esprit

Se former autrement

HONORIS UNITED UNIVERSITIES

2021 / 2022

SPECIALITY:

Software Engineering

End of studies internship report WSO2 Project

Realized by: Mouadh Lafi

Framed by: ESPRIT supervisor: Sonia Mesbah

Company supervisors: Ali Ghaffari, Mohamed Kalia, Yahia Abou El Hassan

VALUE

EUR-ACE® Délivrée par la Commission des Titres d'Ingénieur

CONFÉRENCE DES GRANDES ÉCOLES

cdio

AUF Agence Universitaire de la Francophonie

IFGU

ESPRIT SCHOOL OF ENGINEERING

www.esprit.tn - E-mail : contact@esprit.tn

Siége Social : 18 rue de l'Usine - Charguia II - 2035 - Tél. : +216 71 941 541 - Fax. : +216 71 941 889

Annexe : 1-2 rue André Ampère - 2083 - Pôle Technologique - El Ghazala - Tél +216 70 250 000 - Fax +216 70 685454

Dedications

From the depth of my heart, I dedicate this work to all those who are dear to me

To the Memory of my dear grand father Bechir

To the Memory of my dear grand father Omar

I cannot express my deep sorrow in your absence. I wish you could have been with me today. May this work be a prayer for the rest of your souls.

To my dear mother Laila

for her love and support, and for being the reason I am here today.

To my beloved father Moncef

No dedication can express the love, respect, dedication and devotion I have always had for you. Despite the distance, you have always been at my side to encourage and support me throughout my studies.

To my dear brothers and sisters: Manar, Mayssem and Mohammed

Who have always believed in me and supported me.

And of course, to all my friends, my source of joy and strive, for the exceptional moments and unlimited care and support.

Finally, to everyone who contributed in making this project a reality.

With love,

Mouadh.

Aknowlegments

At the end of this work, I would like to thank all those who, without their invaluable help, this project would never have been completed. My special thanks go to:

- **Mr. Elyes Ben Rayana** , General Manager of Value Digital Services, for giving me the honour of working in his team.
- My supervisor at the company **Mrs. Hela Bellouma** Delivery manager, for investing a great deal of her time and effort in my internship project. I thank **Mr. Yahia Abou Alhassen** my technical supervisor for his valuable advices.
- My academic supervisor **Mrs. Sonia Mesbah** for the quality of her teaching, encouragement and advice, which have been very beneficial to us.
- All the teachers who have participated in my scientific evolution during the past years of education at the Private Engineering And Technology School.
- To all those who have contributed to the development of this work. Unfortunately, it is not possible to mention them all here, I would like to express my deepest thanks to them.
- My last word is to all members of the jury for the honour of participating in the review of my work.

Contents

List of Figures vii

List of Tables x

1 The context of the project **1**

1.1 The host company: Value Digital Services 2

1.1.1 Overview 2

1.1.2 Value Team Structure 2

1.2 Project presentation 3

1.2.1 Problem statement 3

1.2.2 Motivations 3

1.2.3 Study of the existing 4

1.2.3.1 Automobile.tn 4

1.2.3.2 Auto Plus 4

1.2.4 Criticism of the existing 4

1.2.4.1 Proposed solution 5

1.3 Adopted methodology 5

1.3.1 SCRUM artifacts 6

1.3.2 SCRUM roles 7

1.4 Chronogram of tasks 7

2 Sprint 0 : Before starting **9**

2.1 Specification of requirements 10

2.1.1 Identification of the actors 10

2.1.2 Functional requirements 10

2.1.3 Non functional requirements 11

2.2 Technical working environment 12

2.2.1 Development tools 12

2.2.1.1 STAR UML . . . 12
2.2.1.2 Visual Studio Code . . . 12
2.2.1.3 IntelliJ IDEA . . . 12
2.2.1.4 Postman . . . 12
2.2.2 Programming languages et Frameworks . . . 12
2.2.2.1 WSO2 API Manager . . . 12
2.2.2.2 Angular . . . 13
2.2.2.3 Spring Boot . . . 13
2.3 Global Use Case Diagram . . . 14
2.4 Global design . . . 14
2.4.1 Application architecture . . . 14
2.4.1.1 Logical architecture . . . 15
2.4.1.2 Physical architecture . . . 15
2.5 Project planning . . . 16
2.6 Product Backlog . . . 17
2.7 Analysis class diagram . . . 17
2.8 Conclusion . . . 18

3 Sprint 1 : WSO2 API Manager integration 19

3.1 WSO2 goals . . . 20
3.2 WSO2 features . . . 20
3.2.1 API Publisher . . . 20
3.2.2 Developer portal . . . 21
3.2.3 API Gateway . . . 22
3.2.4 Key Manager . . . 22
3.2.5 Traffic Manager . . . 23
3.2.6 Analytics . . . 23
3.3 Exposing APIs to WSO2 . . . 24
3.4 Conclusion . . . 26

4 Sprint 2: Authentication and role management 27

4.1 Sprint Backlog . . . 28
4.2 Key concepts . . . 29
4.2.1 Keycloak . . . 30
4.2.2 Acess token . . . 30
4.2.3 Keycloak session . . . 30

4.2.4 Realm . . . 30
4.3 Requirements specification . . . 31
4.3.1 Role management . . . 31
4.3.2 User management . . . 32
4.4 Implementation . . . 33
4.5 Conclusion . . . 34

5 Sprint 3: Brands and Models Management 35

5.1 Sprint Backlog . . . 36
5.2 Brands Management . . . 38
5.2.1 Functional specification of the brands management . . . 38
5.2.1.1 Use case diagram of the brands management . . . 39
5.2.1.2 Class diagram of the brands management . . . 39
5.2.1.3 Sequence diagram . . . 40
5.3 Models management . . . 42
5.3.1 Functional specification of the models management . . . 42
5.3.1.1 Use case diagram of the models management . . . 43
5.3.1.2 Class diagram of the brand management . . . 43
5.3.1.3 Sequence diagram . . . 44
5.4 Implementation . . . 46
5.4.1 Implementation of the brand management . . . 46
5.4.2 Implementation of the models management . . . 49

6 Sprint 4: Credit Management and Dashboard 53

6.1 Sprint Backlog . . . 54
6.2 Credit management . . . 56
6.2.1 Use case diagram . . . 56
6.2.2 sequence diagram . . . 58
6.3 Dashboard . . . 60
6.3.1 Use Case diagram . . . 60
6.3.2 Sequence diagram . . . 61
6.4 Implementation . . . 61
6.4.1 Credit management . . . 61
6.4.2 Dashboard . . . 67

7 Sprint 5: Deployment 68

7.1 Deployment tools . . . 69

7.1.1 Azure cloud . . . 69
7.1.1.1 Virtual machines . . . 69
7.1.1.2 Kubernetes . . . 69
7.1.2 Docker . . . 69
7.2 Deployment process . . . 70
7.2.1 WSO2 API Manager deployment . . . 70
7.2.1.1 API-M cluster . . . 71
7.2.1.2 Integration clusters . . . 71
7.2.2 Microservices deployment . . . 71
7.3 Implementation . . . 71
7.3.1 WOS2 API Manager . . . 71
7.3.2 microservices deployment . . . 74
Conclusion and perspectives . . . 78

References **79**

List of Figures

1.1 Logo Value 2
1.2 SCRUM Life-cycle [7] 6
1.3 Chronogram of tasks 7

2.1 General use case diagram 14
2.2 Logical architecture 15
2.3 Physical architecture 15
2.4 class diagram 17

3.1 API Publisher architecture 21
3.2 Developer portal architecture 21
3.3 Key Manager architecture 22
3.4 Traffic Manager architecture 23
3.5 Analytics architecture 24
3.6 Swagger Importation 25
3.7 Swagger Importation step 2 25
3.8 Developer Portal Interface 26

4.1 Sprint 2 Backlog 29
4.2 Authentication scenario with keycloak 31
4.3 Role management in Keycloak 32
4.4 User management in Keycloak 33
4.5 Authentication interface 33

5.1 Sprint 3 Backlog 38
5.2 Sprint Use case Diagram 39
5.3 Sprint Brand Class Diagram 40

5.4 Add brand Sequence diagram . . . 41
5.5 Modify brand Sequence diagram . . . 42
5.6 Model Use case Diagram . . . 43
5.7 Model Class Diagram . . . 44
5.8 Add model Sequence diagram . . . 45
5.9 Modify model Sequence diagram . . . 46
5.10 Brand Interface . . . 47
5.11 Brand Adding Interface . . . 47
5.12 Search Interface . . . 48
5.13 Brand Modifying Interface . . . 49
5.14 Models Interface . . . 49
5.15 Models Adding Interface . . . 50
5.16 Models search Interface . . . 51
5.17 Model Modifying Interface . . . 51

6.1 Sprint 4 Backlog . . . 56
6.2 Sprint Use case Diagram . . . 57
6.3 Sprint Use case Diagram 2 . . . 58
6.4 Sequence diagram 1 . . . 59
6.5 Camunda workflow . . . 60
6.6 Dashboard Use case Diagram . . . 61
6.7 Dashboard Sequence Diagram . . . 61
6.8 Unprocessed credits list . . . 62
6.9 Processed credits list . . . 62
6.10 Note Credit step 2 . . . 63
6.11 Note Credit step 2 . . . 64
6.12 Note Credit step 3 . . . 64
6.13 Accepted credit . . . 65
6.14 Credit document list . . . 66
6.15 Credit document list . . . 66
6.16 Dashboard . . . 67

7.1 Deployment pattern . . . 70
7.2 Helm installing . . . 72
7.3 Ingress deploying . . . 73
7.4 wso2 deployment finished . . . 73

7.5 Creation of Spring Boot jar 74
7.6 Docker Image 74
7.7 Docker Hub 75
7.8 Deployment File 76
7.9 service File 76
7.10 Ingress File 77

List of Tables

2.1 Project planning by Sprint 16
2.2 Product Backlog 17

6.1 admin levels 59

Chapter 1

The context of the project

Plan

1.1 The host company: Value Digital Services 2

1.1.1 Overview 2

1.1.2 Value Team Structure 2

1.2 Project presentation 3

1.2.1 Problem statement 3

1.2.2 Motivations 3

1.2.3 Study of the existing 4

1.2.4 Criticism of the existing 4

1.3 Adopted methodology 5

1.3.1 SCRUM artifacts 6

1.3.2 SCRUM roles 7

1.4 Chronogram of tasks 7

Introduction

The general context of our project will be defined in this chapter. In the first section, we'll introduce the host company, its numerous activities, and the department where we interned. The following section will offer a broad overview of the project, including the problem statement and analysis of the existing and proposed solutions, followed by the project's work methodology.

1.1 The host company: Value Digital Services

This section is an introduction to **Value Digital Services** which proposed and accommodated this project.

1.1.1 Overview

Value Digital Services is a young service firm that helps customers establish and manage their strategy, as well as build and implement their digital and data road maps. It has deployed many factories to create value in the fields of digital expertise, data services, artificial intelligence and strategy. Day after day, the Value family works to develop and unleash the potential of its employees. The goal is to make an impression, reinforce and display the trust that their partners have in them. The company was established in 2018 and began operations in August of 2019. Value digital services is a well-known Tunisian firm with more than 60 workers.

Figure 1.1: Logo Value

1.1.2 Value Team Structure

Mr. AHMED LASRAM and Mr. ELYES BEN RAYANA, the company's founders, are in charge of all aspects of Value's operations. They are accompanied by teams that are organized into functional and role groups:

The digital team, headed by Ms. Hela Bellouma, is involved in the full process of developing digital solutions, from expressing needs to implementing and maintaining them.

The data scientists' group: Value focuses on large-scale data and Big Data initiatives, beginning with the development of data strategies that include all economic actors.

Artificial intelligence: Value has created cutting-edge artificial intelligence skills, allowing new predictive models to be implemented.

In a continuously changing environment, the strategic consulting team was created to support Value's customers and guide them in their strategic decisions.

1.2 Project presentation

In this section, we will present our project. Our difficulty, as well as the subject's description and desired missions, will be revealed first. Then we'll go over the work process for the project in depth.

1.2.1 Problem statement

When we start developing APIs for our applications which are for both internal and external use, the information about these have to be shared within the developer and the user community. Along with the information about the API, the need to control, monitor access to the exposed APIs is required. As the number of APIs increase, it becomes difficult to exchange, maintain the APIs, and there is a need to maintain these systematically through processes, tools specifically designed to manage APIs.

- the visibility of APIs is not centralized ,thus increase security vulnerabilities, the number of repetitive APIs and increase the gaps for developers to address.
- the inability to analyze APIs to understand the trend, most commonly used APIs.
- The problem of managing several APIs from various projects while maintaining versioning and compatibility for previous versions.
- The inability to monetize APIs, share revenue with partners and track billing in real time.

1.2.2 Motivations

To address the issues raised above, we emphasize the importance of API management and the benefits it provides, which include decentralization, transparency, security, and API integrity. In this context, our project's aim is to design and develop a new Car dealer Web/Mobile application using the old bank credit management application that will be managed with the WSO2 API Manager.

1.2.3 Study of the existing

The Car dealership market continues to grow, by the increasing number of tools and applications available on the market. This market is measured in particular by the growth and diversity of business applications for car professionals.In this section, we will attempt to conduct a comparative analysis of the available solutions ,this analysis will allow us to:

- Use similar experiences that have already been implemented to inspire our project.
- Locate our system in relation to existing systems.
- Define what our application should accomplish..

There are several car dealership application options available today :

1.2.3.1 Automobile.tn

Automobile.tn is a Tunisian portal specialized in the automotive sector in Tunisia. Through its various sections, Automobile.tn allows Internet users to learn about the prices and technical characteristics of new vehicles marketed in Tunisia, by the various official dealers.[8].

1.2.3.2 Auto Plus

Auto Plus is a web portal that includes all brands, models and versions of new cars being marketed on the Tunisian market.it provides its visitors with several tools to help them make a decision in their choice of new cars and has integrate a leasing simulator that calculates the monthly amount needed for leasing Tunisia.

1.2.4 Criticism of the existing

While the above solutions save time for both vehicle dealers and their customers, they are lacking in many features:

- **Automobile.tn :** Despite the wide range of vehicles available on this website, it is merely a portal to display the vehicles' technical specifications and costs, with no capacity to purchase the vehicles, obtain loan applications, or even perform a simple simulation to determine loan eligibility.
- **Auto Plus :** This application provides a limited set of features that do not provide the client with a clear picture of the credit application procedure. Indeed, simply stating the monthly payment amount is sufficient. This does not provide the customer with a clear image of the application process or whether their request has been accepted or not.

1.2.4.1 Proposed solution

The goal of this project is to create a 2 channel Car dealership web and mobile application.Indeed the first application will provide the following features:

- allows customers to research the prices and technical specifications of vehicles sold in Tunisia by numerous official dealers.
- Apply for a credit and the system will automatically assess his eligibility and hence whether he will benefit from the credit.
- Accept or reject the simulation as appropriate. If the credit is approved, this application will allow you to file all required documents, consult, and sign the contract.

The second application is back office application , designed for Cars agents, allows them to:

- Add and modify available cars and brands.
- Car request analysis.
- Examine any current or accepted credit applications, as well as any document that have been filed.

The two applications will be managed by the WSO2 API Manager, which will ensure access control, secure application access, and centralize visibility that allow to see all the API connections in one place.

1.3 Adopted methodology

Managing a project entails more than just designing and writing code. The selection of an appropriate technique for a project is a critical stage that determines its success. It guarantees a particular degree of quality and efficiency, which are critical requirements for our project. We chose agile as a methodology and SCRUM as an agile implementation to manage the project development life cycle. This decision is based on the following requirements:

- The project specifications will change over time.
- Constant feedback is necessary..
- The requirement for regular software delivery.

SCRUM is a framework used by teams to manage their work. SCRUM implements the principles of Agile as a concrete set of artifacts, practices, and roles[7].

1.3.1 SCRUM artifacts

The figure 1.2 presents the different artifacts of SCRUM.

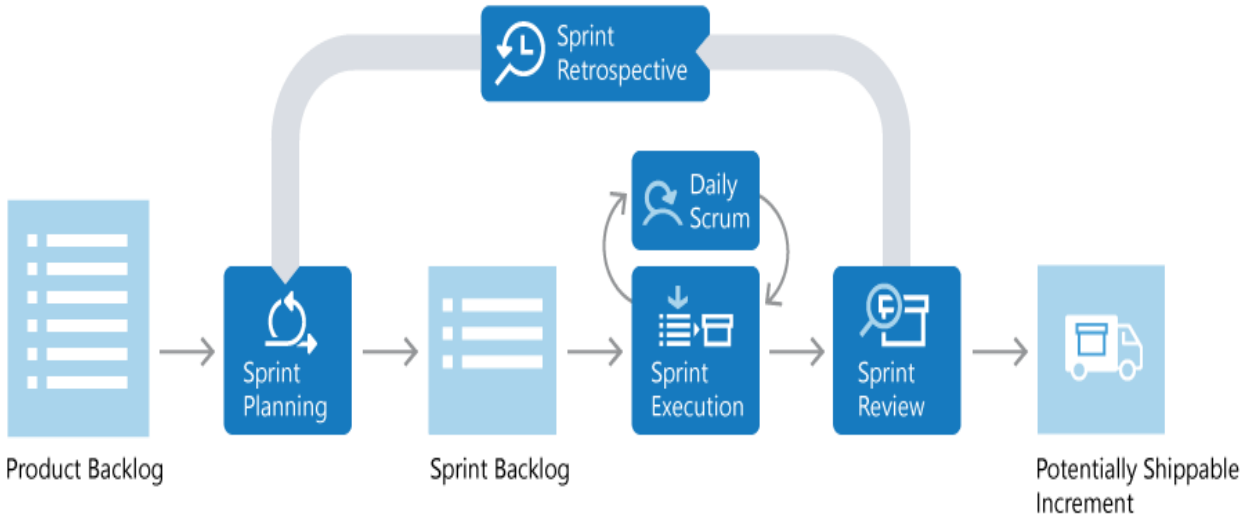


Figure 1.2: SCRUM Life-cycle [7]

There are a specific, unchanging set of steps in the SCRUM flow. They include:

- **Product backlog:** The Product Backlog is a prioritized list of value the team can deliver. The Product Owner owns the backlog and adds, changes, and reprioritizes as needed. The items at the top of the backlog should always be ready for the team to execute on[7].
- **Sprint Planning and Sprint Backlog:** In Sprint Planning, the team chooses the backlog items they will work on in the upcoming sprint. The team chooses backlog items based on priority and what they believe they can complete in the sprint. The Sprint Backlog is the list of items the team plans to deliver in the sprint. Often, each item on the Sprint Backlog is broken down into tasks. Once all members agree the Sprint Backlog is achievable, the Sprint starts[7].
- **Sprint Execution and Daily SCRUM:** The Daily Once the Sprint starts, the team executes on the Sprint Backlog. SCRUM does not specify how the team should execute. That is left for the team to decide.
  SCRUM defines a practice called a Daily SCRUM, often called the Daily Stand-up. The Daily SCRUM is daily meeting limited to 15 minutes. Team members often stand during the meeting, to ensure it stays brief. Each team member briefly reports their progress since yesterday, the plans for today, and anything impeding their progress[7].
- **Sprint review meeting:** The team shows stakeholders what they've accomplished at the end of each sprint. They demonstrate the software and its utility.

- **Sprint Retrospective:** The team takes time to reflect on what went well and which areas need improvement. The outcome of the retrospective are actions for next sprint[7].

1.3.2 SCRUM roles

- **Product owner :**In the majority of projects, the product owner is the representative of clients and users. He will define and prioritize the list of product functionalities and choose the date and content of each sprint.
  In our case, it is Mrs. Maha Ben chaaben.
- **SCRUM Master :**Often considered the coach for the team, the SCRUM Master organize meetings, and working with the Product Owner to ensure the product backlog is ready for the next sprint. In our case, it's Mr. Yahia Aboulhassan.
- **SCRUM team :**The SCRUM Team is comprised of 3 members: Mouadh Lafi, Aymen Ghorbel, Firas Soltani.

1.4 Chronogram of tasks

According to the internship term, we will offer a chronogram of the many tasks of our project. This timetable is represented by the figure, which is a Gantt chart. 1.3.

Figure 1.3: Chronogram of tasks

Conclusion

We provide the host company and its sphere of activity throughout this chapter, as well as

the problem statement, analysis of the existing and suggested solutions, and a summary of the project.
Following that, we reviewed the present problem before recommending a solution. The start-up sprint will be discussed in the following chapter.

Chapter 2

Sprint 0 : Before starting

Plan

2.1 Specification of requirements 10

2.1.1 Identification of the actors 10

2.1.2 Functional requirements 10

2.1.3 Non functional requirements 11

2.2 Technical working environment 12

2.2.1 Development tools 12

2.2.2 Programming languages et Frameworks 12

2.3 Global Use Case Diagram 14

2.4 Global design 14

2.4.1 Application architecture 14

2.5 Project planning 16

2.6 Product Backlog 17

2.7 Analysis class diagram 17

2.8 Conclusion 18

Introduction

Our start-up sprint is covered in this chapter. We're most interested in the requirements analysis first and then the design, that's where we explain our application's architecture. We then go on to work planning, where the product backlog and sprint division are shown. Finally, we present the work environment, which comprises the software as well as the technologies used.

2.1 Specification of requirements

2.1.1 Identification of the actors

We shall identify the major stakeholders in our system in this section:

- **Clients:** The initial web application will be available to clients who want to finance their car purchases.
- **Agents:** The second web application is for car dealer agents who are in charge of the vehicle stock and the credit application procedure at various stages.

2.1.2 Functional requirements

All users' needs must be met by the functional requirements. The client web application must allow the customer to do the following after valid authentication:

- Authenticate and register.
- Consult car brands and models.
- Consult the user's profile and modify it.
- Simulate the financing of a car purchase.
- Opt for a credit on a car purchase.
- Submit documents in case of a credit.

- Consult the state of credit demands.

The Admin web application must allow the agent to do the following after valid authentication:

- Manage car models and brands.
- Analysis of car requests.
- Consultation of car requests.
- Consultation of credit applications.
- Validate credits.
- Check the customer's papers.
- Send contract.

2.1.3 Non functional requirements

We must ensure that our solution aims for a specific level of satisfaction, which can be discovered in non-functional criteria, after identifying the functional requirements. The following are a few of them:

- **User-friendliness :** To ensure that users are correctly focused throughout their interactions, the application must have a clear and user-friendly interface.
- **Performance :** The application must be able to respond quickly and effectively to the diverse demands of users.
- **Security :** The client's credit request data must be kept private by the application.
- **Maintainability :** To allow both error repair and detection the application's code must be well-commented.

2.2 Technical working environment

2.2.1 Development tools

2.2.1.1 STAR UML

In a development project, design is a crucial step. As a result, we employed the StarUml modeling software. This software is simple to operate. This software can model most of the diagrams defined in the UML 2.0 standard.

2.2.1.2 Visual Studio Code

Microsoft's Visual Studio Code is an extendable code editor. Debugging support, syntax highlighting, smart code completion, snippets, code refactoring, and integrated Git are among the features.

2.2.1.3 IntelliJ IDEA

IntelliJ IDEA is an integrated development environment written in Java for developing computer software written in Java, Kotlin, Groovy, and other JAR based languages.

2.2.1.4 Postman

a platform for API developers to collaborate. Postman's features make cooperation and the API development process easier, allowing you to create better APIs faster.

2.2.2 Programming languages et Frameworks

2.2.2.1 WSO2 API Manager

WSO2 API Manager is a complete platform for building, integrating, and exposing your digital services as managed APIs in the cloud, on-premise, and hybrid architectures to drive your digital transformation strategy.

