of individuals and companies. Angular is a complete rewrite of AngularJS, a framework built by the same team.

The technology was used in the existing project, and in terms of time and efforts, sticking with the same technology was the ore reasonable choice.

Python

Python[2] is a high-level, interpreted, and versatile dynamic programming language that focuses on code readability and easy syntax. Python is widely used in large organizations due to its multiple programming paradigms. It has a complete and a large standard library which has the automatic management of memory and dynamic features.

Some python libraries have been a great help to us during the learning and discovery phase. We can give as an example:

* **Matplotlib:** it is used to draw curves and histograms.

* **Shapely:** it is used for manipulating features such as polygons, polylines and dots.

* **Numpy, Pandas:** Libraries for data structures, used to manipulate arrays and matrices.

6.2.2 Data Base Management System

MongoDB

MongoDB[3] is a document-oriented database management system that can be distributed across any number of computers and does not require a predefined data schema, it is also a part of the NoSQL movement. It was used to store all data in the existing project and we continued to use it while refactoring.

[1] https://fr.wikipedia.org/wiki/Angular
[2] https://fr.wikipedia.org/wiki/Python_(*langage*)
[3] https://fr.wikipedia.org/wiki/MongoDB

Redis

Redis, which stands for Remote Dictionary Server, is a key-value data storage system in memory, open source and fast, to be used as a database, cache, message broker, and waiting line. Redis is now offering response of less than a millisecond allowing millions of requests per second for real-time applications.

In our case we use it to store pricing data to be used in prediction system, and also so we can manipulate the persistence of that data.

6.2.3 IDEs

Visual Studio Code

Visual Studio Code is an extensible code editor developed by Microsoft for Windows, Linux and macOS. It is based on Electron, a framework used to deploy Node.js applications for the desktop running on the Blink engine. Although it uses the Electron framework, the software does not use Atom but uses the same publisher component (named "Monaco") used in Azure DevOps (formerly known as Visual Studio Online and Visual Studio Team Services).

PyCharm

It is an IDE used for programming in Python. It allows code analysis and contains a graphical debugger. We used PyCharm for the development of python scripts and data processing and even to visualize the data.

6.3 Collaboration environment

After 3 months of our internship at Valley Solutions, there was that sudden disaster event of Covid-19. The work turned from office presential work, to remote work from home, and it was at this time when collaborative tools played a huge role in the success of the project.

6.3.1 GitLab

GitLab is a free, git-based forge software that offers collaboration functionalities, a bug tracking system, continuous integration, and continuous deployment. Developed by GitLab Inc and created by Dmitriy Zaporozhets and Valery Sizov, the software is used by several large IT companies including IBM, Sony, NASA, Alibaba, Oracle, etc. We used it mostly to work together on the web application features.

6.3.2 Trello

Trello is an online project management tool, launched in September 2011 and inspired by Toyota's Kanban method. It is based on an organization of projects in boards listing cards, each representing tasks. The cards are assigned to users and are mobile from one board to another, reflecting their progress.
We used trello board to keep on track all tasks of the implemented project and to keep it visible with other colleagues as seen in figure (Fig. 6.1).