It allows API developers to design, publish, and manage the lifecycle of APIs and API product managers to create API products from one or more APIs. [1]

2.2.2.2 Angular

Angular is an HTML and TypeScript-based platform and framework for creating single-page client applications. TypeScript is used to write Angular. As a set of TypeScript libraries that you load into your apps, it implements core and optional functionality.

2.2.2.3 Spring Boot

Spring Boot is a Java-based open source framework for developing microservices. Pivotal Team created it, and it's used to create stand-alone and production-ready spring apps.

2.3 Global Use Case Diagram

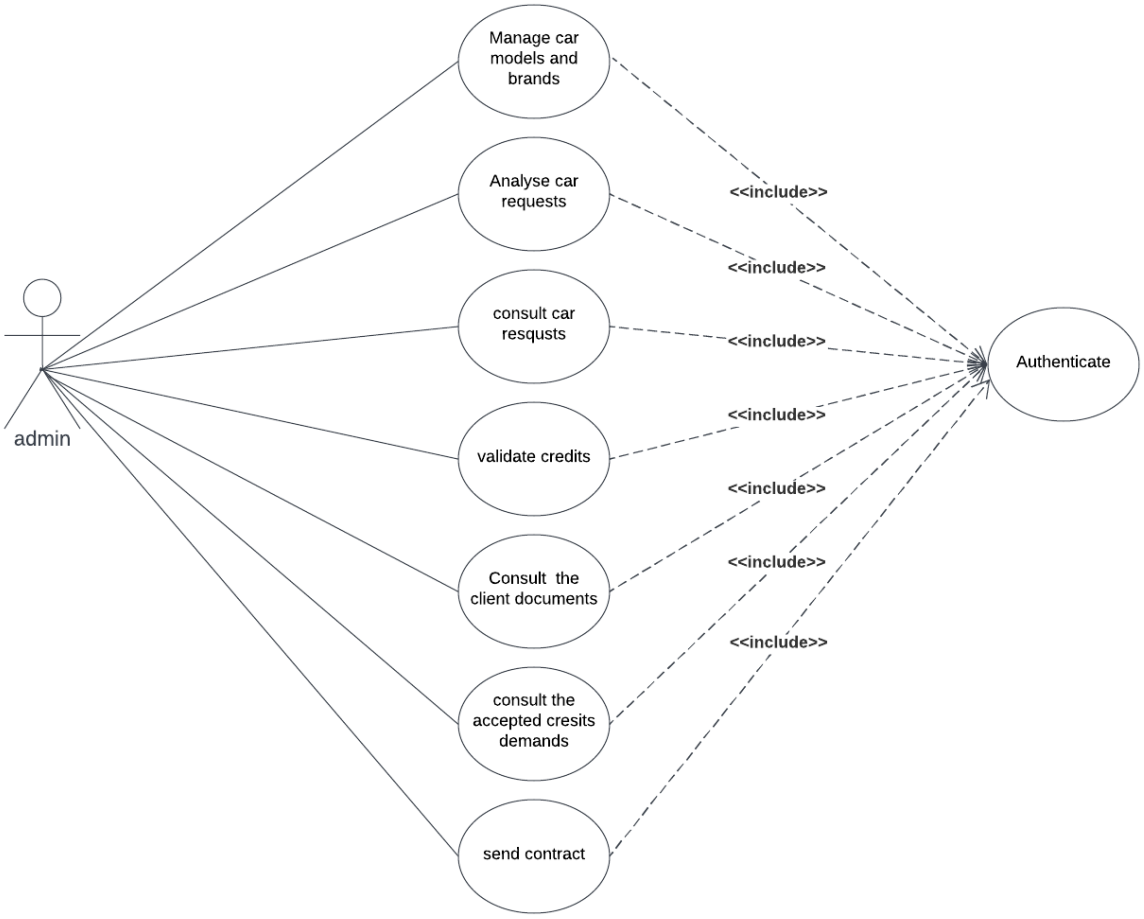


Figure 2.1: General use case diagram

2.4 Global design

To ensure optimal operation and performance, it is vital to adopt the suitable architecture while creating any program.

2.4.1 Application architecture

The physical and logical structure of an application are addressed by its architecture. This involves the system being divided into logical layers of components based on the type of processing they perform, as well as several physically separated third parties. The architecture of our system, including the physical and logical architecture, is presented in this part.

2.4.1.1 Logical architecture

The logical architecture of our program is depicted in Figure 2.2, which describes the software components required for project implementation and displays the links between them.

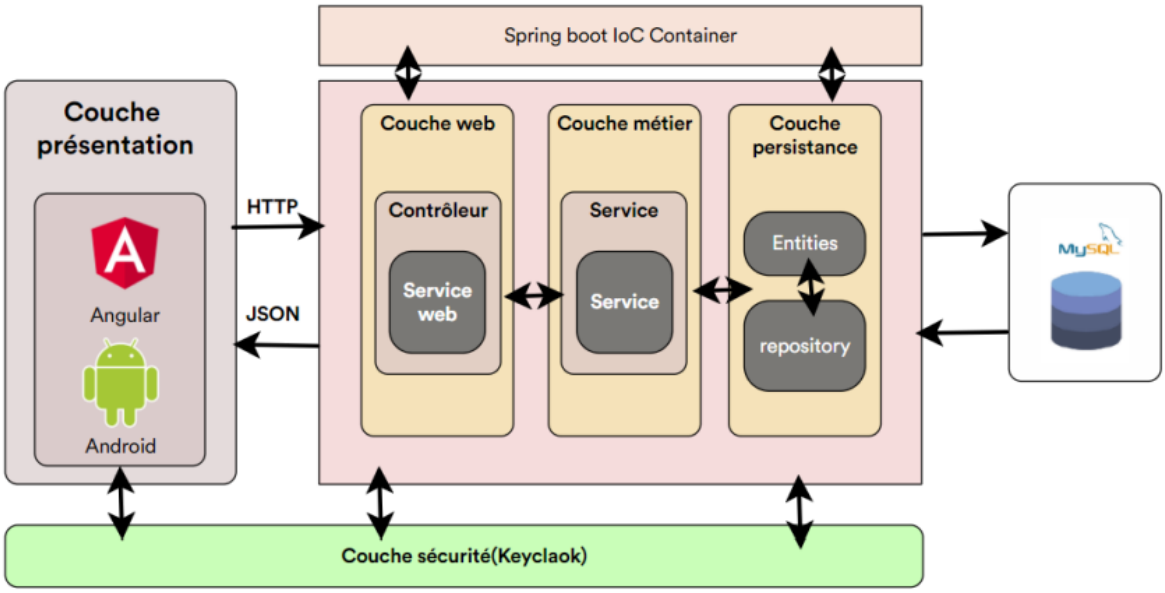


Figure 2.2: Logical architecture

2.4.1.2 Physical architecture

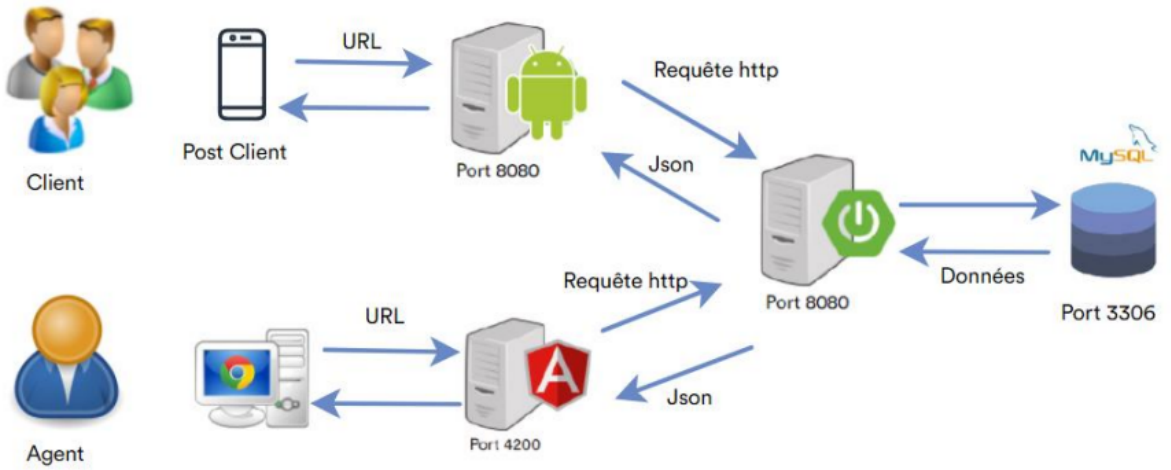


Figure 2.3: Physical architecture

We chose a multi-layer architecture comprised mostly of the following tiers:

- **Presentation tier:** This is the layer that contains the application's web page, which is accessed through the user's web browser. This tier is typically constructed using web technologies like HTML5, JavaScript, CSS, or other popular web development frameworks like ANGULAR, and communicates with other layers via API calls. We'll be using a Node Js server with the Angular framework for our project.
- **Application tier:** It is the intermediary layer that manifests the communication between the two layers, and it is referred to as the application's engine. All processing essential to assure a result is carried out through this layer, which is implemented in PHP, Java, C, or other computer languages. We'll use the Spring framework to construct a Web server for our project.
- **Database tier:** It's the name of the database server. This is where the data is accessed. On this tier, a DBMS (Database Management System) such as MySQL or Microsoft SQL Server is installed, and the server application requests that this server consume a particular quantity of data.

2.5 Project planning

sprint number	Tasks
Sprint 0	Requirements specification and analysis
Sprint 1	Integrate the APIs using WSO2
Sprint 2	Authentication and registration using Keycloak
Sprint 3	Brands and models management
Sprint 4	Credits management and Dashboard
Sprint 5	Deployment

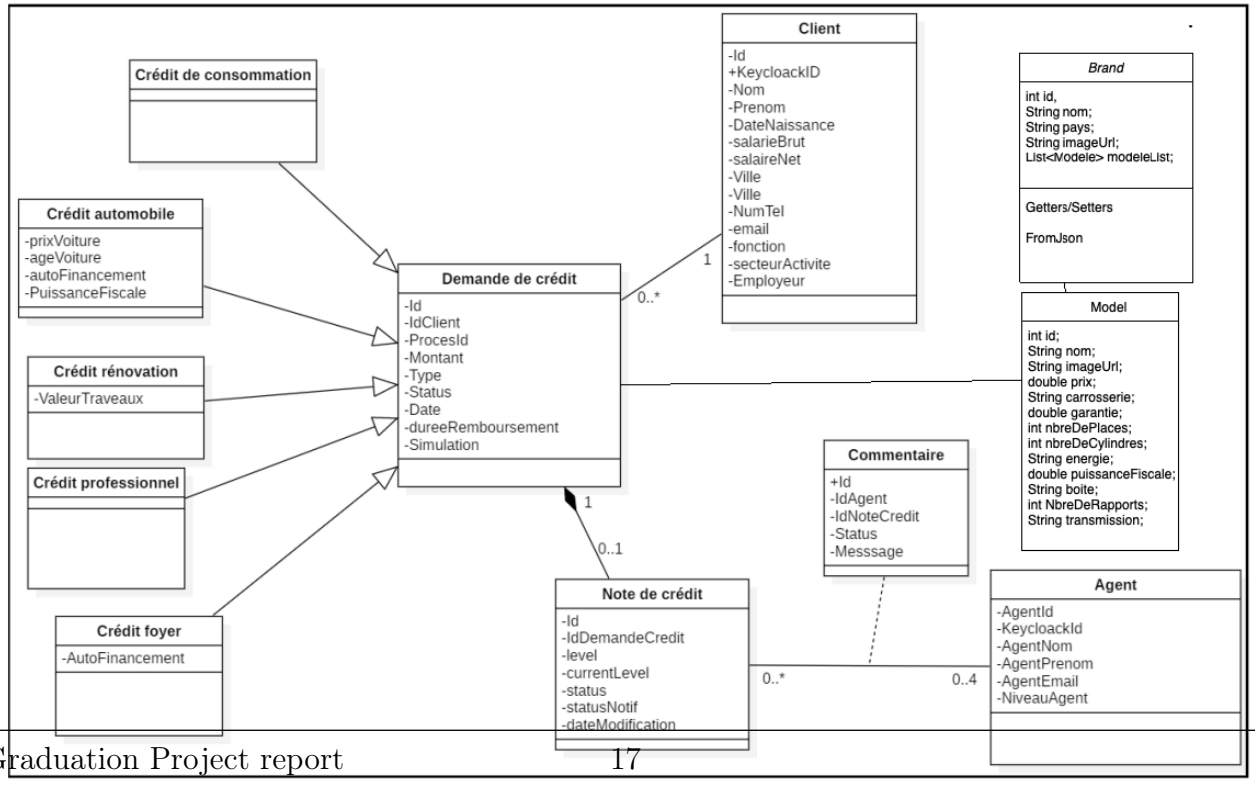
Table 2.1: Project planning by Sprint

ID-F	Features	User Stories	Priority
1	Authentication	As an admin, I want to authenticate myself	80
		As an admin, I want the error cases to be handled during authentication	100
		As an admin, I want to disconnect	80
		As an admin, I want to reset my password	80
2	Brand Management	As an admin, I want to display the list of car brands.	100
		As an admin, I want to add a car brand.	70
		As an admin, I want to change a brands.	70
		As an admin, I want to delete brands.	70
		As an admin, I want to search for a brand	70
3	Models Management	As an admin, I want to display the list of car models.	100
		As an admin, I want to add a car models.	70
		As an admin, I want to change the models.	70
		As an admin, I want to delete models.	70
		As an admin, I want to search for a models	70
4	Credit Management	As an admin, I want to display the list of Credit demands .	100
		As an admin, I want to Consult the eligibility of demands.	80
		As an admin, I want to consult the demands documents .	80
		As an admin, I want send the contract	100
5	Dashboard	As an admin, I want to analyze car requests.	60
		As an admin, I want to analyze the demands of car loans.	60

Table 2.2: Product Backlog

2.6 Product Backlog

2.7 Analysis class diagram



2.8 Conclusion

In this chapter,we discussed the functional requirements specifications of our application, as well as the working environment The project planning, including the product backlog, was then presented, followed by the two types of logical and physical architectures that we would use.

Chapter 3

Sprint 1 : WSO2 API Manager integration

Plan

3.1 WSO2 goals 20

3.2 WSO2 features 20

3.2.1 API Publisher 20

3.2.2 Developer portal 21

3.2.3 API Gateway 22

3.2.4 Key Manager 22

3.2.5 Traffic Manager 23

3.2.6 Analytics 23

3.3 Exposing APIs to WSO2 24

3.4 Conclusion 26

Introduction

We'll go over how WSO2 works, what its benefits are, and how we integrated it with our existing projects in this chapter. We'll also talk about Keycloak authentication and how it works with WSO2 API Manager.

3.1 WSO2 goals

If we return to our challenge, we may deduce a requirement: the desire to group older APIs under a single gateway, making management of those APIs easier while also modernizing those older APIs with new security and access control capabilities.

3.2 WSO2 features

The following are some of the main capabilities of the product.

3.2.1 API Publisher

WSO2 API Manager provides a state-of-the-art web interface called WSO2 API Publisher for API development and management. It is a structured GUI designed for API creators to develop, document, scale and version APIs, while also facilitating more API management-related tasks such as publishing API, monetizing APIs, and promoting. [2]

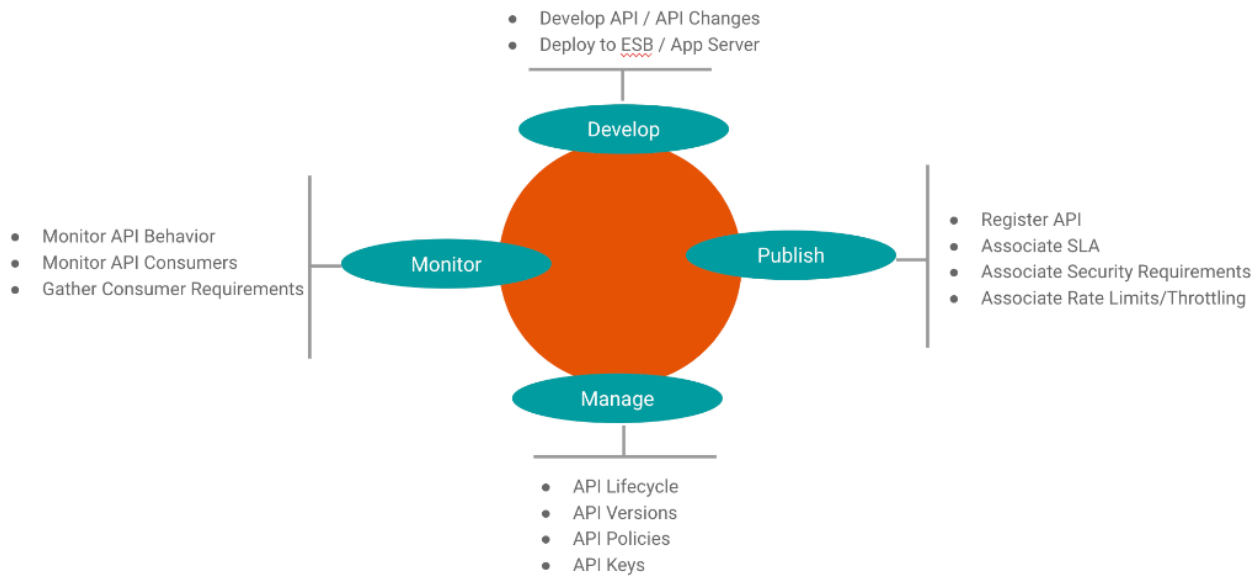


Figure 3.1: API Publisher architecture

3.2.2 Developer portal

The Developer Portal is a state-of-the-art web interface that allows API publishers to host and advertise their APIs while allowing API consumers to self-register, discover, evaluate, subscribe to and consume APIs. [2]

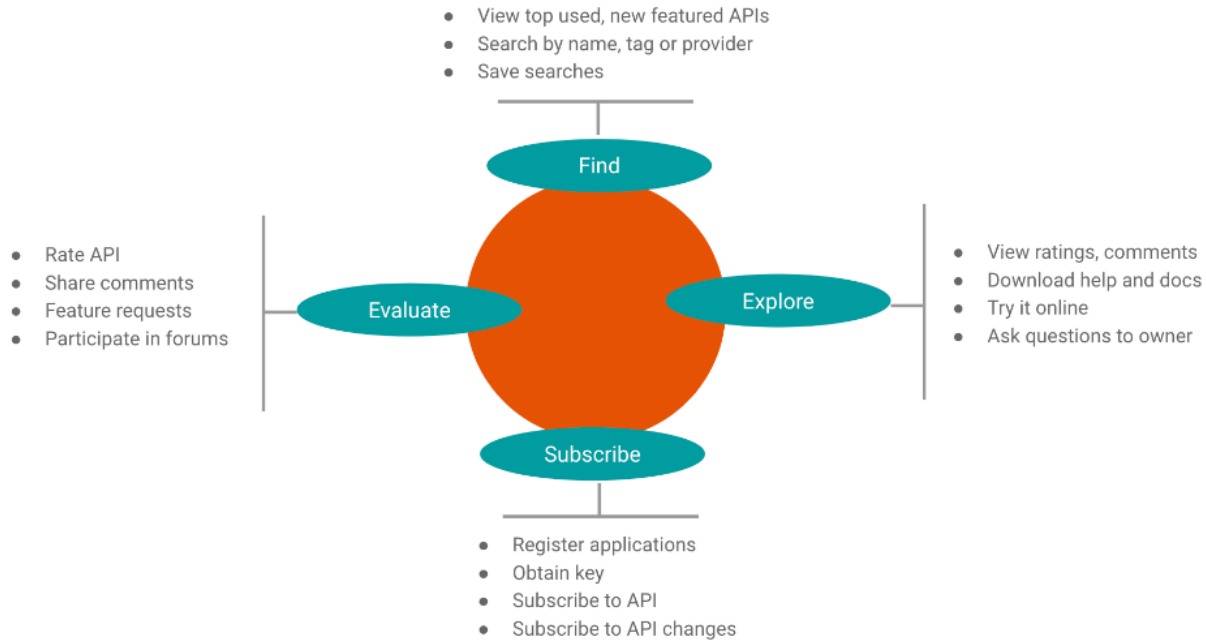


Figure 3.2: Developer portal architecture

3.2.3 API Gateway

The API Gateway is a runtime, backend component (an API proxy) developed using WSO2 EI. API Gateway secures, protects, manages, and scales API calls. It intercepts API requests, applies policies such as throttling and security using handlers, and manages API statistics. [2]

3.2.4 Key Manager

The Key Manager manages all clients, security and access token-related operations. The Gateway connects with the Key Manager to check the validity of access tokens, subscriptions and API invocations. [2]

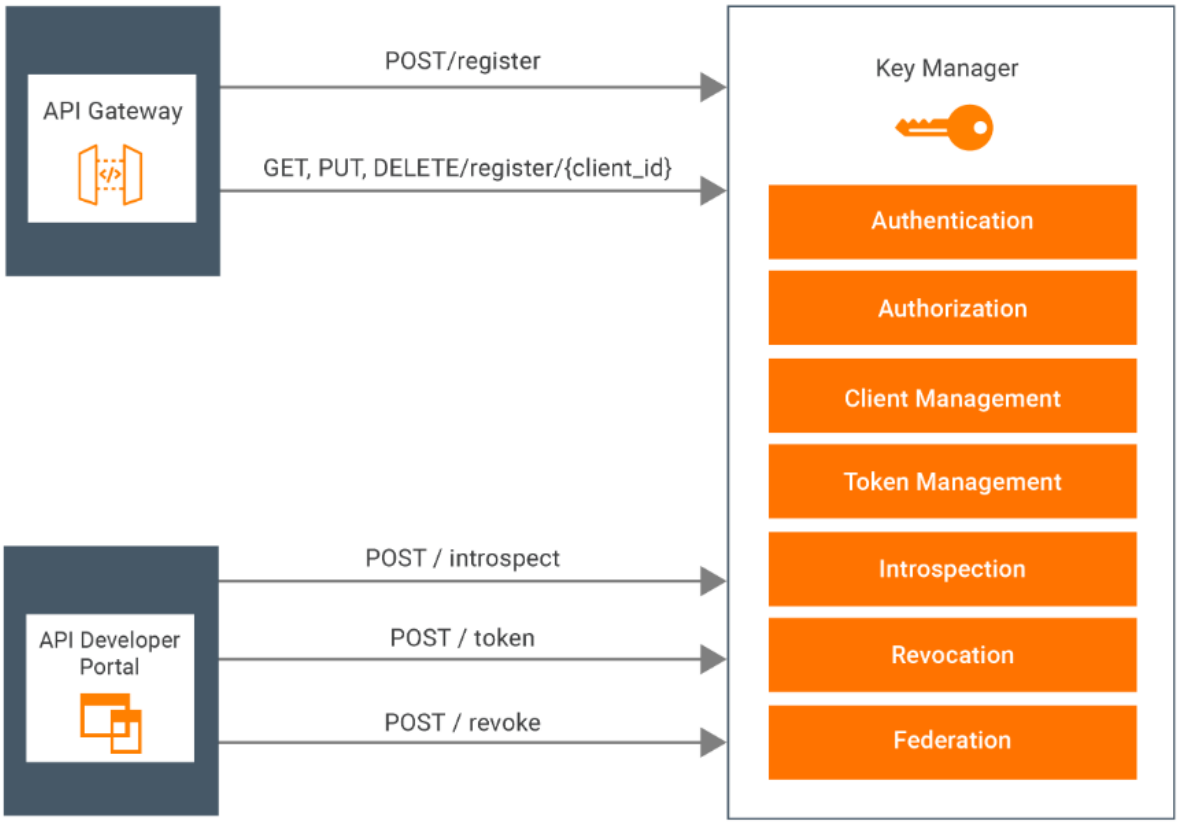


Figure 3.3: Key Manager architecture

3.2.5 Traffic Manager

The Traffic Manager helps users to regulate API traffic, make APIs and applications available to consumers at different service levels, and secure APIs against security attacks. The Traffic Manager features a dynamic throttling engine to process throttling policies in real-time, including rate limiting of API requests. [2]

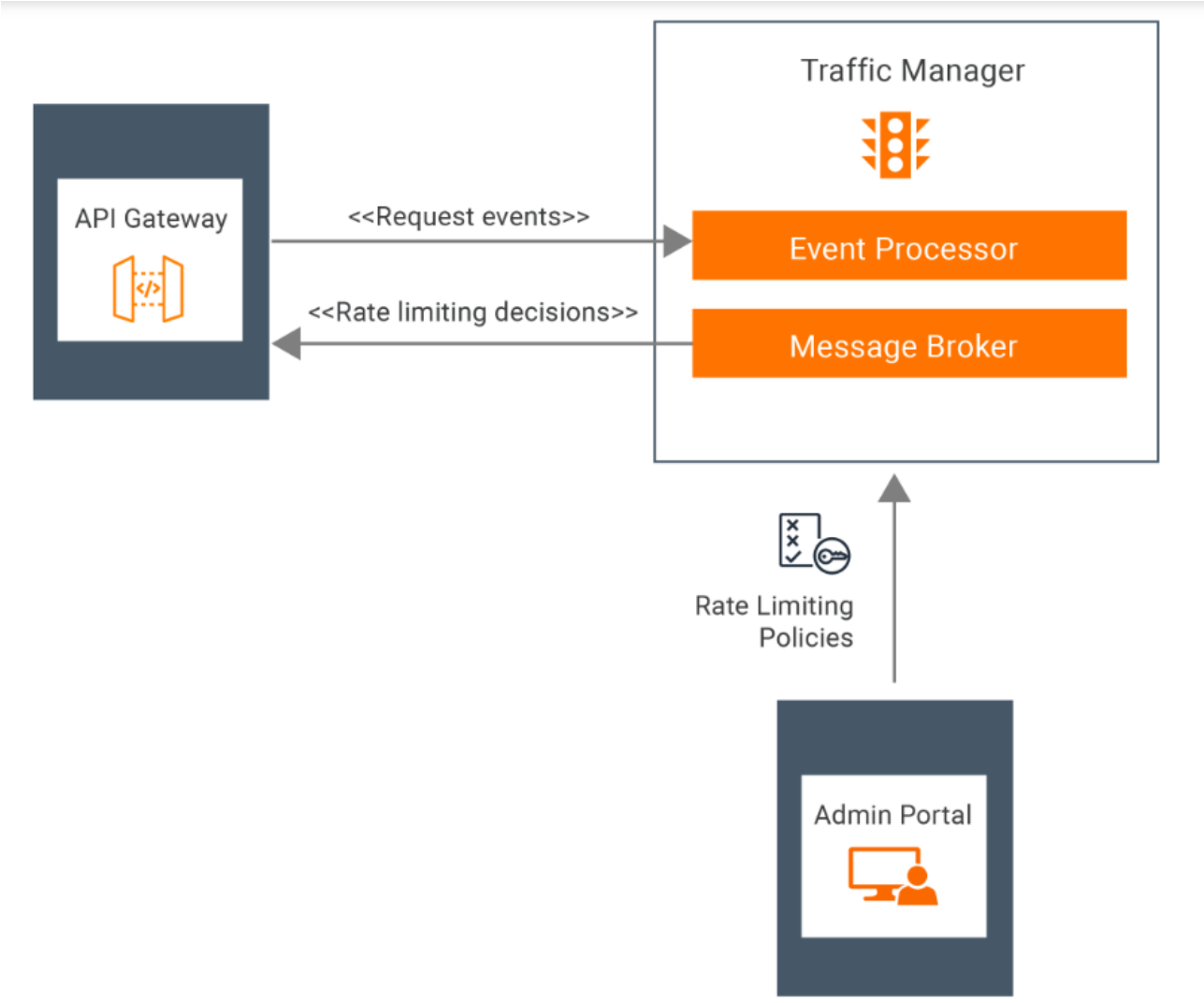


Figure 3.4: Traffic Manager architecture

3.2.6 Analytics

Additionally, monitoring and analytics are provided by the analytics component, WSO2 API Manager Analytics. This component provides a host of statistical graphs and an alerting mechanism on pre-determined events. [2]

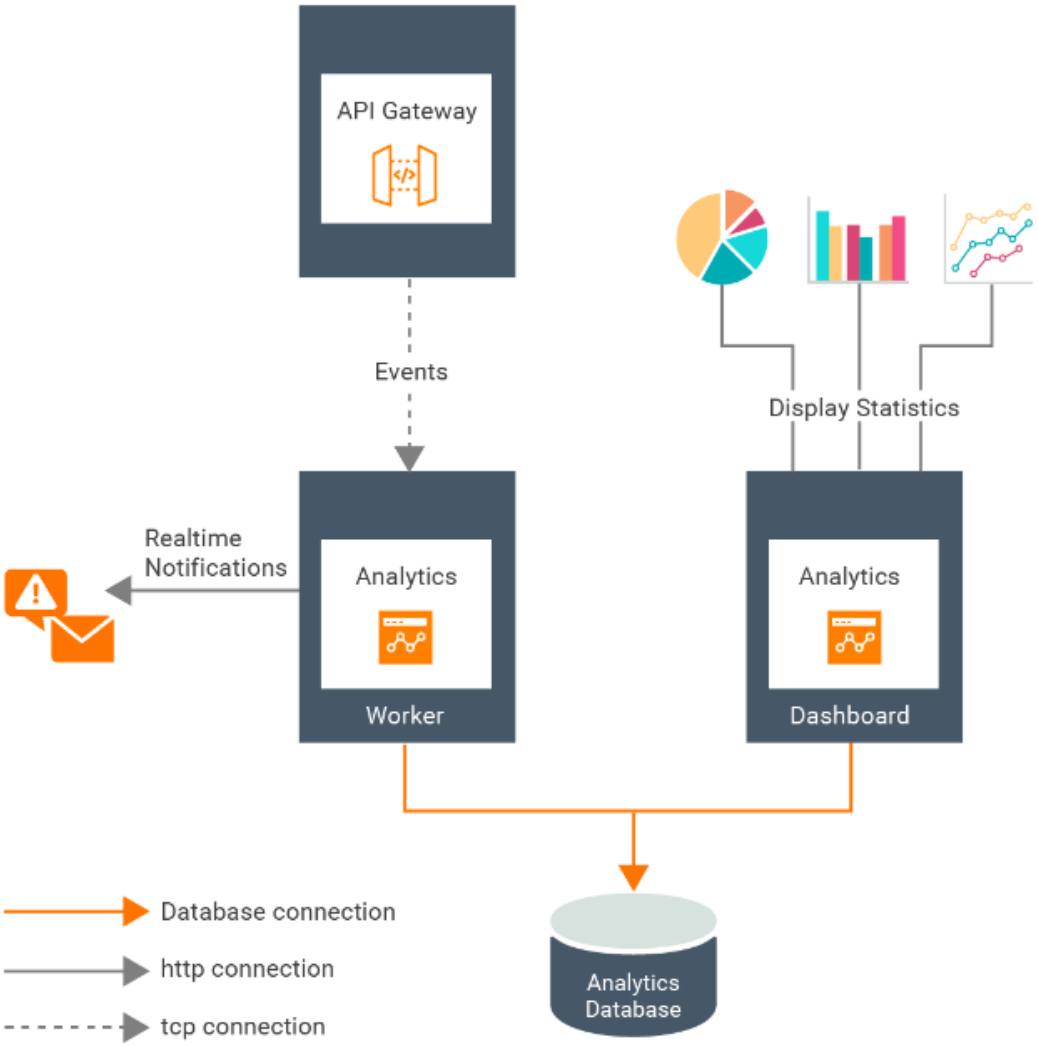


Figure 3.5: Analytics architecture

3.3 Exposing APIs to WSO2

We can import Swagger documentation into WSO2 using the API publisher interface .

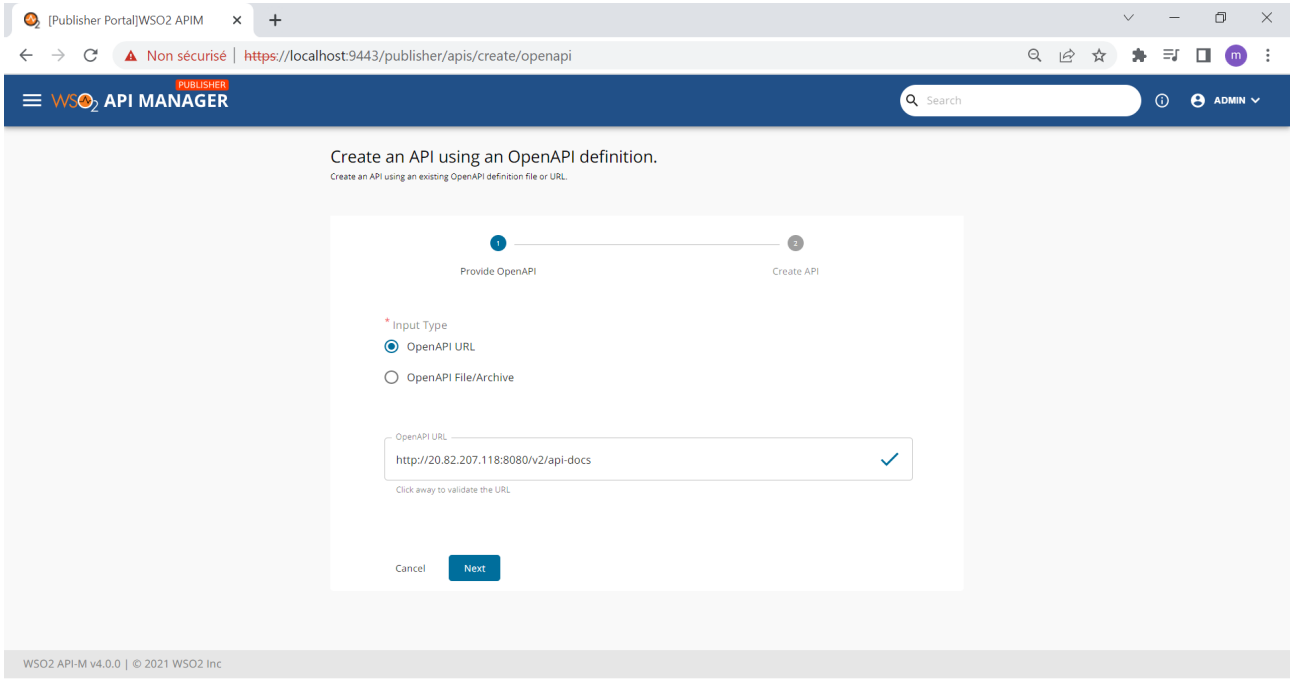


Figure 3.6: Swagger Importation

where we can configure our endpoint as well as an API version :

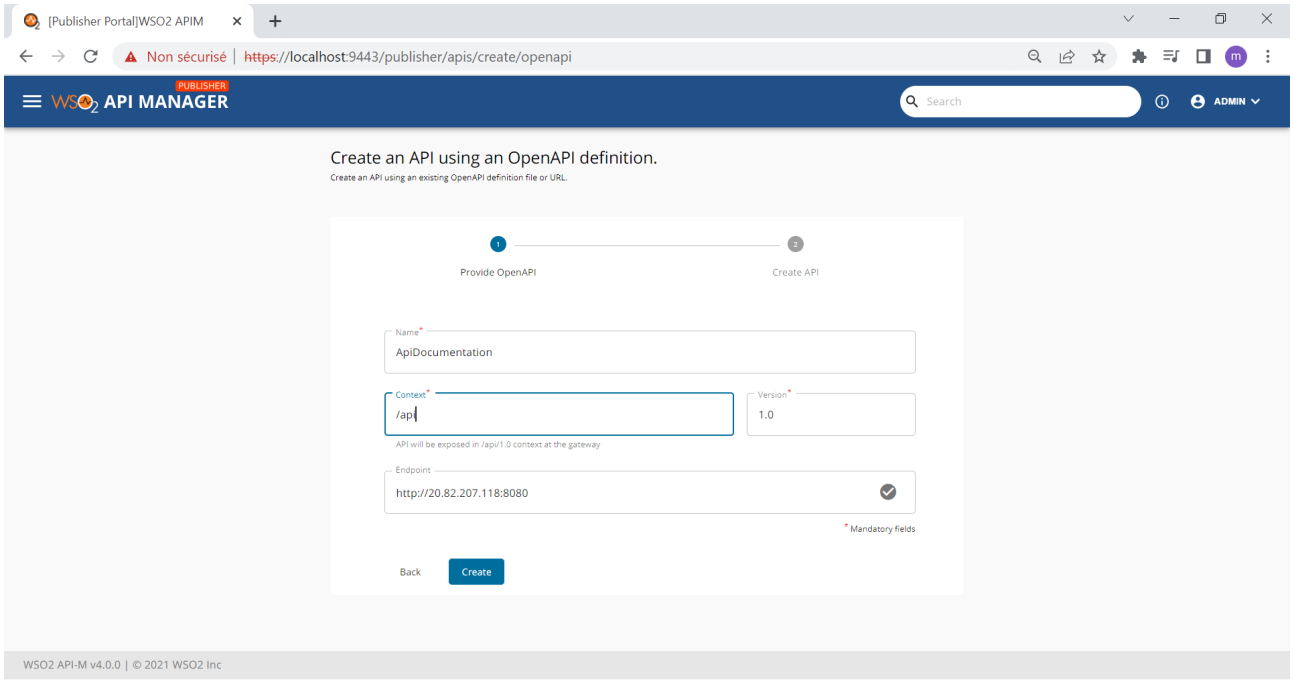


Figure 3.7: Swagger Importation step 2

Then we can use the publisher interface to change our resources, which gives us access to features such as:

- API endpoints naming scheme and parameters

- Rate limiting policies
- Scopes for which users can access the endpoints
- CORS configuration as well as security and TLS policies

We can deploy and publish our API after we've finished editing it, allowing users to access the endpoints in the Developer portal as you can see in the Figure 3.8.

We may access the developer portal after publishing our API and configure it to use an access management application, where each application represents a mobile, web, or desktop application. A subscription is a method that allows us to test our APIs straight from the developer site before deploying our end-user application.

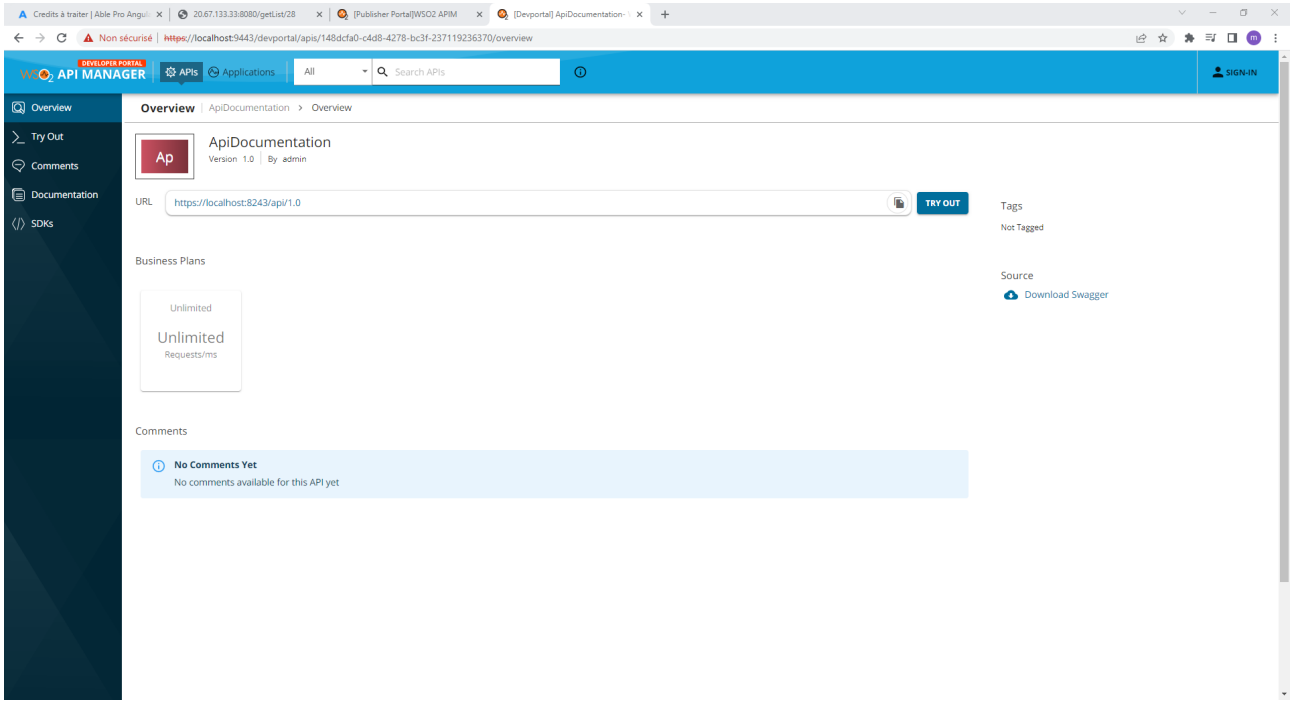


Figure 3.8: Developer Portal Interface

3.4 Conclusion

We discussed what WSO2 is and why we utilize it in this chapter. We also went through its many components before giving a quick rundown of how to connect an API into a single common gateway.

Chapter 4

Sprint 2: Authentication and role management

Plan

4.1 Sprint Backlog 28

4.2 Key concepts 29

4.2.1 Keycloak 30

4.2.2 Acess token 30

4.2.3 Keycloak session 30

4.2.4 Realm 30

4.3 Requirements specification 31

4.3.1 Role management 31

4.3.2 User management 32

4.4 Implementation 33

4.5 Conclusion 34

Introduction

We'll discuss the many steps we did during this second sprint to set up authentication and role management after evaluating and outlining the general requirements for our project. The sprint backlog will be presented first, and then a thorough analysis will follow. The captured interfaces will then be displayed.

4.1 Sprint Backlog

We were able to determine the tasks that needed to be completed during this sprint from the Product Backlog that was shown in the previous chapter.

The Sprint Backlog is displayed in this section in the following table

User Stories	Tasks	Estimation
Story Less	Sprint use Case du 1	5h
	Create the sequence diagram	3h
	Create the Class analysis diagram	8h
	Create the Class Design diagram	3h
	create and deploy the database	3h
	Research Keycloak	2h
	create the architecture of the application	8h

1.1 As an admin, I want to authenticate myself	1.1.1 Create the keycloak realm	1h30
	1.1.2 Configure the keycloak service	3h30
1.2 As an admin, I want the error cases to be handled	1.2.1 Implement the keycloak errors handler	1h
	.1.2.2 Test	1h
1.3 As an admin, I want to disconnect	1.3.1 Configure the app.guard and the app.init	5h
	1.3.2 Implement the keycloak authentication service and the keycloak token handler	5h
	1.3.3 Test	1h
1.4 As an admin, I want to reset my password	1.4.1 Configure the authentication adapter	5h
	1.4.2 Add and configure the reset password privilege	5h
	1.4 Test	1h

Figure 4.1: Sprint 2 Backlog

4.2 Key concepts

We outline the essential components for implementing authentication in this section.

4.2.1 Keycloak

Keycloak is a cutting-edge, application- and service-oriented open source identity and access management system. It makes it possible to secure services and applications. ‘

4.2.2 Acess token

An access token is a token issued by the keycloak server that enables an application to access a resource. Access tokens having a relatively short lifetime are sent to client applications by the keycloak server (usually less than a few minutes). An access token is presented to the resource server by the client application. To determine whether the client program is authorized to access the resource, the resource server verifies the access token.

4.2.3 Keycloak session

The keycloak session is established when :

- When a user successfully authenticates with keycloak
- The session is shown in the session list

The user can travel between the many realm applications after being granted access through a session.

4.2.4 Realm

In Keycloak, a realm is an organization that looks after a set of users' credentials, roles, and groups. Each user in Keycloak only has access to one realm, which he connects to upon registration. One can have multiple realms on a Keycloak server, each of which is independent of the others and of the others and controls just its own user base.

4.3 Requirements specification

We will describe each of the requirements that we must meet throughout this sprint in this section.

The authentication system's flowchart is displayed in Figure 4.2 : The token ensures that the user's session is managed. Each HTTP request that is sent must include the header produced during authentication in order to permit the client to access the application's various services, depending on the token's validity.

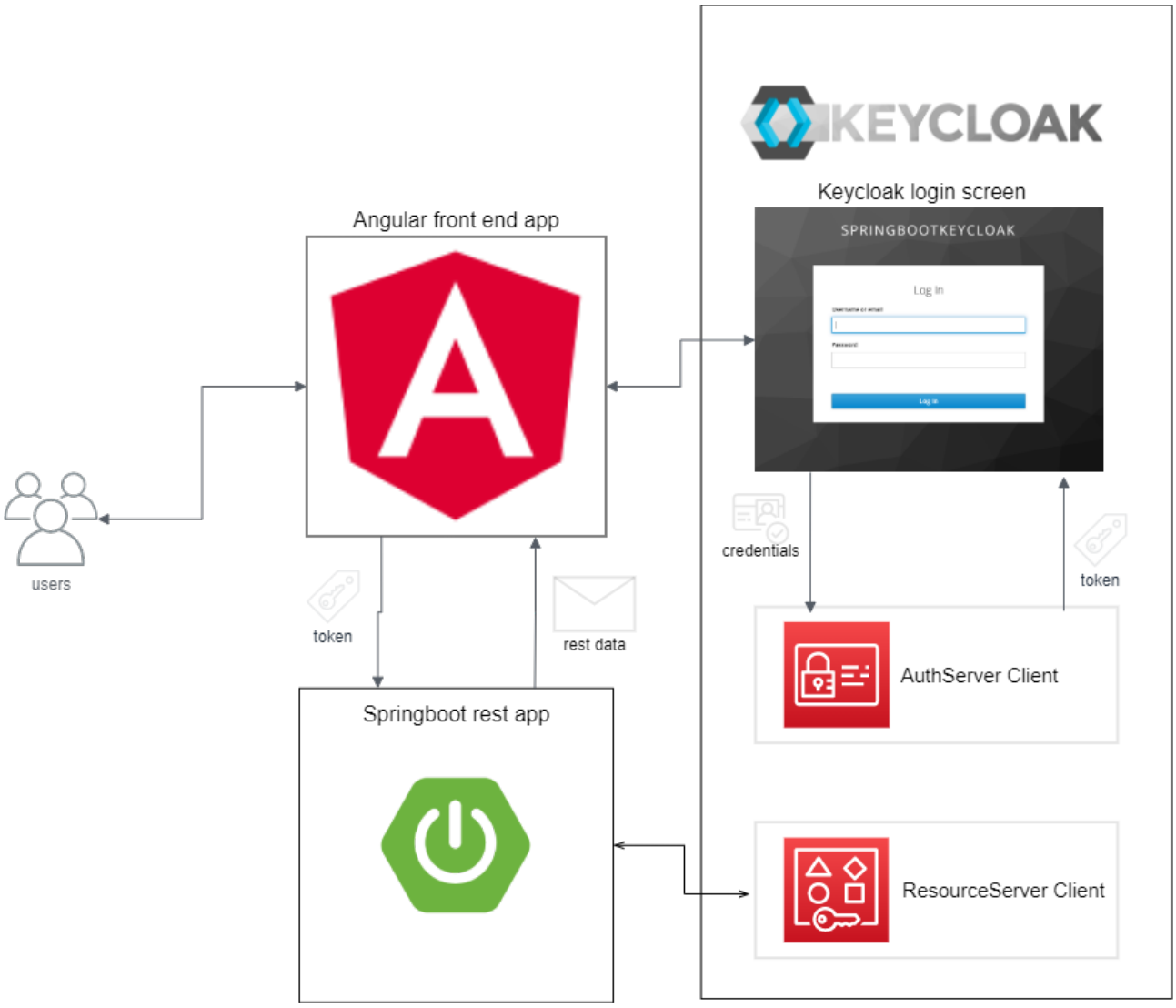


Figure 4.2: Authentication scenario with keycloak

4.3.1 Role management

Figure 4.3 displays the management of roles in the keycloak authentication server.