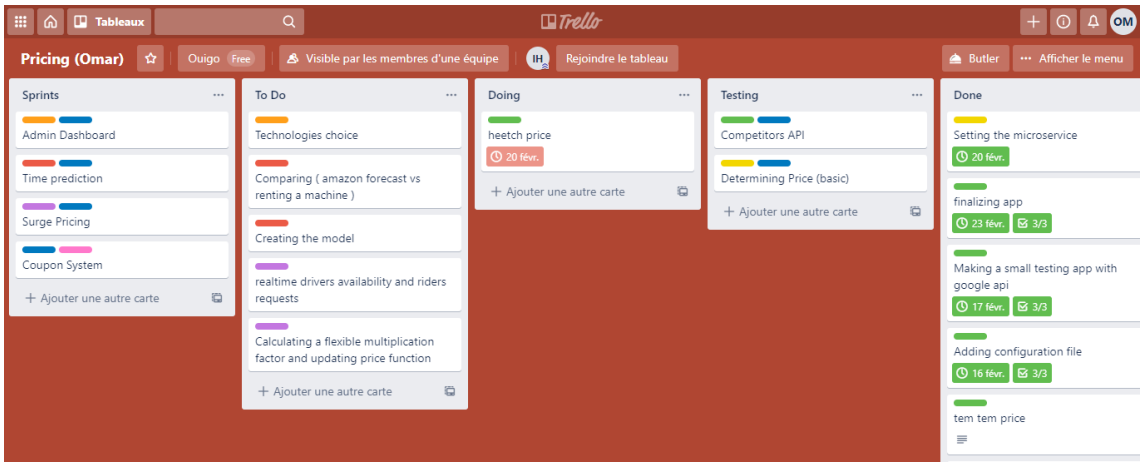


Figure 6.1: Trello board - Ouigo Pricing tasks

6.3.3 Slack

Slack is a proprietary collaborative communication platform (SaaS) as well as project management software. It works like an Internet Relay chat organized into channels corresponding to as many discussion topics. The platform also allows us to keep track of all exchanges.

Slack was our way of communication, sending files, sharing resources and asking for help and feedback through the whole time of the internship.

6.4 Deployment environment

The deployment of all the micro-services and web application was the responsibility of the IT team leader using Amazon Web Services and Kubernetes. Our job was setting up a ready to deploy solution using Docker technology.

6.4.1 Docker

Docker is a tool that can package an application and its dependencies in an isolated container, which can be run on any server.

It's not virtualization, it's containerization, a lighter form that relies on parts of the host machine for its operation. This approach increases the flexibility and portability of running an application, which will be able to run reliably and predictably on a wide variety of host machines, whether on the local machine, a private or public cloud, a bare machine, etc.

6.5 Results analysis

For our system, evaluation is based on the quality of results, the precision and the time of response.

6.5.1 Basic price

Basic pricing is the first feature of Ouigo pricing that was tested, deployed into production. After being tested, and doing a simulation of a real scenario with all different cases, including the flaws of other platforms, the results were as requested. The basic price takes up to 1 second to return a result.

6.5.2 Surge price

The zones micro-service was deployed on an Amazon Web Service (AWS) cloud server. AWS Step Function service allows the training of the model daily, at a certain time, and despite the fact that the precision is already good, daily training would make it consistently improving. The surge pricing takes up to 3 seconds to return a result, but the time can change by enhancing the server's settings.

6.5.3 Price prediction

For our work we experimented 3 regression algorithms, Multiple Linear Regression (MLR), Random Forest Regression (RFR) and XGBoost Regression (XGBR). In the following table 6.1 we present a comparison between the 3 algorithms in terms of Root Mean Square Error (RMSE) and accuracy.

Algorithm	**Training Time**	**RMSE**	**Score**
Multiple Linear Regression	Fast	2750.0010086255475	0.07
Random Forest Regression	Slow	1252.8439283328469	0.80
XGBoost Regression	Very Slow	1032.834868843036	0.86

Table 6.1: Comparative table between regression algorithms for price prediction system

After evaluating the 3 approaches, we can see that the MLR training time is so fast but the accuracy is very low (7%), making it a non fit model to implement. As for RFR the training takes some time, but the accuracy is good enough for our goal. XGBR is very slow compared to other approaches, and the accuracy was too much high (99%) resulting a possible over-fitting, but after executing features extraction that had positive results on other approaches the score gets low to (86%).
Figures (Fig. 6.2) and (Fig. 6.3) show the learning curves for both approaches.