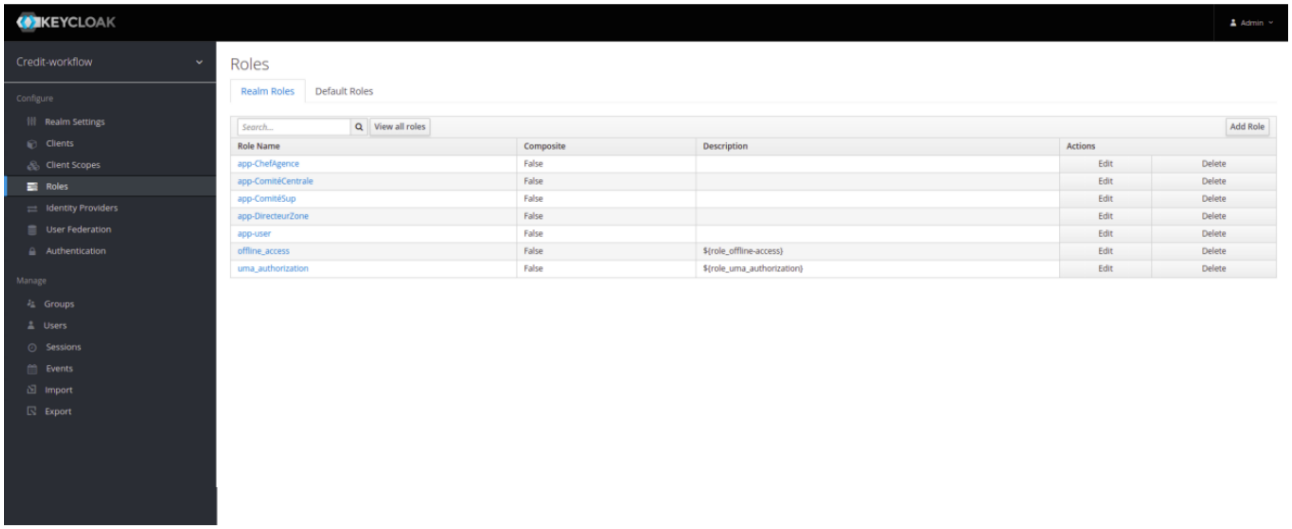


Figure 4.3: Role management in Keycloak

There are five types of roles in our system:

- **app-user:** this is the role of a simple client.
- **app-Agency-Head:** this is the role of an agency head.
- **app-ZoneManager:** this is the role of a zone manager.
- **app-SuperiorCommittee :** it's the role of a superior committee.
- **app-CentralCommittee :** this is the role of the central committee.

4.3.2 User management

The users of the system in keycloak are represented in figure 4.4 each user has an assigned role that grants him access to particular rights.

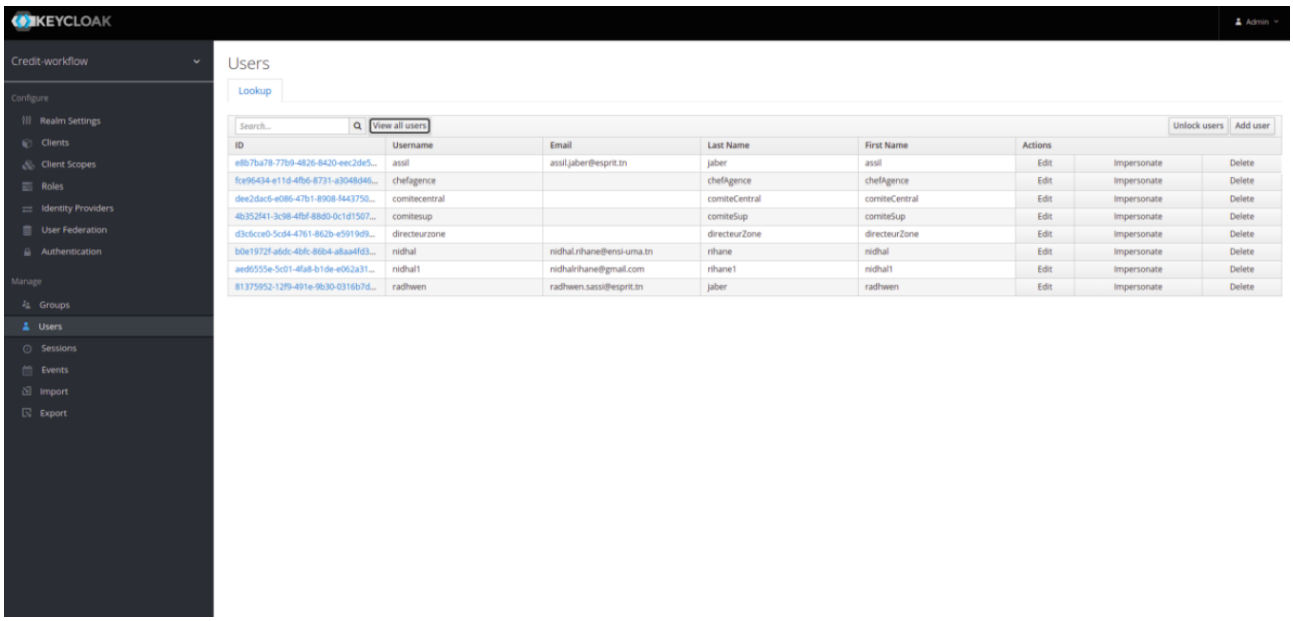


Figure 4.4: User management in Keycloak

4.4 Implementation

To access the functionalities offered by the application, any agent must connect through an interface illustrated in figure 4.5

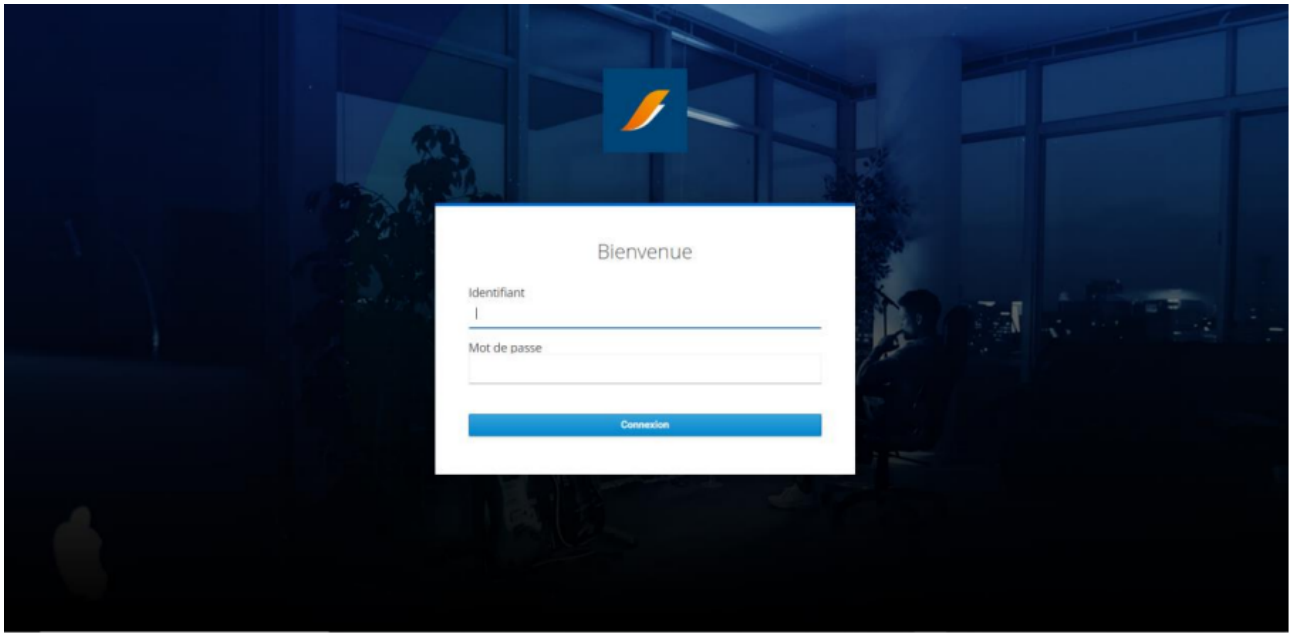


Figure 4.5: Authentication interface

4.5 Conclusion

This chapter contains a presentation of the first sprint's "Authentication" implementation. The API call and the scenario for authentication were both defined. In addition, we displayed some Keycloak grabs. Finally, we showed off a few of this sprint's interfaces.

Chapter 5

Sprint 3: Brands and Models Management

Plan

5.1 Sprint Backlog 36

5.2 Brands Management 38

5.2.1 Functional specification of the brands management 38

5.3 Models management 42

5.3.1 Functional specification of the models management 42

5.4 Implementation 46

5.4.1 Implementation of the brand management 46

5.4.2 Implementation of the models management 49

Introduction The purpose of this chapter is to go over the process of conceiving the brands and models management, their diagrams and the implementation process. We will start by presenting the sprint backlog followed by a detailed analysis of each part.And we will then present the interfaces of the application.

5.1 Sprint Backlog

User Stories	Tasks	Estimation
2.1 As an admin, I want to display the list of car brands	2.1.1 Create the display DAO, service and controller	2h
	2.1.2 Integrate the the display module to the template	5h
	2.1.3 Populate the database and testing	1h
2.2 As an admin, I want to add a car brand	2.2.1 Create the add brand service and controller	5h
	2.2.2 Expose the API to WSO2 and make it a public scope	5h
	2.2.3 Create the add modal and integrate the the add brand module to the template	1h

2.3 As an admin, I want to change a brand.	2.3.1 Create the modify brand service and controller and expose it to WSO2	5h
	2.3.2 Create the modify modal and integrate the the modify brand module to the template	5h
2.4 As an admin, I want to delete a brand.	2.4.1 Create the delete brand service and controller and expose it to WSO2	5h
	2.4.2 Implement the delete method in the table	5h
	2.4.2 Test	1h
2.5 As an admin, I want to search a brand.	3.5.1 Create the search brand service and controller and expose it to WSO2	5h
	R.3.6.2 Implement the search method and Add the filter with name	5h
	R.3.6.3 Test	1h
3.1 As an admin, I want to display the list of car model	3.1.1 Create the display DAO, service and controller	1h
	3.1.2 Integrate the the display module to the template	4h
	3.1.3 Populate the database and testing	1h
3.2 As an admin, I want to add a car model	3.2.1 Create the add model service and controller	3h
	3.2.2 Expose the API to WSO2 and make it a public scope	1h

3.3 As an admin, I want to change a model.	3.3.1 Create the modify model service and controller and expose it to WSO2	3h
	3.3.2 Create the modify modal and integrate the the modify model module to the template	3h
3.4 As an admin, I want to delete a model.	3.4.1 Create the delete model service and controller and expose it to WSO2	3h
	3.4.2 Implement the delete method in the table	3h
	3.4.2 Test	1h
3.5 As an admin, I want to search a model.	3.5.1 Create the search model service and controller and expose it to WSO2	3h
	3.6.2 Implement the search method and Add the filter with name	2h
	R.3.6.3 Test	1h

Figure 5.1: Sprint 3 Backlog

5.2 Brands Management

We have examined the Brand Management in detail in this section.

5.2.1 Functional specification of the brands management

UML diagrams will be used in this section: the use case diagram and sequence diagrams.

5.2.1.1 Use case diagram of the brands management

The use case diagram for the brand management is presented in diagram 5.2.

The admin has the possibility to add ,modify or delete any brand he want. Also all the existed brands are showed in a table that contains all the brand details such as the name,country and the logo of the brand. In addition to that,the admin can do search by the brand name to make the browsing easier.

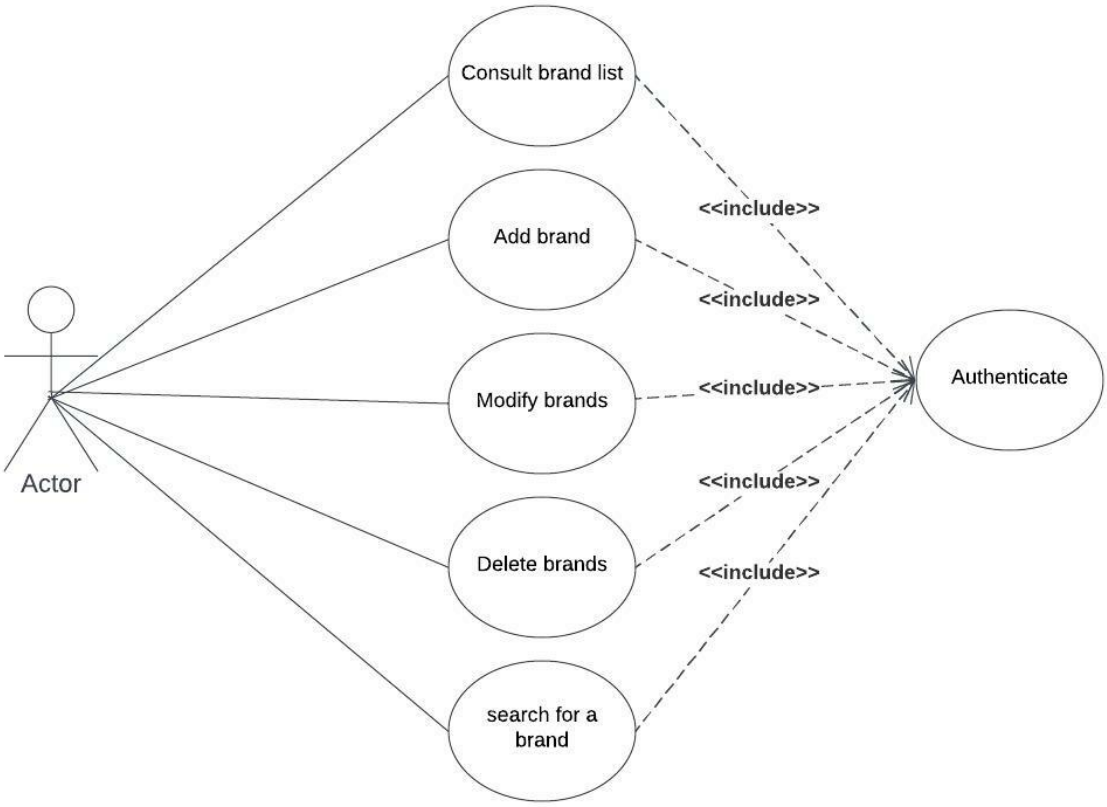


Figure 5.2: Sprint Use case Diagram

5.2.1.2 Class diagram of the brands management

The diagram below represent the brand class diagram :

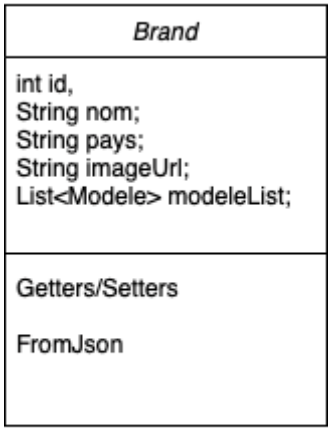


Figure 5.3: Sprint Brand Class Diagram

5.2.1.3 Sequence diagram

after authentication the admin can consult the list of brands by clicking on the "Marques" Button which he can find it in the navigation bar also he can search for a specific brand with the help of search bar. He can add a new brand by clicking on the "+" button then putting the brand information and confirm by clicking on the submit button.

In addition to that the admin can edit the brand characteristics by clicking on "Modifier" button and then modify them and submit. Finally if the brand is not needed anymore the admin can simply delete it by clicking on "Supprimer".

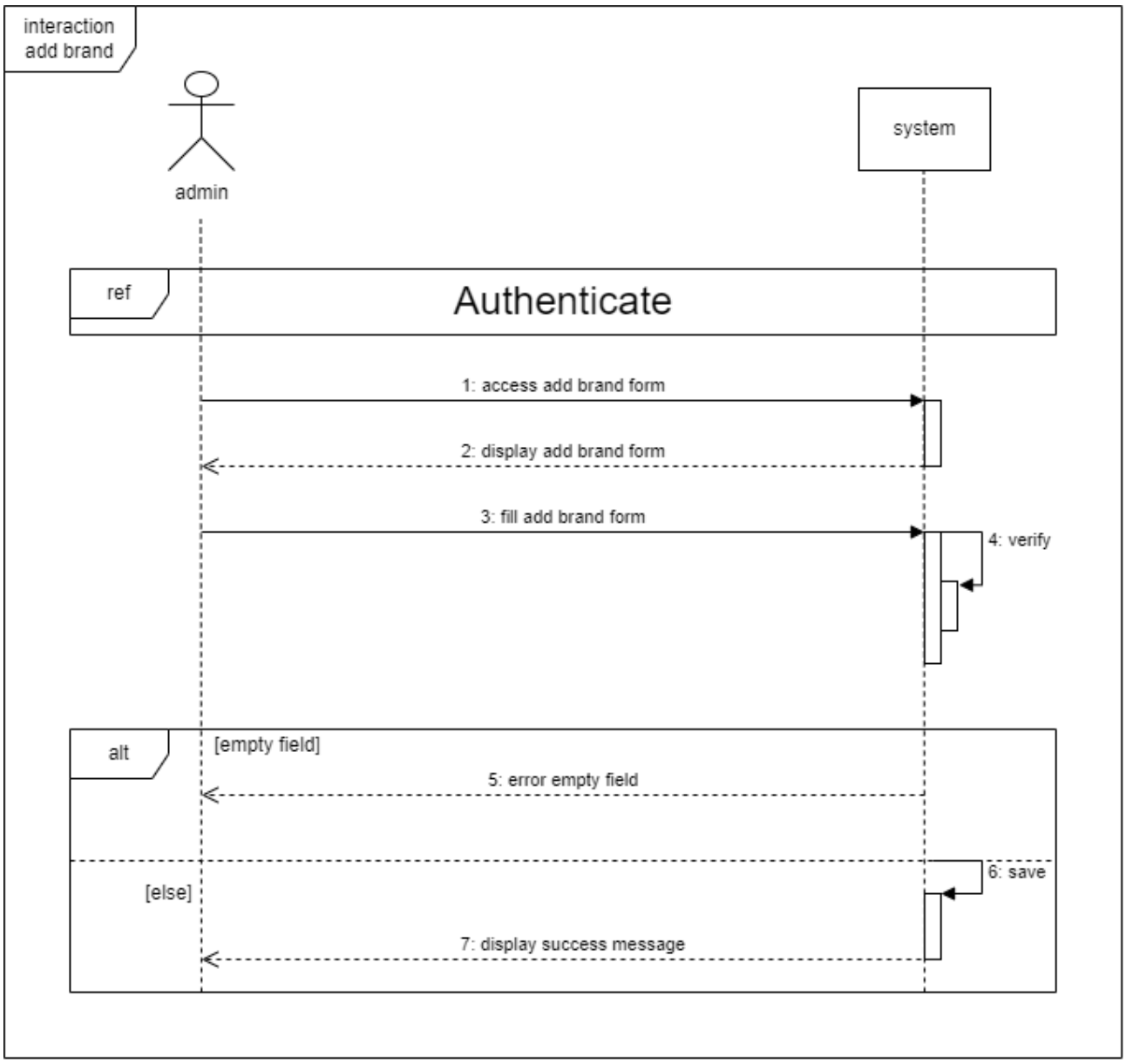


Figure 5.4: Add brand Sequence diagram

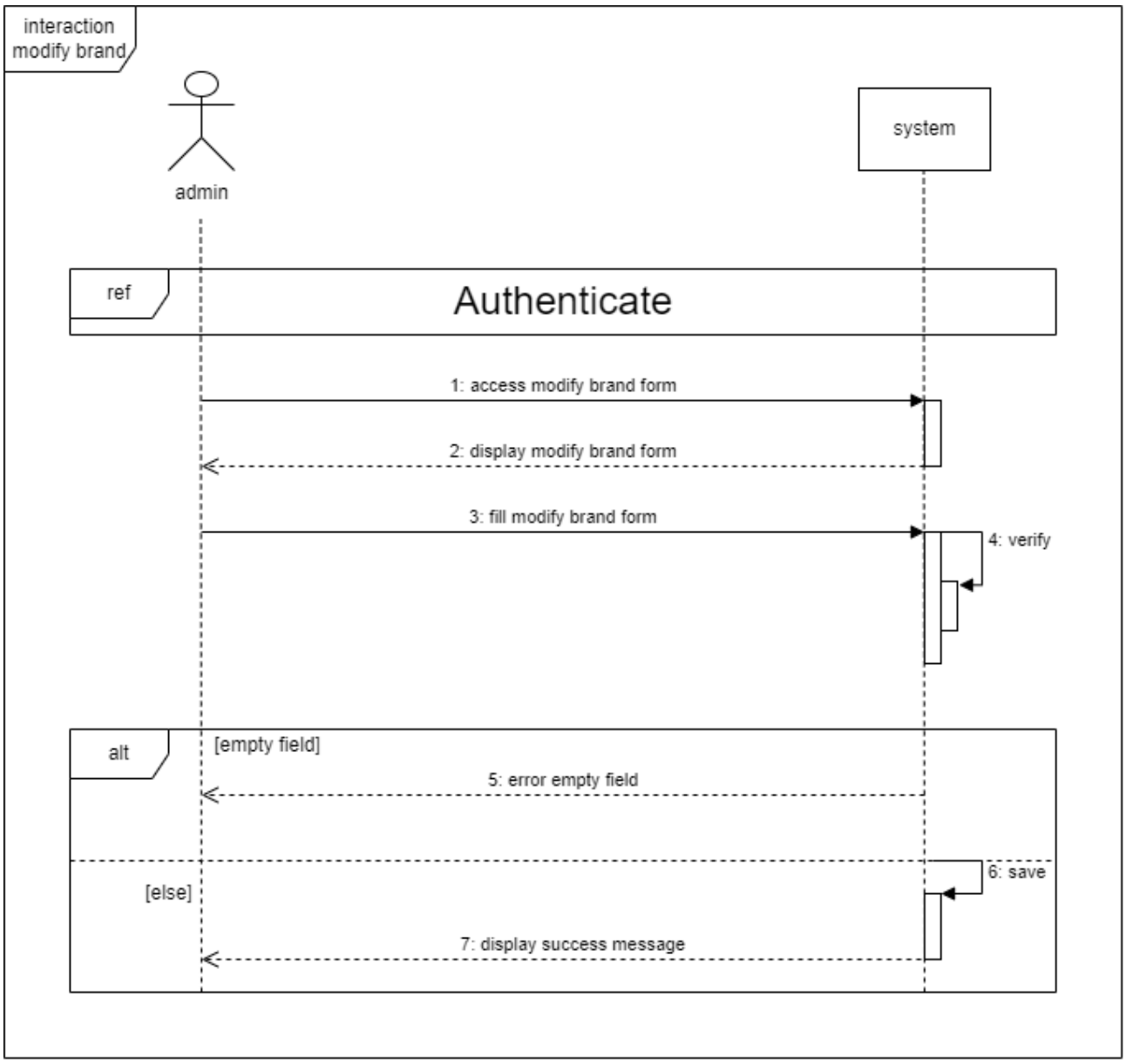


Figure 5.5: Modify brand Sequence diagram

5.3 Models management

We have examined the Models Management in detail in this section.

5.3.1 Functional specification of the models management

UML diagrams will be used in this section: the use case diagram and sequence diagrams.

5.3.1.1 Use case diagram of the models management

The use case diagram for the brand management is presented in diagram 5.6. The admin has the possibility to add ,modify or delete any model he want. Also all the existed models are showed in a table that contains all the brand details such as the name,country and the logo of the model.

In addition to that,the admin can do search by the model name to make the browsing easier.

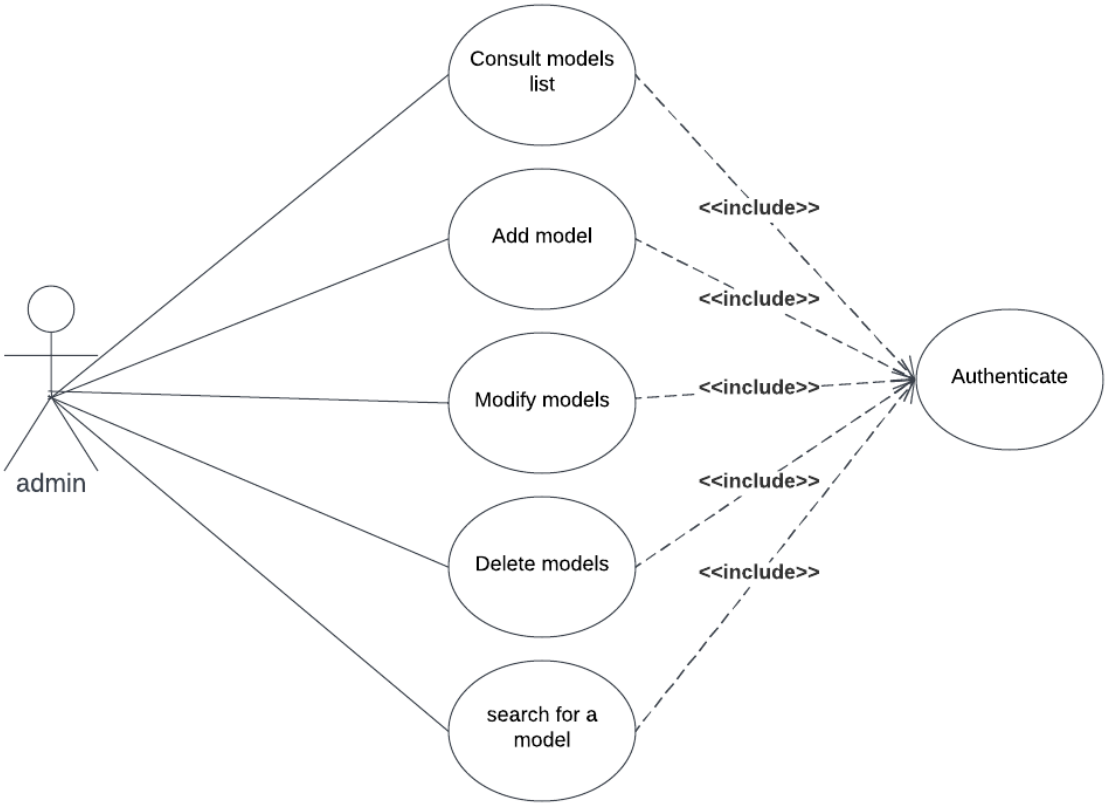


Figure 5.6: Model Use case Diagram

5.3.1.2 Class diagram of the brand management

The diagram below represent the model class diagram :

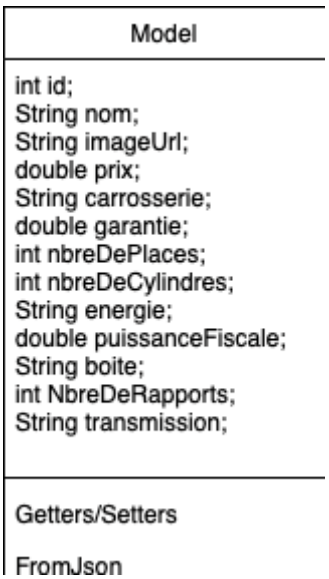


Figure 5.7: Model Class Diagram

5.3.1.3 Sequence diagram

after authentication the admin can consult the list of models by clicking on the "Modèles" Button which he can find it in the navigation bar also he can search for a specific model with the help of search bar.

He can add a new model by clicking on the "+" button then putting the model information then affect it to one of the brand then confirm by clicking on the submit button.

In addition to that the admin can edit the model characteristics by clicking on "Modifier" button and then modify them and submit. Finally if the model is not needed anymore the admin can simply delete it by clicking on "Supprimer".

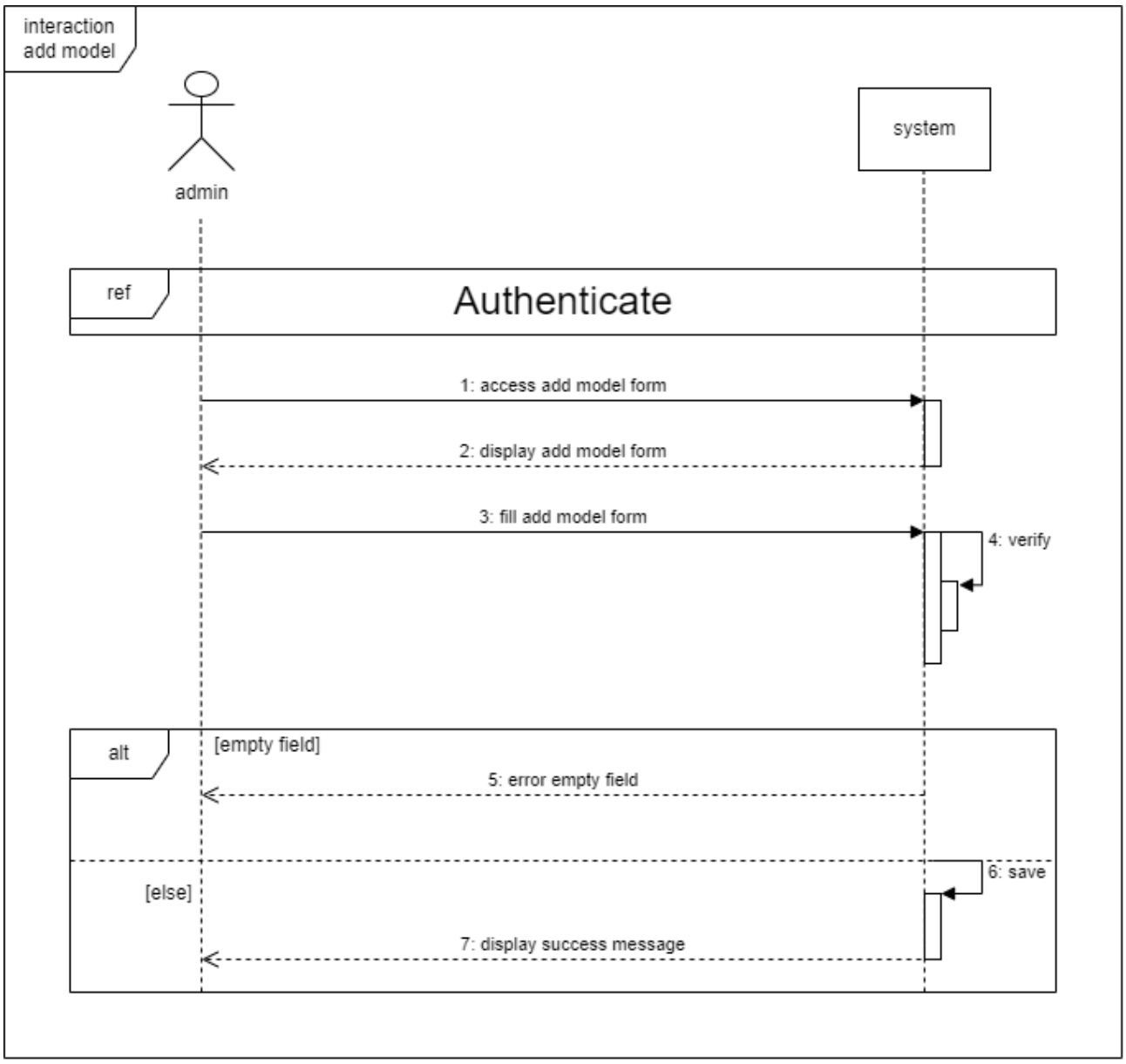


Figure 5.8: Add model Sequence diagram

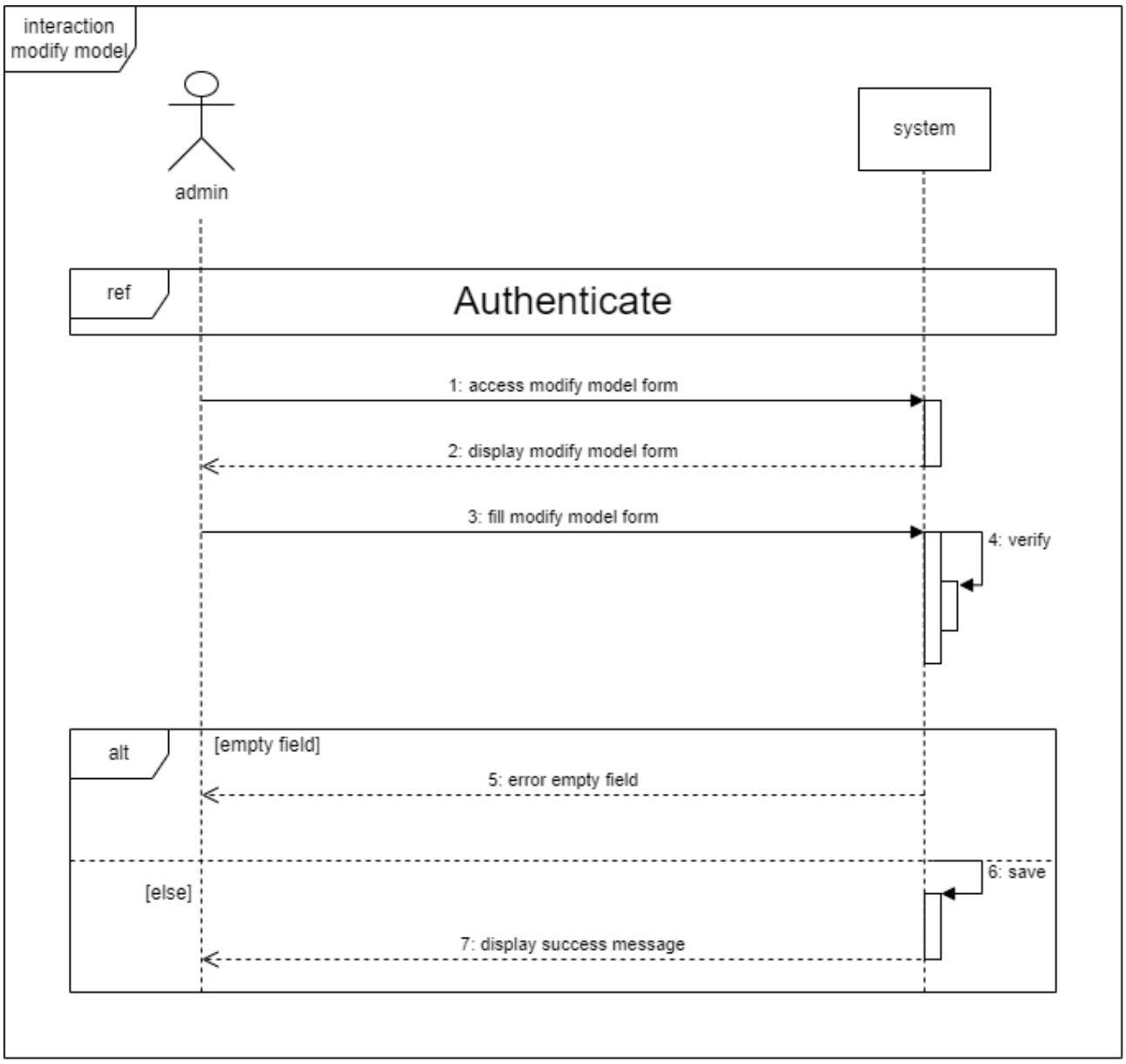


Figure 5.9: Modify model Sequence diagram

5.4 Implementation

5.4.1 Implementation of the brand management

The Figure 5.10 represents the brand interface that is displayed as soon as you click on "Marques".

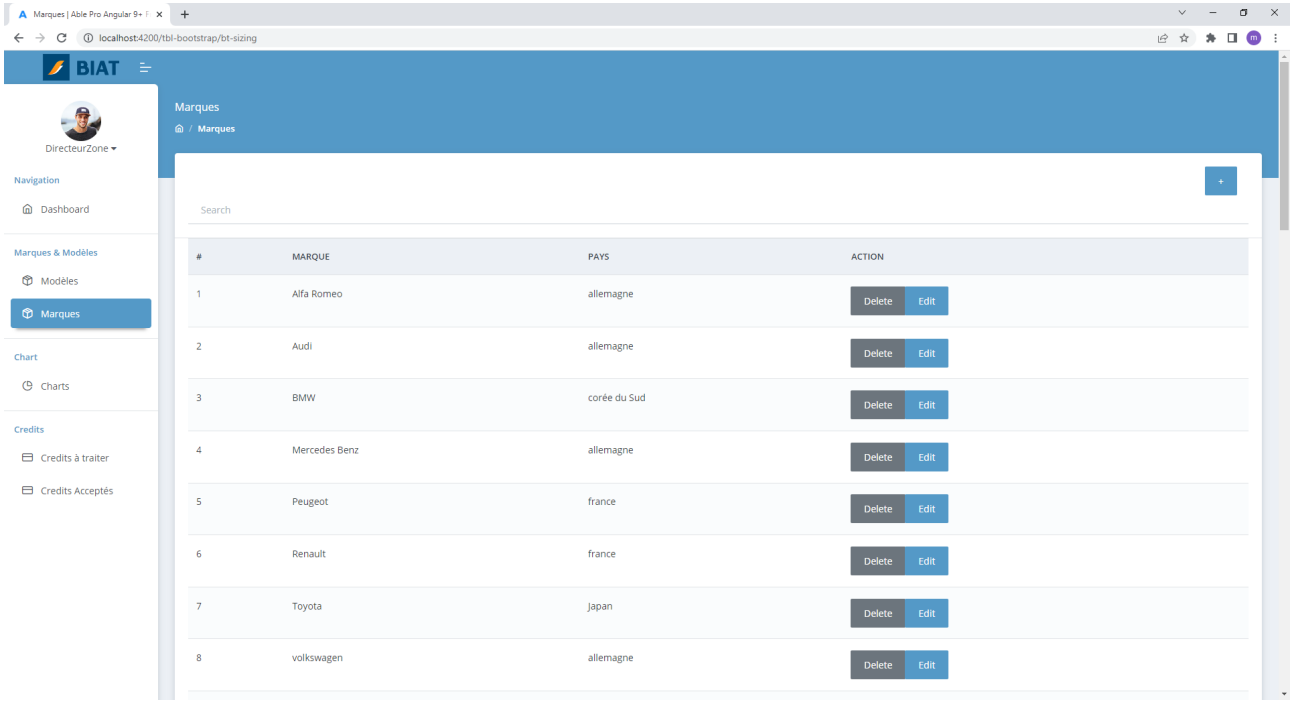


Figure 5.10: Brand Interface

The admin can click on "+" to bring up the interface 5.11 and add a new brand. The user then has the option to either complete the form and submit it or cancel it by clicking on "x" .

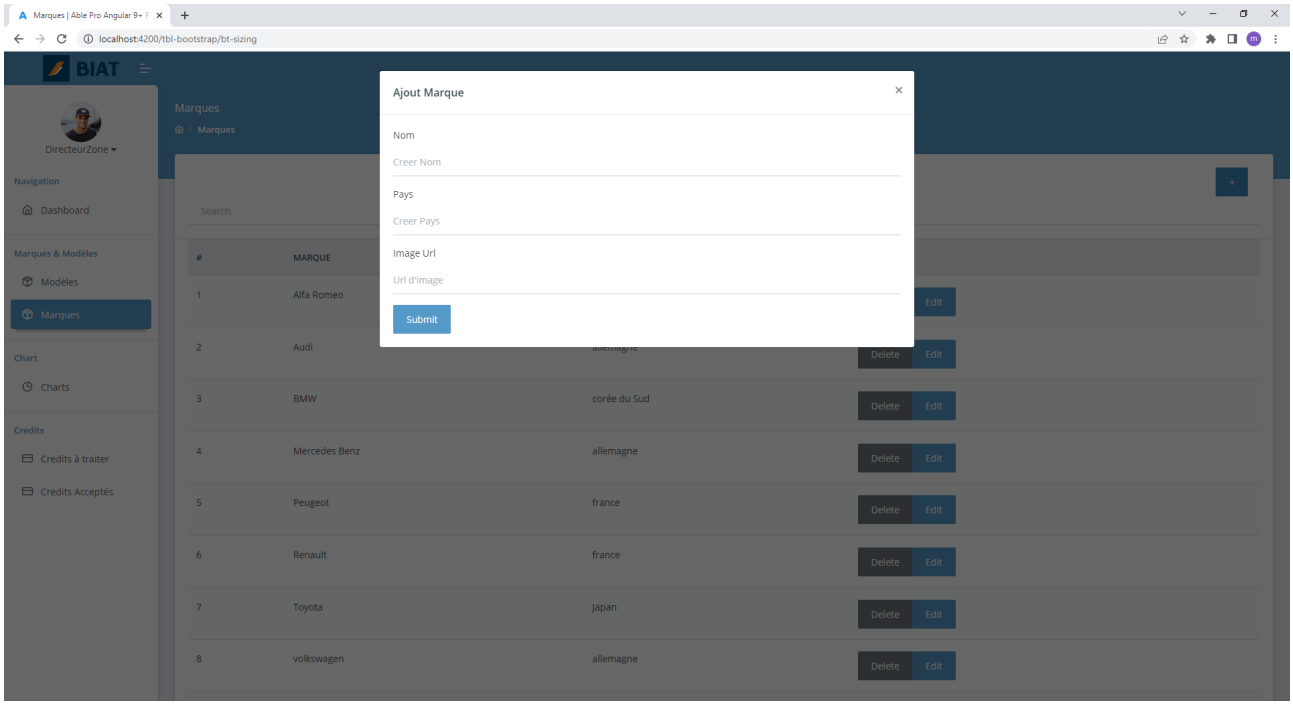


Figure 5.11: Brand Adding Interface

The admin can search for a specific brand with the help of the search bar like in the figure 5.12

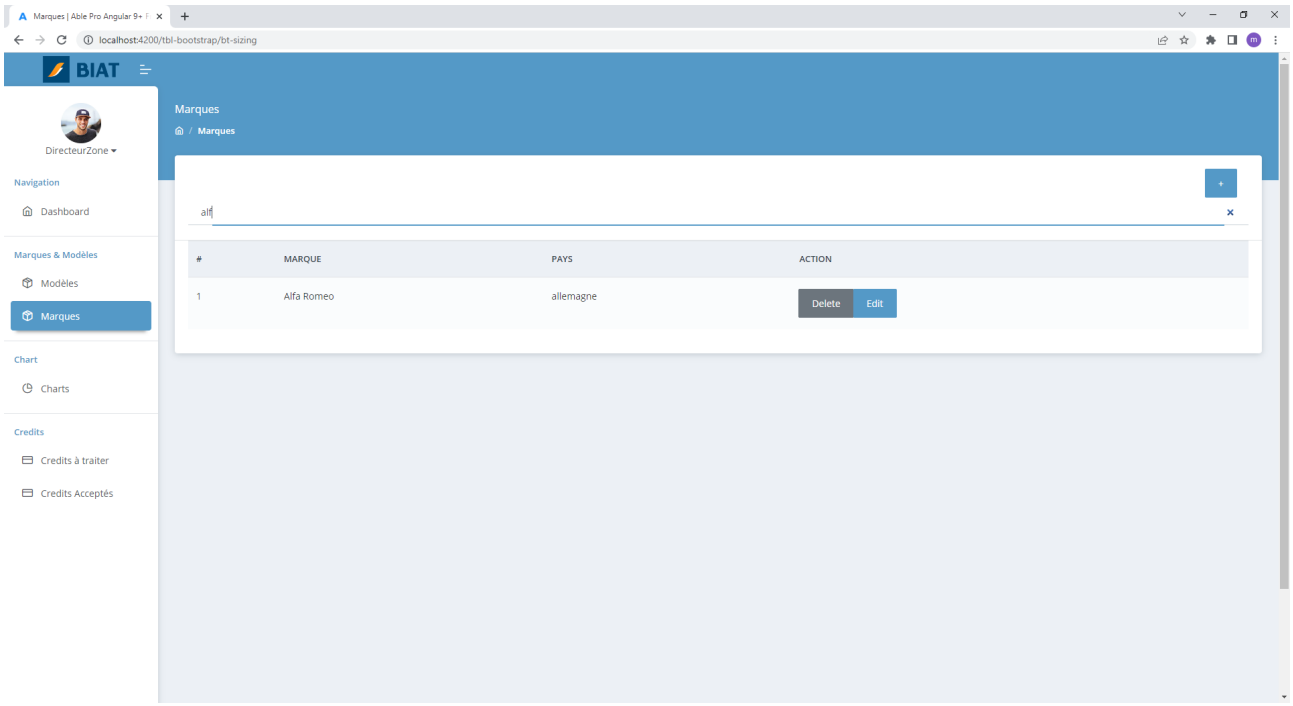


Figure 5.12: Search Interface

After choosing the brand, the admin can either delete the brand by clicking on "Supprimer"or just modify it by pressing on the "Modifier " button which will the interface in figure 5.13 to pop up then he can modify the information and save the modification

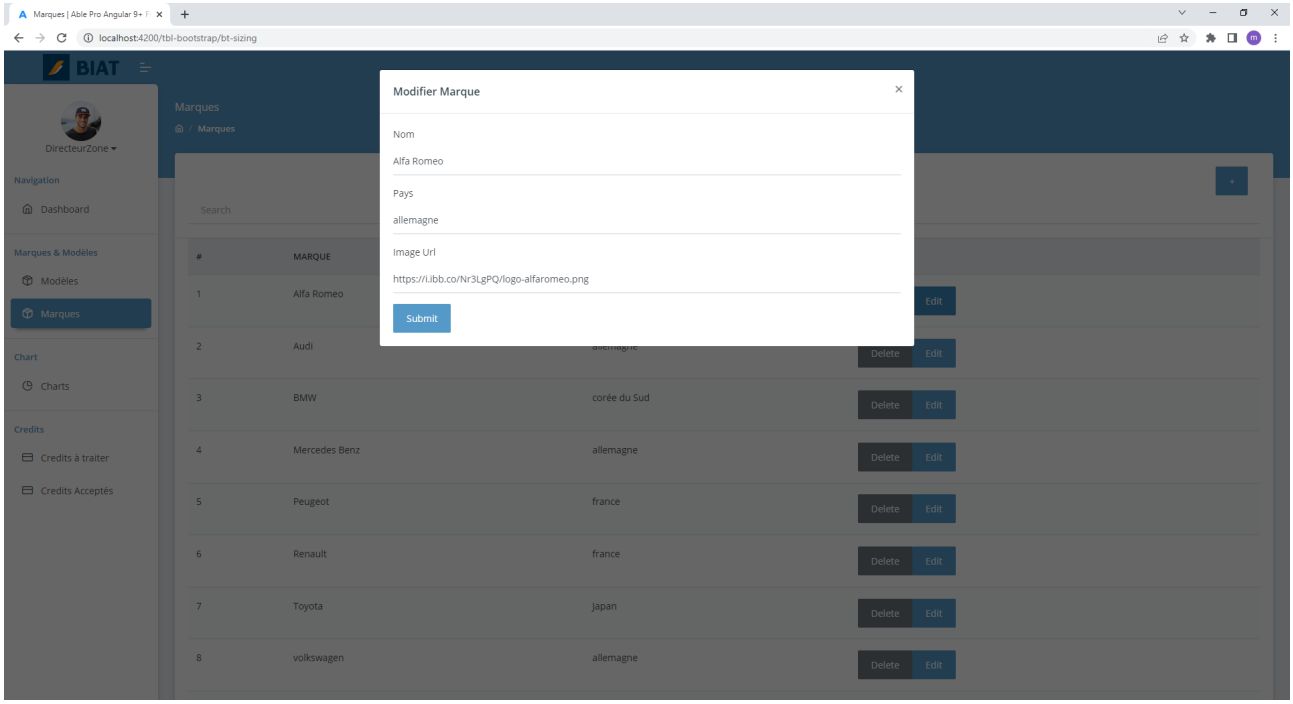


Figure 5.13: Brand Modifying Interface

5.4.2 Implementation of the models management

The Figure 5.14 represents the brand interface that is displayed as soon as you click on "Modéles".