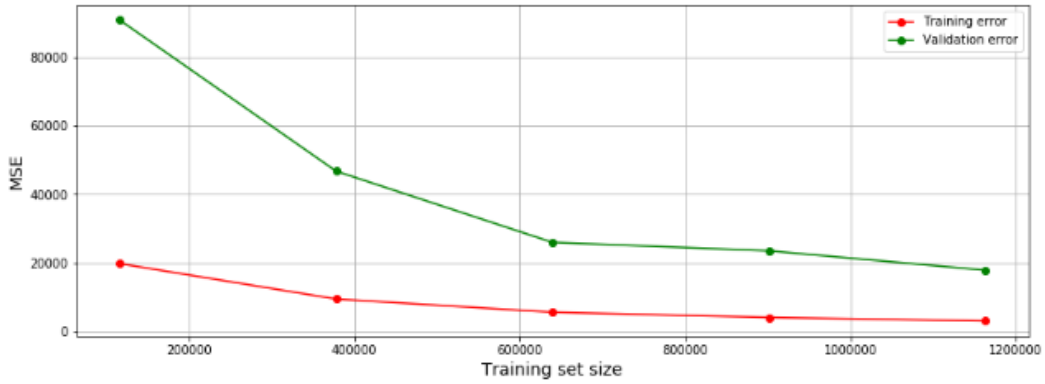


Figure 6.2: Learning curves of Random Forest Regressor

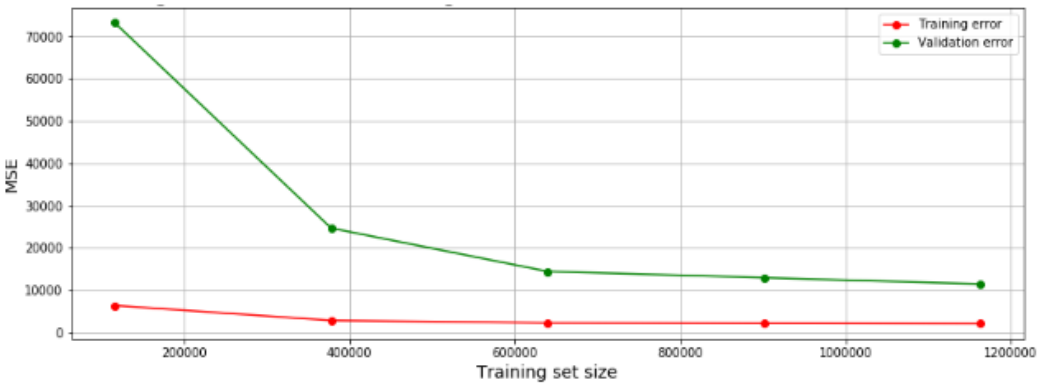


Figure 6.3: Learning curves of XGBoost Regressor

6.6 Conclusion

When we analyse ride fare on XGBoost approach, we can see that the error estimated at 14% which is a 14dzd loss of the driver or 14dzd loss of the passenger when the fare is at minimum of 200dzd, and a 70dzd loss when the price is at maximum of 1000dzd.

General Conclusion

In our final year project, we have dealt with the issues related to the pricing of a ride in a Driver On Demand platform by implementing our Ouigo Pricing system. Surge pricing and price prediction take a big role by adding new features that improve both the company's outcome and client's satisfaction.

The purpose of surge pricing is to augment the price in a certain zone when drivers availability is low, and ride requests are high, and as a goal, controlling the displacement of drivers, and enriching empty places. For price prediction, the purpose is having an exact estimation without the need of Google Maps API, which is an economy on the company's investment.

This project allows full control of the pricing system, and providing factors that help the marketing team to decide on their strategy and their configuration.

Bibliography

[1] Dr. Michael Schwind (auth.). *Dynamic Pricing and Automated Resource Allocation for Complex Information Services: Reinforcement Learning and Combinatorial Auctions*. Lecture Notes in Economics and Mathematical Systems 589. Springer-Verlag Berlin Heidelberg, 1 edition, 2007.

[2] Roy T Fielding and Richard N Taylor. *Architectural styles and the design of network-based software architectures*, volume 7. University of California, Irvine Irvine, 2000.

[3] Kyungmin Brad Lee, Marcus Bellamy, Nitin Joglekar, Shan Jiang, and Christo Wilson. Surge pricing on a service platform under spatial spillovers: evidence from uber. *Available at SSRN 3261811*, 2018.

[4] Sam Newman. *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc.", 2015.