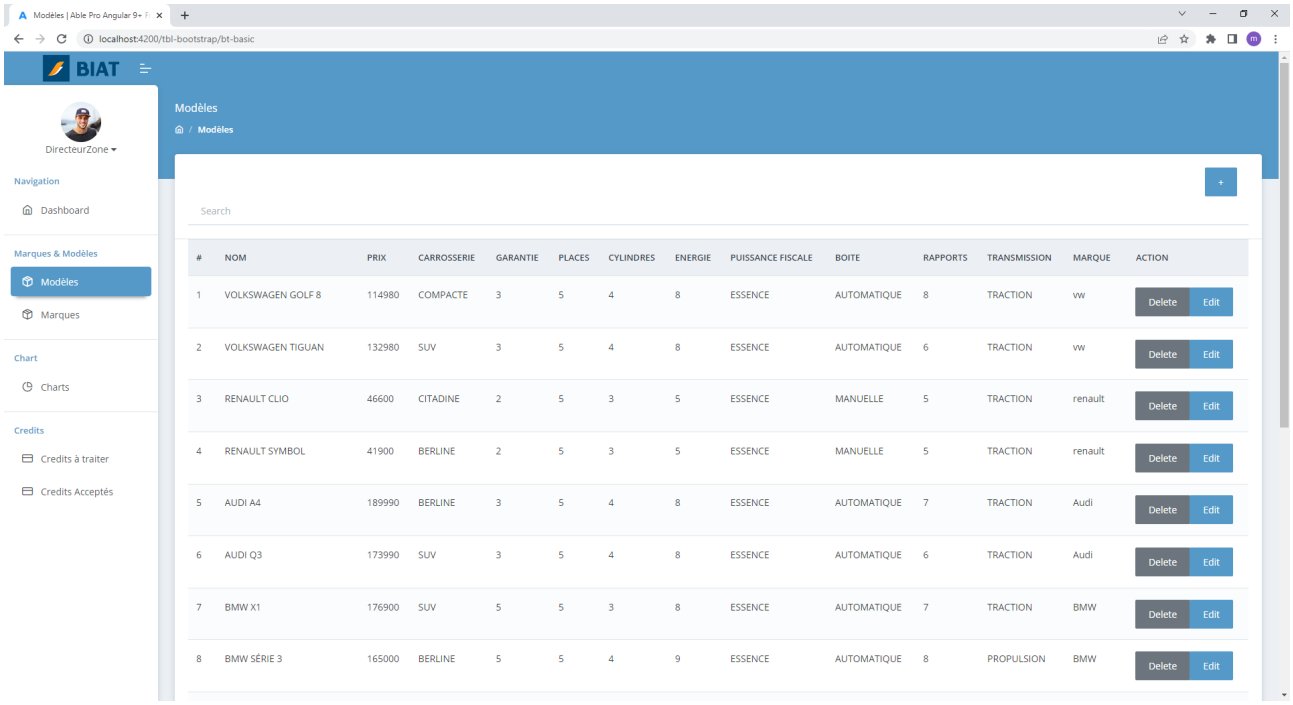


Figure 5.14: Models Interface

The admin can click on "+" to bring up the interface 5.15 and add a new Model. The admin then has the option to either complete the form and submit it or cancel it by clicking on "x" .

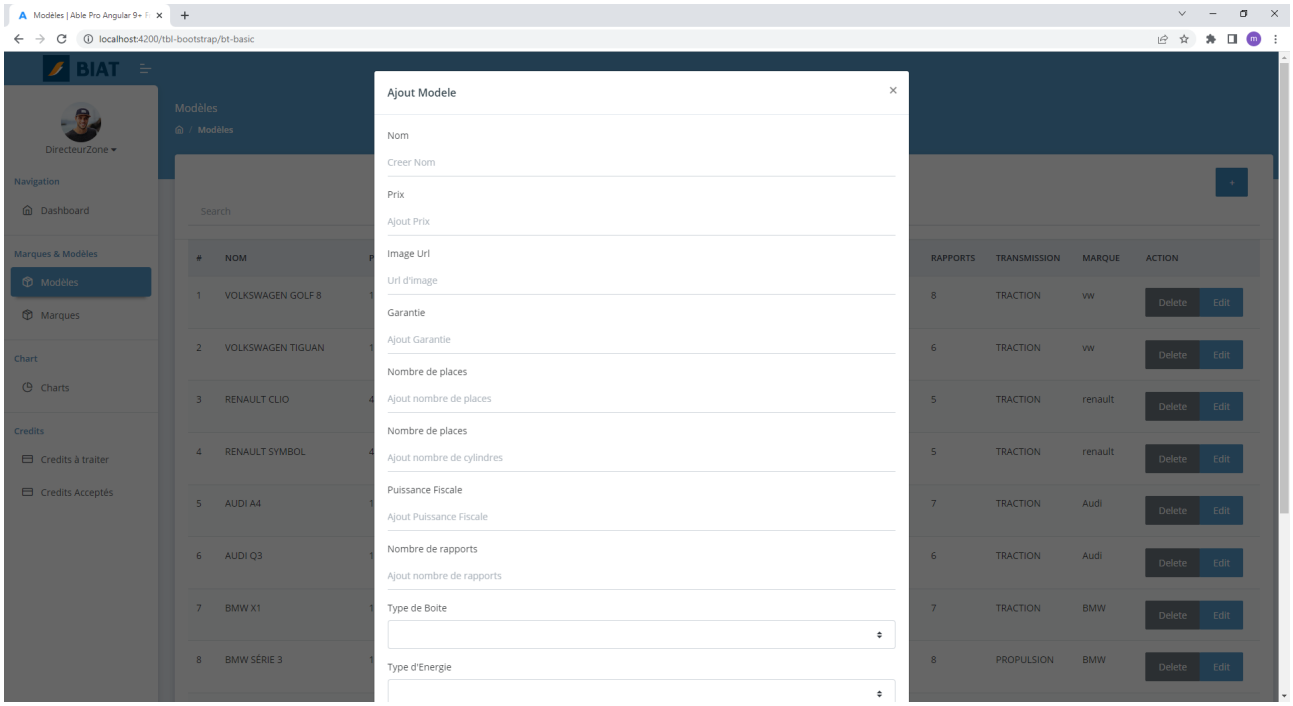


Figure 5.15: Models Adding Interface

The admin can search for a specific model with the help of the search bar like in the figure 5.16

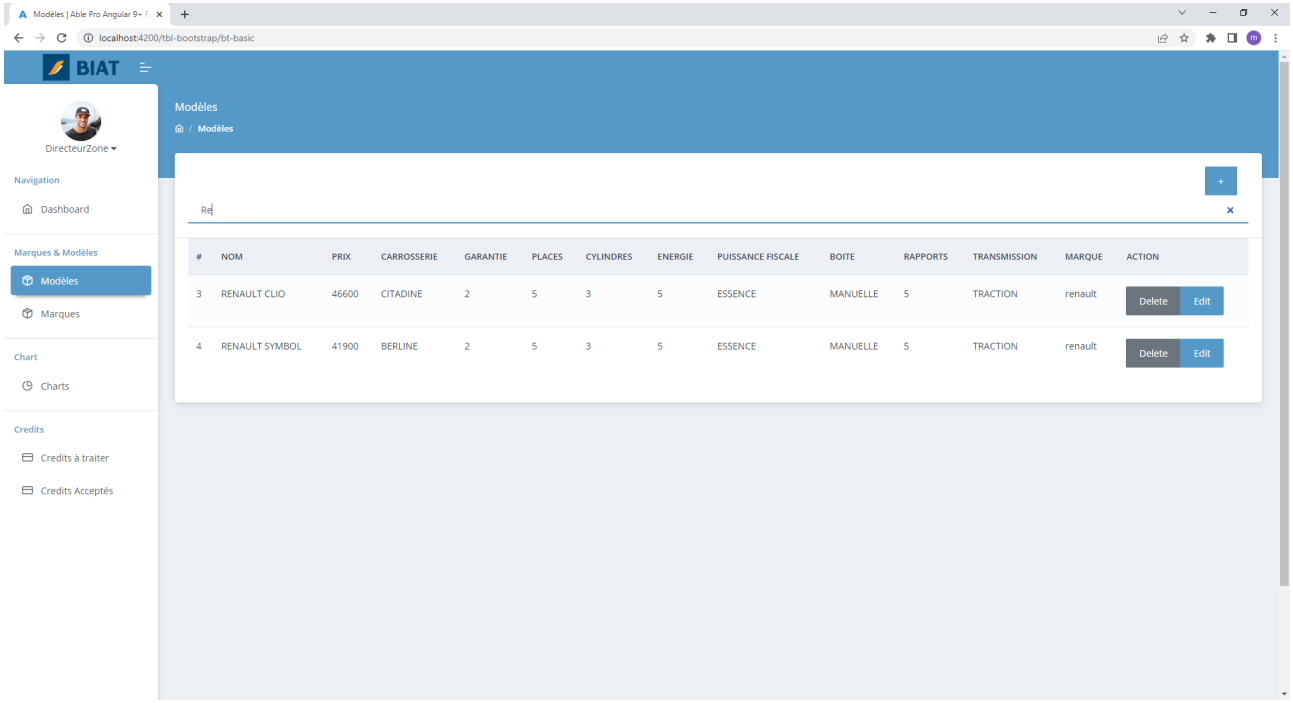


Figure 5.16: Models search Interface

After choosing the model, the admin can either delete the model by clicking on "Supprimer"or just modify it by pressing on the "Modifier " button which will the interface in figure 5.17 to pop up then he can modify the information and save the modification

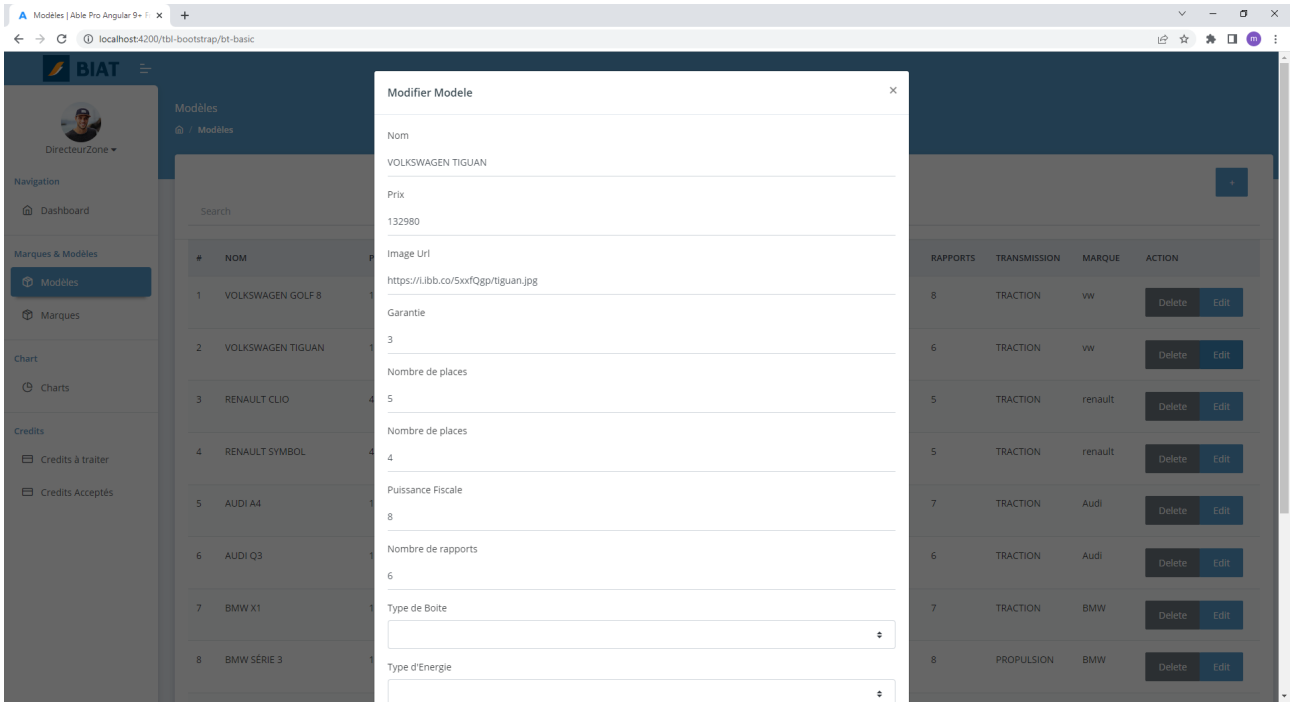


Figure 5.17: Model Modifying Interface

Conclusion

We have covered the specifics of managing brands and models in this chapter. We have modeled various diagrams of our application and shown the brand management microservice before moving on to the models management microservice. We also used a few of our application's APIs to demonstrate the realization component. We will look at credit management and workflow management in the following chapter.

Chapter 6

Sprint 4: Credit Management and Dashboard

Plan

6.1 Sprint Backlog 54

6.2 Credit management 56

6.2.1 Use case diagram 56

6.2.2 sequence diagram 58

6.3 Dashboard 60

6.3.1 Use Case diagram 60

6.3.2 Sequence diagram 61

6.4 Implementation 61

6.4.1 Credit management 61

6.4.2 Dashboard 67

Introduction

After finishing the sprint 3. we introduce the sprint 3, which is composed of the credit management and the dashboard. We will start by presenting the sprint backlog followed by a detailed analysis of each part.And we will then present the interfaces of the application.

6.1 Sprint Backlog

In this section, we present the Backlog for this sprint illustrated in the following table

User Stories	Tasks	Estimation
4.1 As an admin, I want to display the list of Credit demands	4.1.1 Identify the usable APIs and Integrate WSO2 With Keycloak	5h
	4..1.2 modify the mailing service and change it to smtp	4h
	4.1.3 Expose the APIs to the WSO2 API Manager	5h
	4.1.4 Integrate the AllCredits and All Credit Accepted to the interface using the Exposed APIs on WSO2 Gateway	5h
	4.1.5 Add the name filter	2h
	4.1.6 Test	1h

4.2 As an admin, I want to Consult the eligibility of demands	4.2.1 Run the Camunda on WSO2 and intercept the workflow	5h
	4.2.2 add the verifyAge and salaryCheck module then make the eligibility check automated using Camunda	5h
	4.2.3 test the Camunda workflow	1h
4.3 As an admin, I want to consult the demands documents	4.3.1 Expose the DocumentService on WSO2 gateway	5h
	4.3.2 Implement the Exposed get List service and the getDateDemande to the interface	5h
	4.3.3 integrate the download documents services to the interface and test	5h
4.4 As an admin, I want send the contract	4.4.1 Expose the ContratService on WSO2 gateway	5h
	4.4.2 Implement the generateCotrat service checkContratStatus and Sendcontrat modules to the interface the contract modal in the interface	5h
	4.4.3 Test	1h

5.1 As an admin, I want to analyze car requests	5.1.1 Create the checkCarNumber , carViewNumber services and expose them to WSO2	5h
	5.1.2 Implement the exposed services to the interface	1h
	5.1.3 Test the integrity of graph of data on the dashboard	1h
5.2 As an admin, I want to analyze the demands of car loans	5.2.1 Implement the All Credit services to the interface and categorized every loan by its status	5h
	R.14.2.2 Create and implement the checkGain and tauxGain services	4h
	5.2.3 test results	1h

Figure 6.1: Sprint 4 Backlog

6.2 Credit management

6.2.1 Use case diagram

The following figures shows the credit management use case diagram

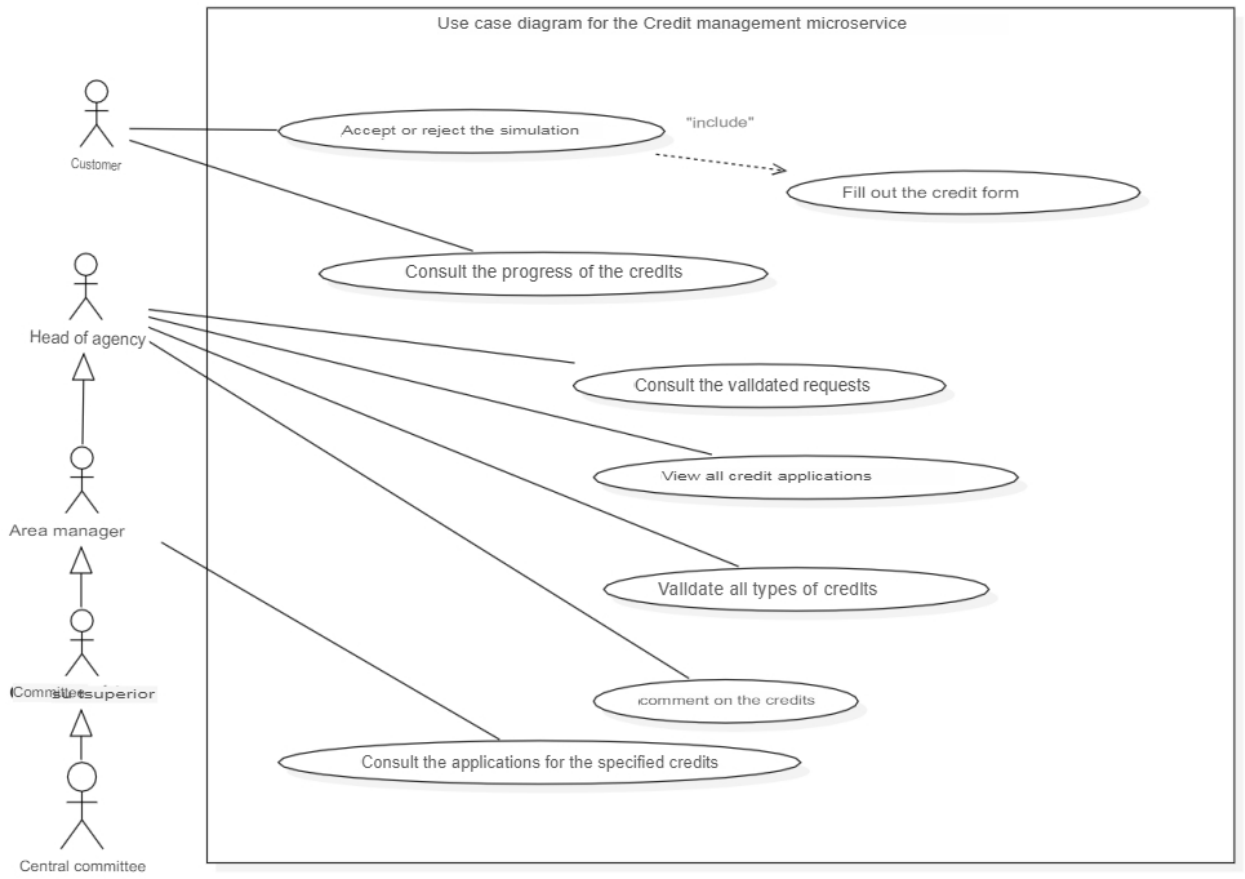


Figure 6.2: Sprint Use case Diagram

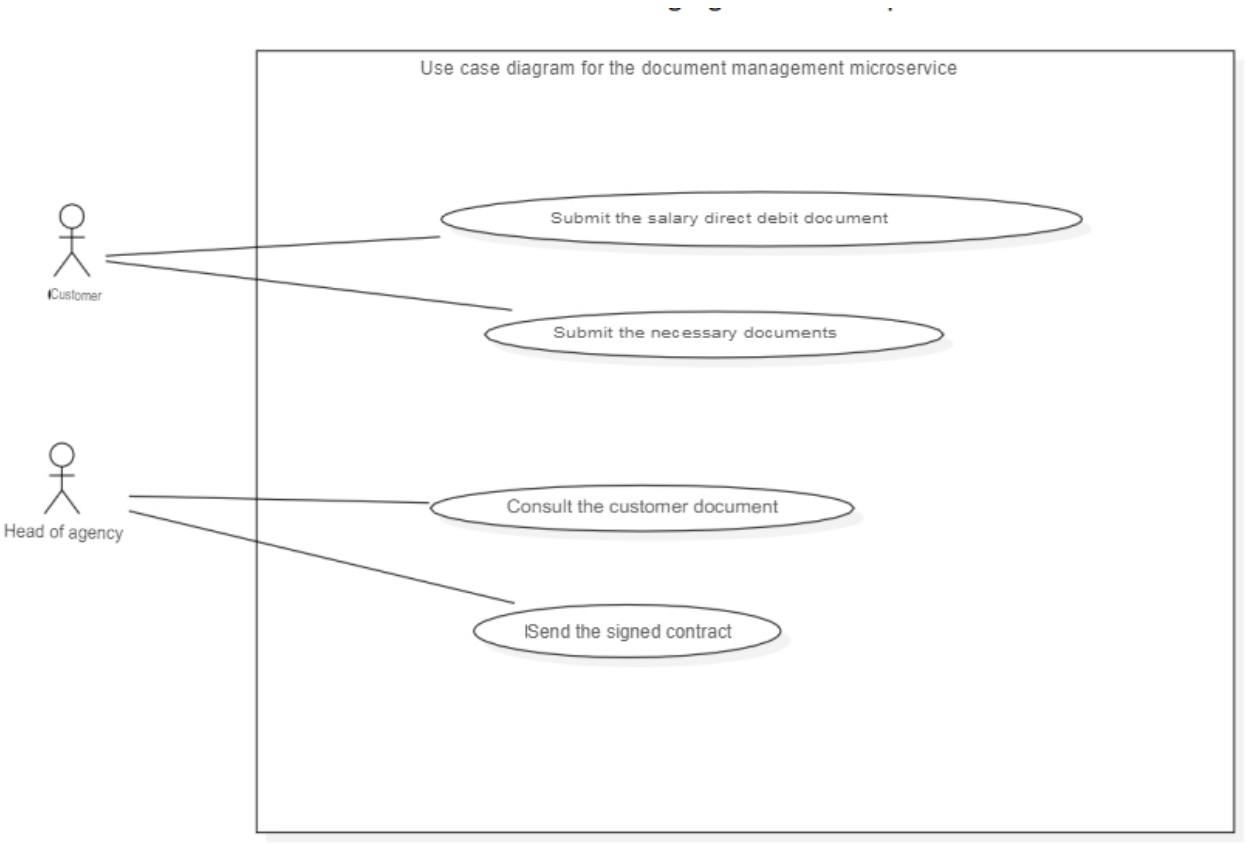


Figure 6.3: Sprint Use case Diagram 2

6.2.2 sequence diagram

As we can see the in the diagram 6.4 ,following authentication, the system displays a list of credit notes to the administrator. The administrator selects a credit application to validate from the list, and the system then displays the credit application details to the administrator. At this point, the administrator can either refuse the application and leave a comment to have the system change the credit status to refused, or he can accept the credit and leave a comment to have the system change the credit status to accepted.

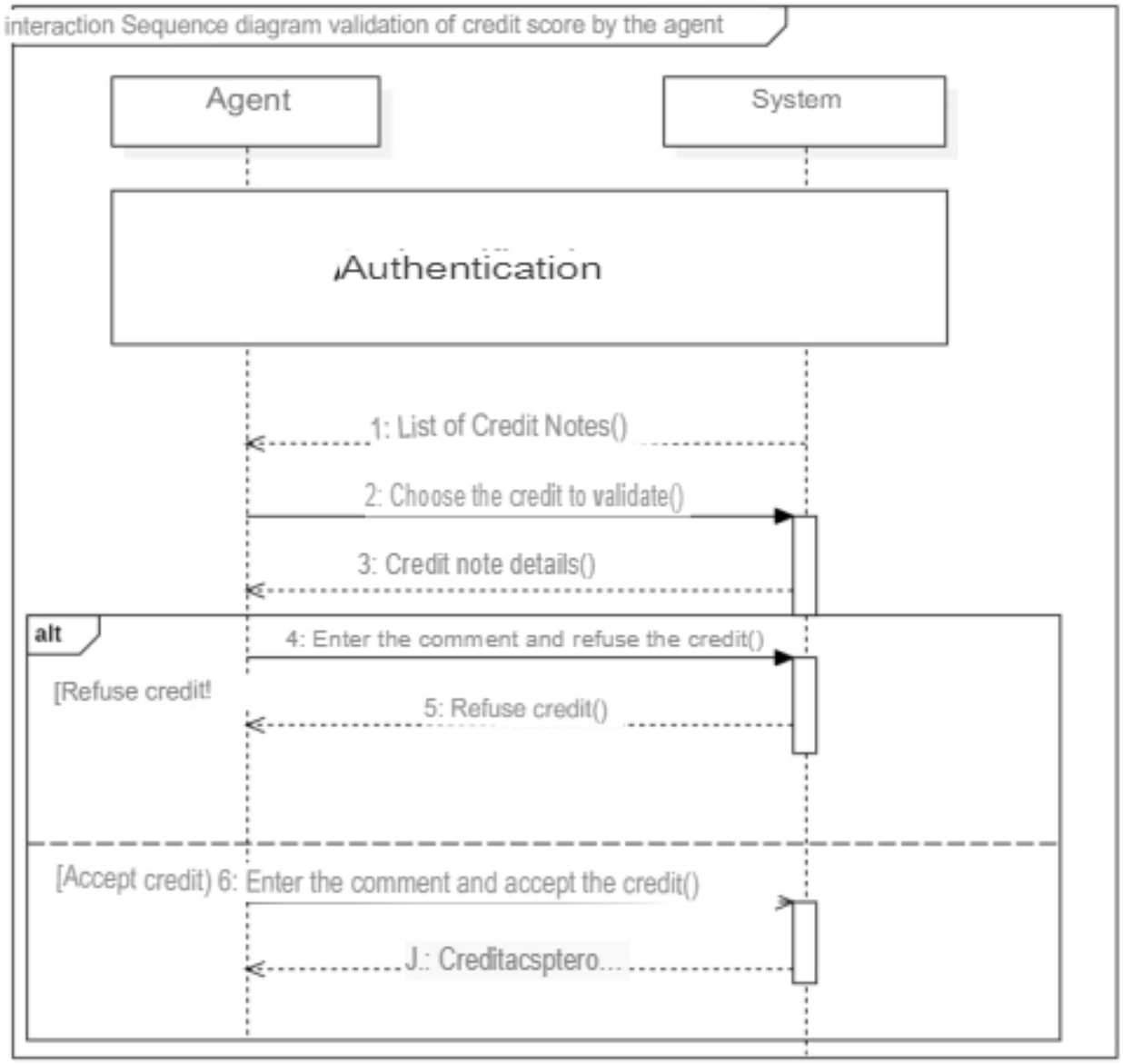


Figure 6.4: Sequence diagram 1

After Validation the admin will wait for the client documents and when the necessary paperwork are sent an automatic credit note will be created and a level from 1 to 4 will be assigned based on the type and quantity of credit like the table 6.1.

Level	Admin type
1	"Chef Agence"
2	"directeur de zone"
3	"comité superieur"
4	"Comité central"

Table 6.1: admin levels

The credit grade assigned determines how the credit application is validated. At level 1, only the "chef agence" can validate the request.

At level 2, the "chef agence" must first validate the request before the "directeur de zone" validates the request.

For Level 3, it must first be passed to the "chef agence " then approved by the "directeur de zone" before being passed to the "comité superieur".

If level 4 validation is required, it will be validated at four levels, starting from level 1 to level 4. The client is instructed to send it after the request has been approved.The list of required documents is sent by the client. After sending them, the attribute pending contract is added to the request's status.the admin will send a contract that the client will retrieve and need to sign to finish the process. This entire process is automated by a A Camunda workflow as you can see in the following figure

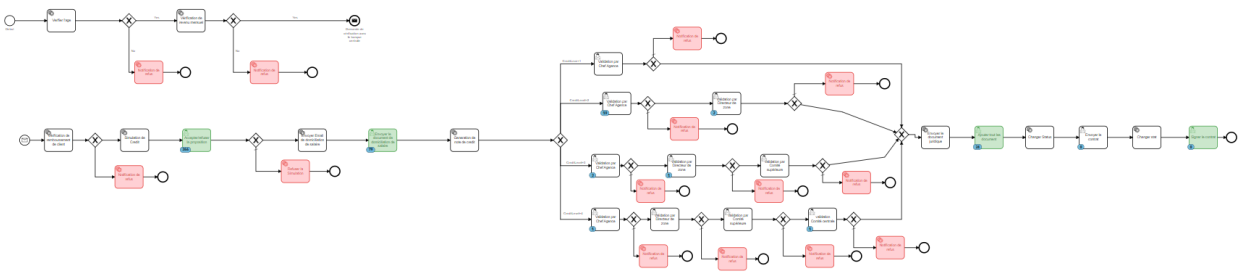


Figure 6.5: Camunda workflow

6.3 Dashboard

6.3.1 Use Case diagram

The following figures shows the Dashboard use case diagram

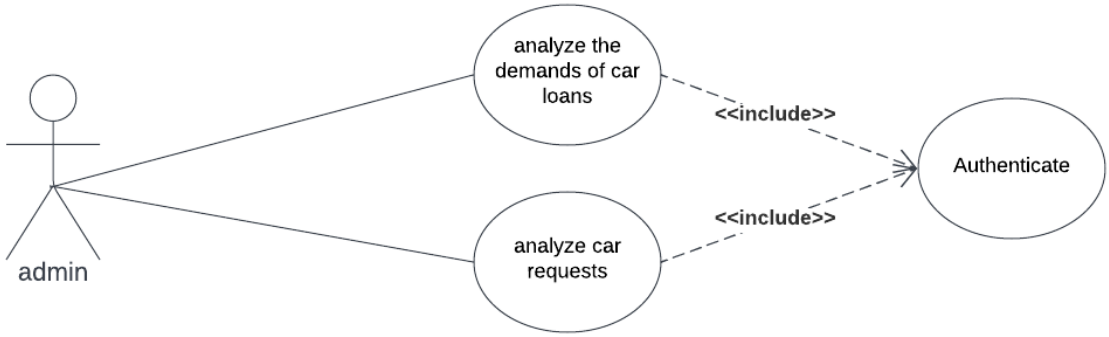


Figure 6.6: Dashboard Use case Diagram

6.3.2 Sequence diagram

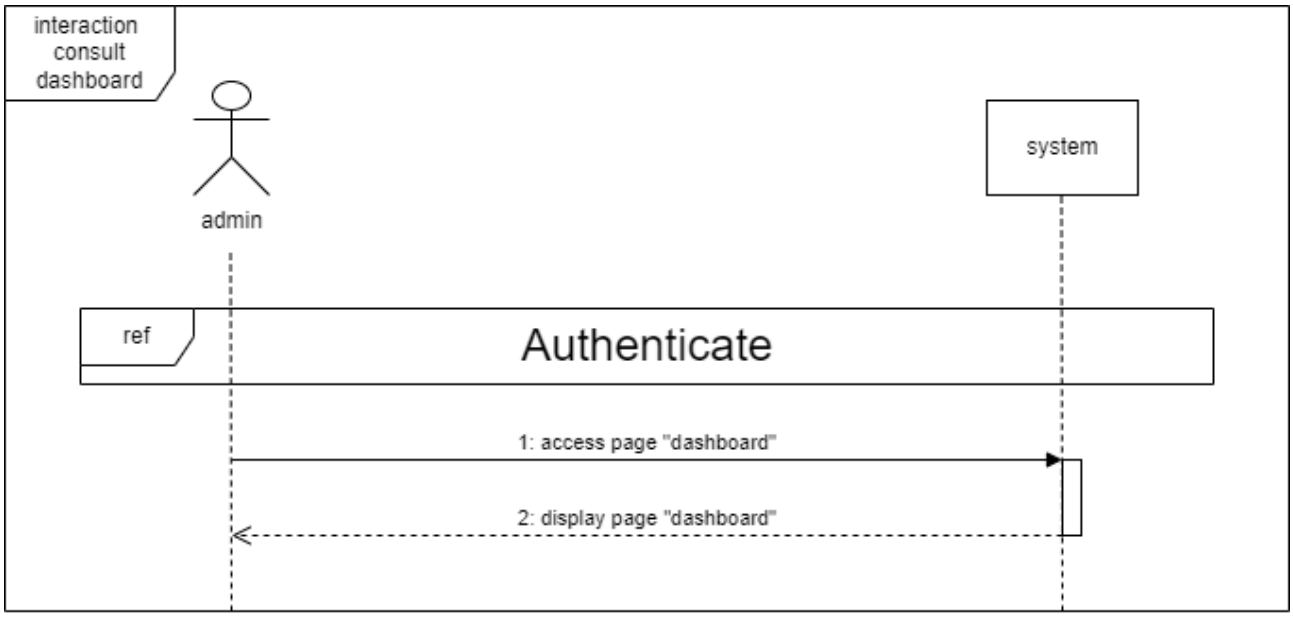


Figure 6.7: Dashboard Sequence Diagram

6.4 Implementation

6.4.1 Credit management

After authentication the admin can access the unprocessed credits list by clicking on "credits à traiter". The first thing he can do is checking the unprocessed credits list then click on "Détails" to consult the credit.

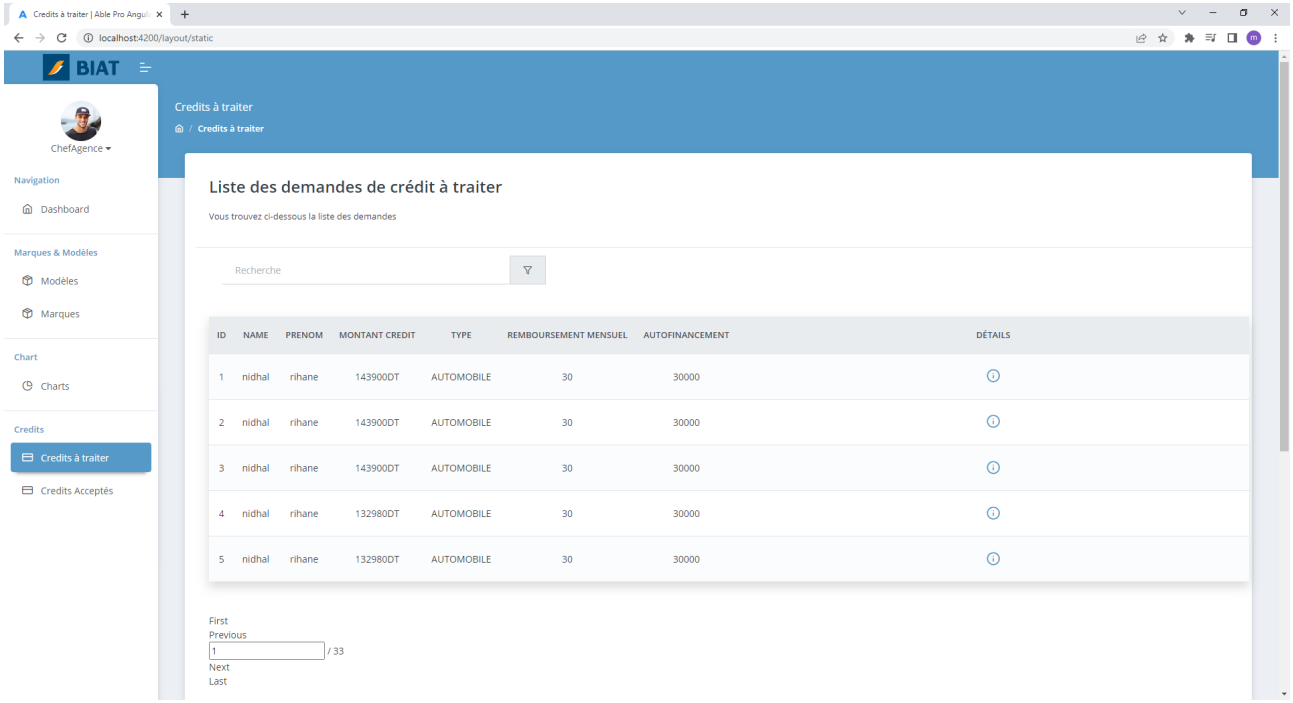


Figure 6.8: Unprocessed credits list

The admin can also check the already processed credits and make modifications on them by clicking on "Modifier".

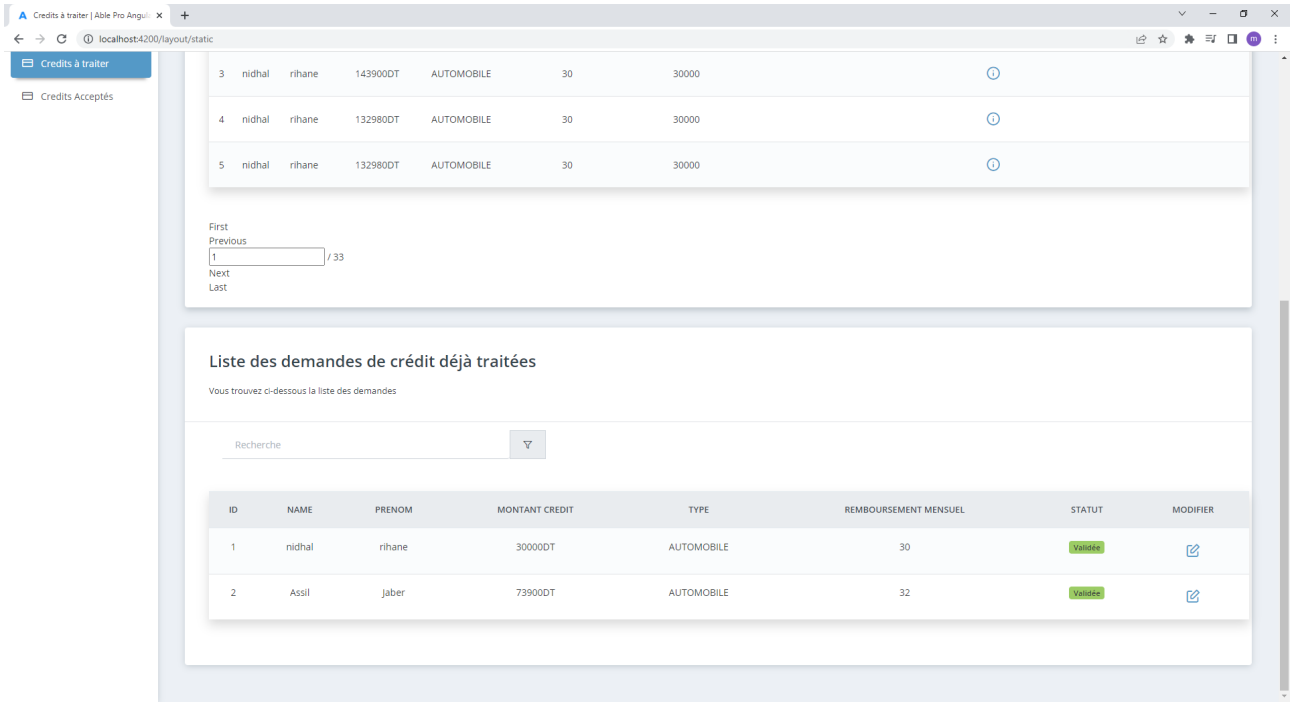


Figure 6.9: Processed credits list

This credit needs two levels of processing: The first one is done by the Agency

Manager. The following figure shows the credit notes which contains the credit's information. The Agency Manager needs to click "Consulter document Domiciliation de Salaire" and verify the legibility of the document as in figure 6.11. Then, he writes his credit's review and chooses either to validate or refuse the credit. Finally, he confirms his review by clicking on "Traiter Demande" which will passes the process to the second level.

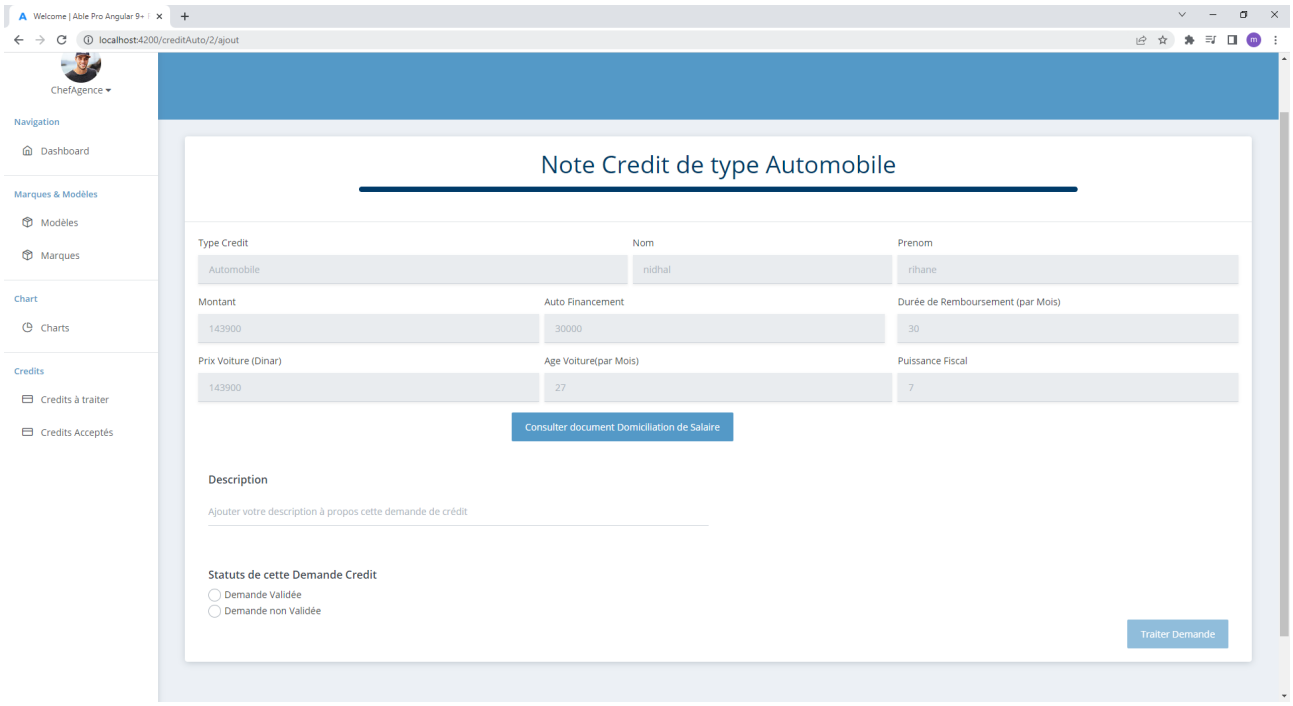


Figure 6.10: Note Credit step 2

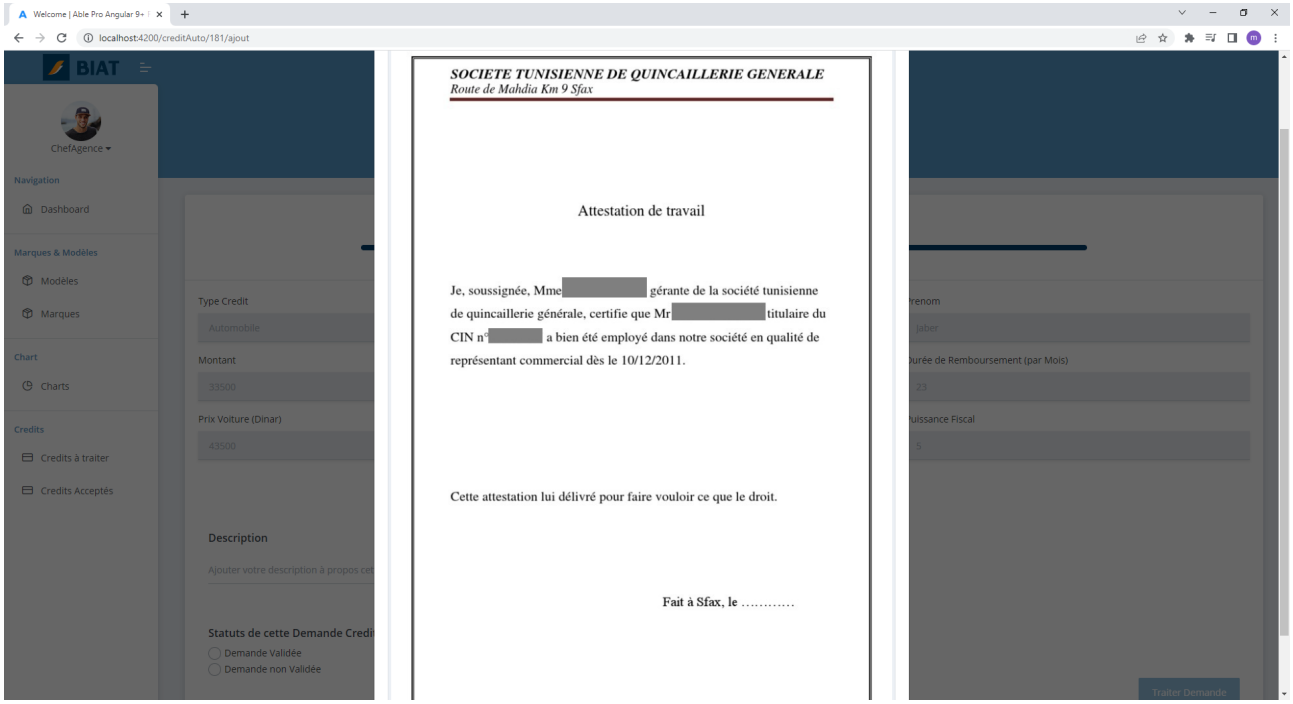


Figure 6.11: Note Credit step 2

The second level is done by the Area Manager, he consults the document too and sees the the Agency Manager review.

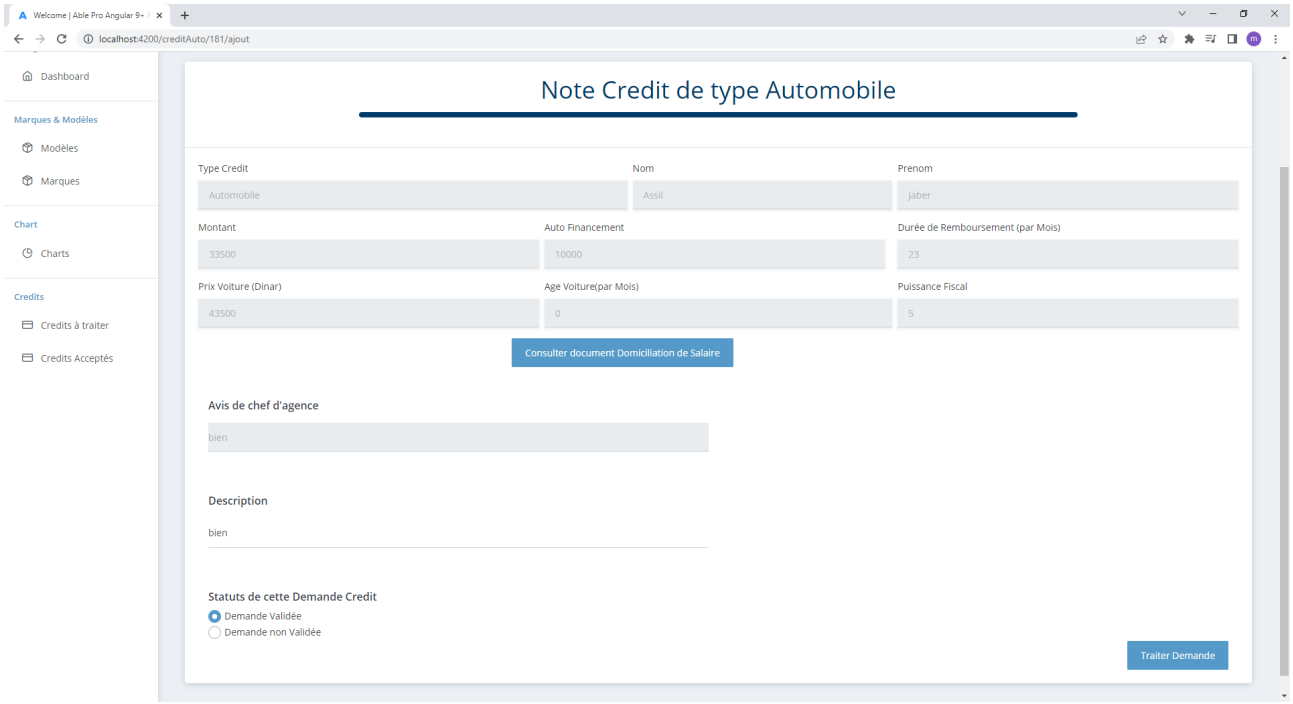


Figure 6.12: Note Credit step 3

After the validation of the Area Manager, the credit will be added to the List

of accepted demands. The admin will need to wait for the necessary documents sent by the client.

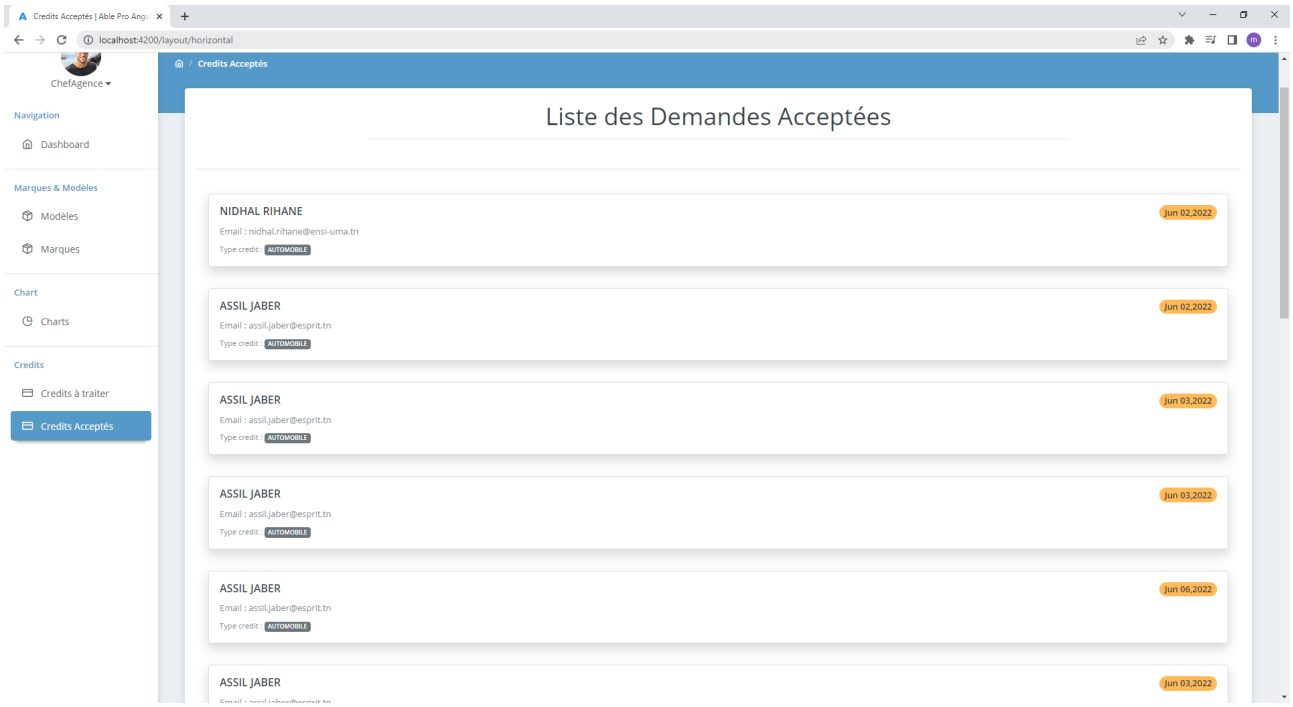


Figure 6.13: Accepted credit

Once the document is sent correctly, the status of the doucment will change from red to green and the admin can consult the document by clicking on "Consulter" as we can see in the figure 6.15.

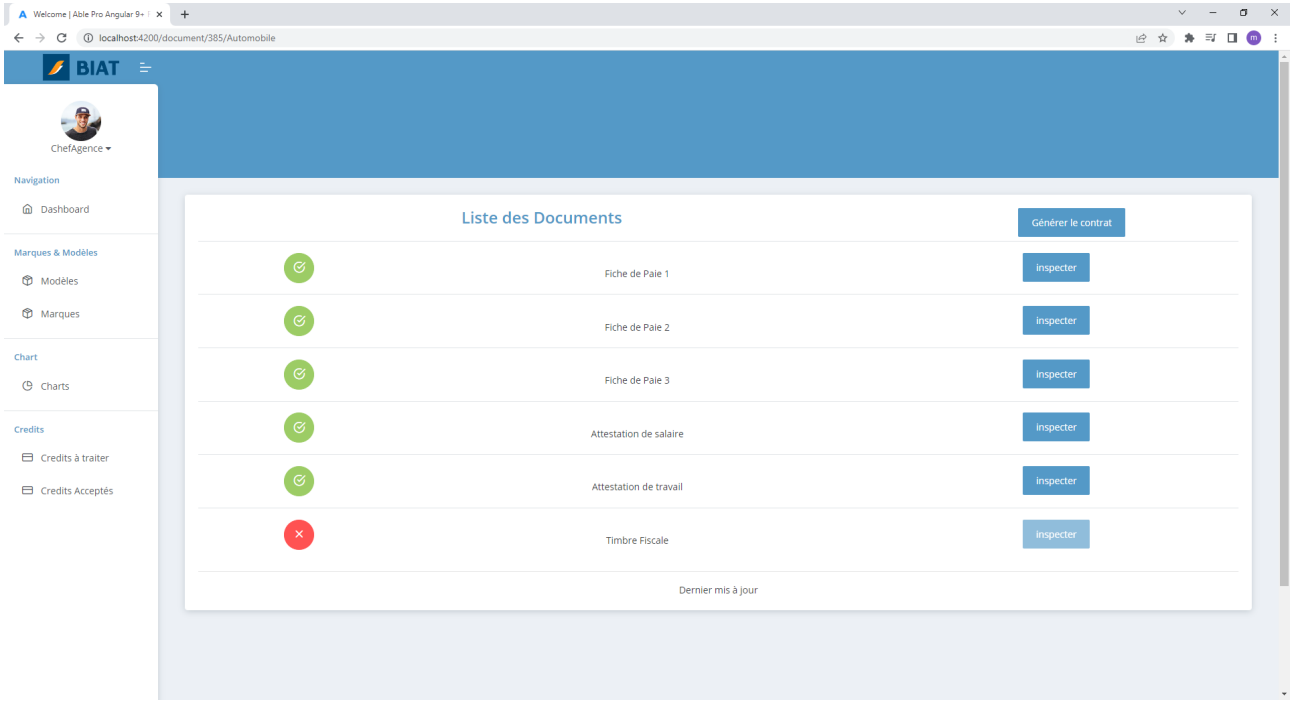


Figure 6.14: Credit document list

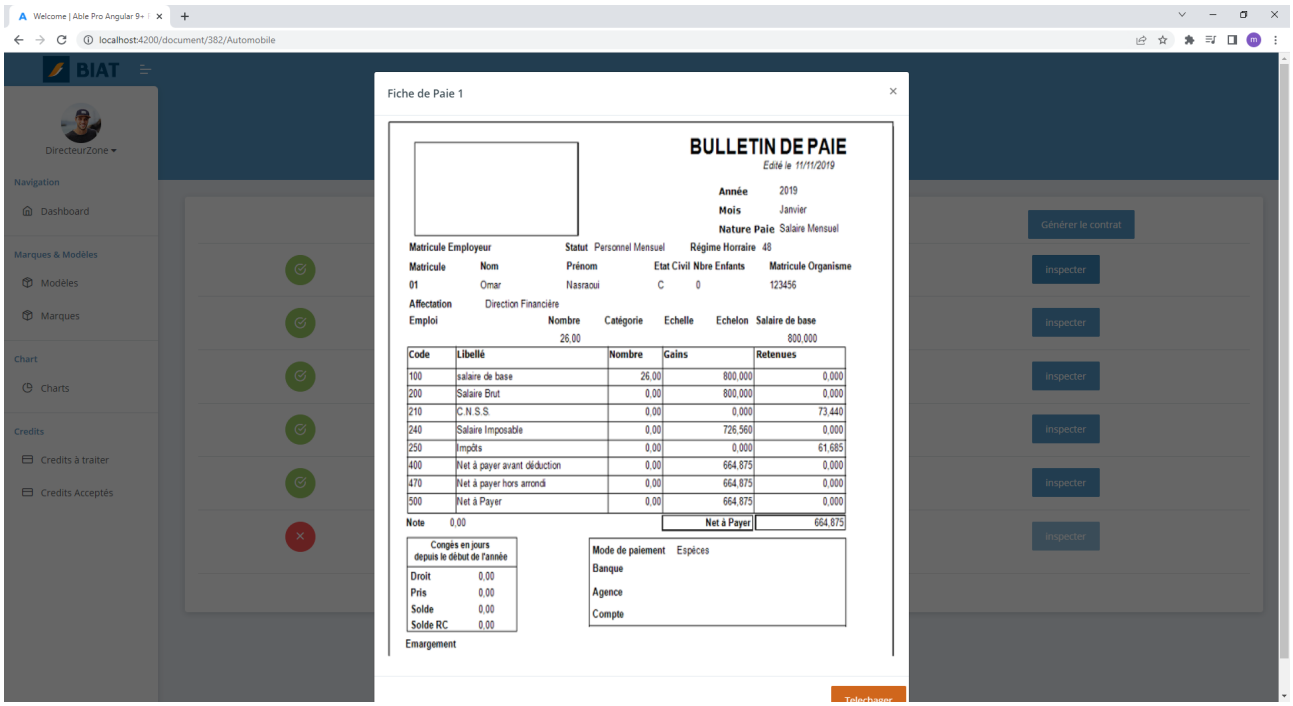


Figure 6.15: Credit document list

Finally, if all the documents are legible and valid, the admin can generate the contract and send it to the client by clicking on "Envoyer".

6.4.2 Dashboard

After the authentication the admin will be redirected to the dashboard page when he can consult the credit process statistics and the car demands. As we can see the following figure

Figure 6.16: Dashboard

Conclusion

In this chapter, we have presented the details of the credits management and the dashboard. In the next chapter, we will look at the deployment and the azure cloud

Chapter 7

Sprint 5: Deployment

Plan

7.1 Deployment tools 69

7.1.1 Azure cloud 69

7.1.2 Docker 69

7.2 Deployment process 70

7.2.1 WSO2 API Manager deployment 70

7.2.2 Microservices deployment 71

7.3 Implementation 71

7.3.1 WOS2 API Manager 71

7.3.2 microservices deployment 74

Conclusion and perspectives 78

Introduction

In this chapter we will be going over the deployment process, the Azure cloud,docker and Kubernetes.

7.1 Deployment tools

7.1.1 Azure cloud

7.1.1.1 Virtual machines

Virtual machines are standalone computers running Linux or Windows that are stored in the cloud and may be accessed by users through an SSH connection. Clients can access public resources using the IP address that a VM exposes and makes available to them.

7.1.1.2 Kubernetes

Azure Kubernetes Service (AKS) offers the quickest way to start developing and deploying cloud-native apps, with built-in code-to-cloud pipelines and guardrails. Get unified management and governance for on-premises, edge, and multicloud Kubernetes clusters. Interoperate with Azure security, identity, cost management, and migration services.

7.1.2 Docker

Docker is a software platform that makes it easier to create, operate, manage, and distribute applications. there is two main terminology which are :

- **Docker Image :** It is a template that includes the application and every dependency needed to run it on Docker.
- **Docker Container :** It is a running instance of the Docker Image

We used docker to create images of the microservices and then deployed them using Kubernetes

7.2 Deployment process

7.2.1 WSO2 API Manager deployment

As a first step,we started with deployment of WOS2 API Manager which has many deployment patterns so we choose the the default pattern as we can see the following figure 7.1

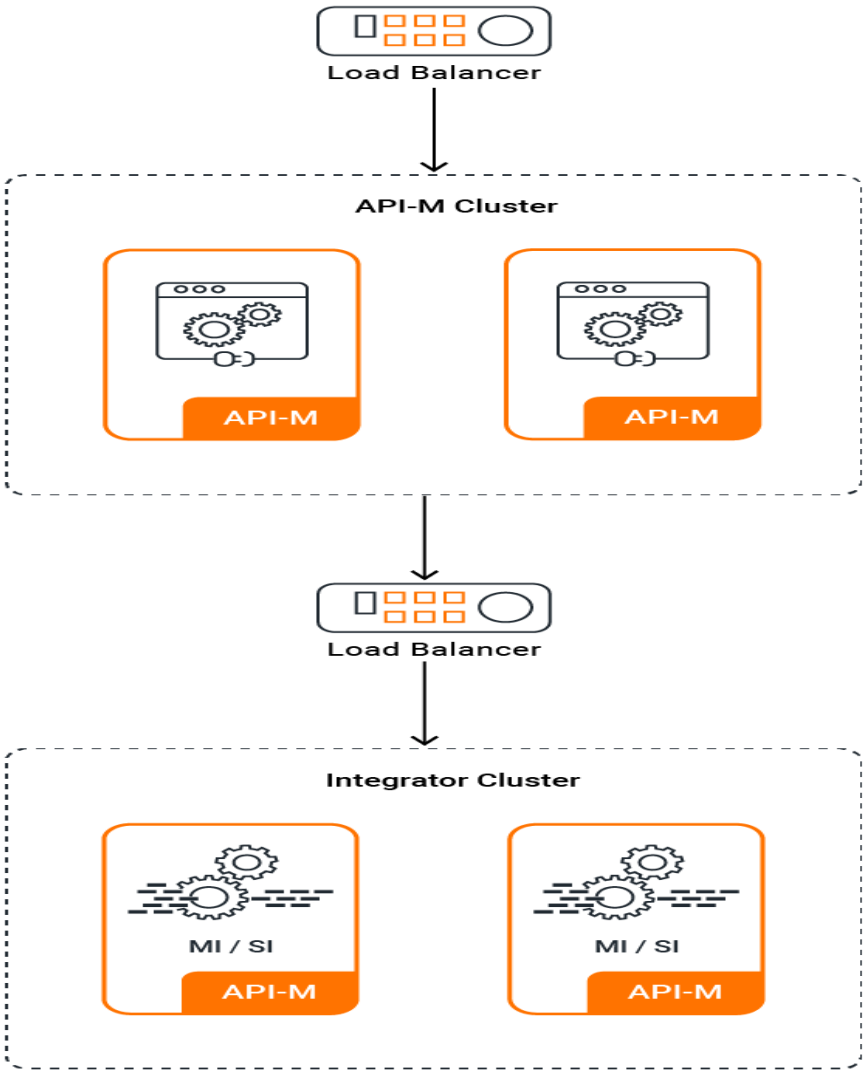


Figure 7.1: Deployment pattern

This deployment consists of an API-M cluster with two nodes of the API-M

runtime and two nodes each of the integration runtimes (Micro Integrator/Streaming Integrator).

7.2.1.1 API-M cluster

The API-M cluster consists of two All-in-One API-M nodes. See the following link for instructions on how to set up this cluster.

7.2.1.2 Integration clusters

The integration cluster may be a Micro Integrator cluster or a Streaming Integrator cluster or two clusters of each. See the following links for instructions on how to set up this cluster.

7.2.2 Microservices deployment

the deployment of miscroservices have multiple steps which are :

- Creation of Spring Boot jar for microservice.
- Creation of a Docker image with the application jar.
- Defining an Kubernetes deployment
- Defining a service that defines rules to access pods
- Defining an ingress that maps a web request

‘

7.3 Implementation

7.3.1 WOS2 API Manager

the deployemnt of WOS2 Api manager is automated with a Helm chart repository which need 4 main steps :

- Installing the Helm Chart
- Obtain the external IP : with the command "kubectl get ing -n wso2"
- Adding a DNS record mapping the hostnames and the external IP

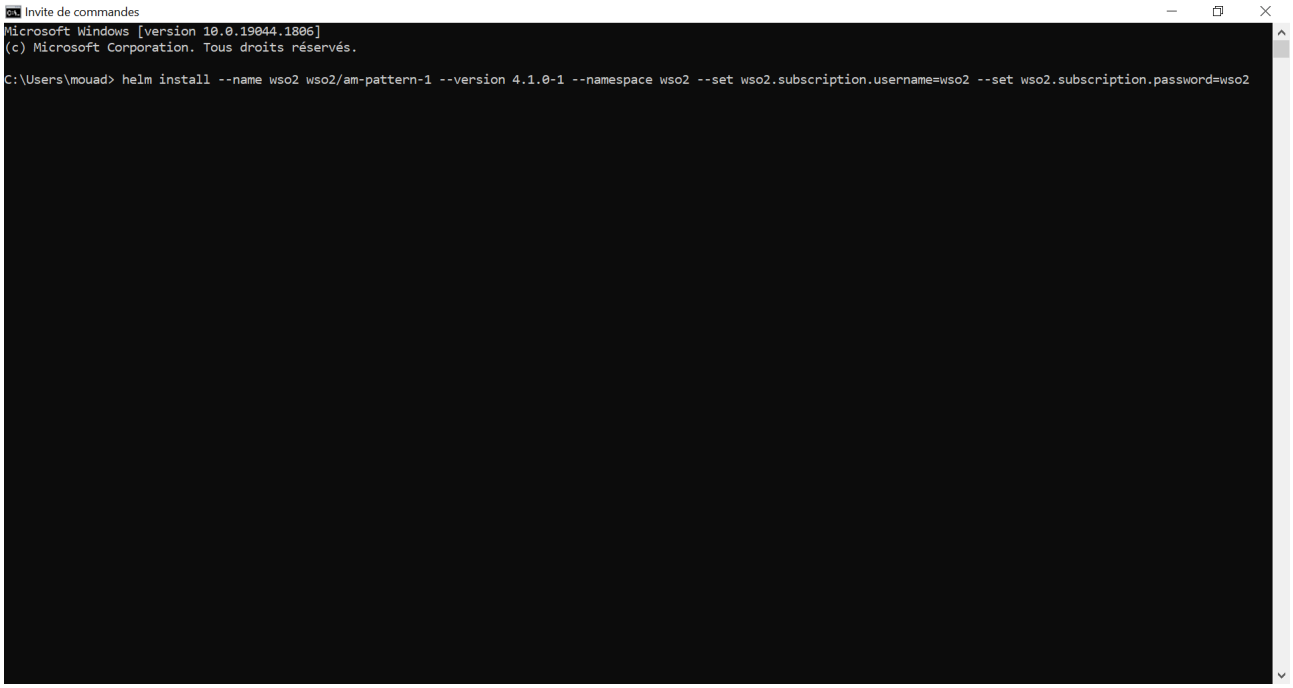


Figure 7.2: Helm installing

```
wso2s-MacBook-Air:scripts wso2$ ./deploy.sh --h
-e ERROR: unknown parameter "--h"
-e This script automates the installation of WSO2 EI Integrator Analytics Kubernetes resources\n
-e Allowed arguments:\n
-e -h | --help
-e --wu | --wso2-username\t\tYour WSO2 username
-e --wp | --wso2-password\t\tYour WSO2 password
-e --cap | --cluster-admin-password\tKubernetes cluster admin password\n\n
wso2s-MacBook-Air:scripts wso2$ ./deploy.sh --wu=$username --wp=$password --cap=6ilqXX7hDbHltbeG
namespace "wso2" created
serviceaccount "wso2svc-account" created
Context "gke_apim-dogfooding-221708_us-central1-c_demo-standard-cluster-1" modified.
secret "wso2creds" created
role.rbac.authorization.k8s.io "endpoints-reader-role" created
rolebinding.rbac.authorization.k8s.io "wso2-endpoints-reader-role-binding" created
-e Creating ConfigMaps...
configmap "apim-conf" created
configmap "apim-conf-datasources" created
configmap "apim-analytics-conf-worker" created
configmap "mysql-dbscripts" created
-e Deploying WSO2 API Manager Databases...
persistentvolumeclaim "wso2apim-with-analytics-rdbms-volume-claim" created
persistentvolume "wso2apim-with-analytics-mysql-pv" created
deployment.apps "wso2apim-with-analytics-mysql-deployment" created
service "wso2apim-with-analytics-rdbms-service" created
-e Deploying persistent storage resources...
persistentvolume "wso2apim-with-analytics-shared-deployment-pv" created
-e Deploying WSO2 API Manager Analytics...
deployment.extensions "wso2apim-with-analytics-apim-analytics-deployment" created
service "wso2apim-with-analytics-apim-analytics-service" created
-e Deploying WSO2 API Manager...
persistentvolumeclaim "wso2apim-with-analytics-apim-deployment-volume-claim" created
deployment.extensions "wso2apim-with-analytics-apim" created
service "wso2apim-with-analytics-apim-service" created
-e Deploying Ingresses...
ingress.extensions "wso2apim-with-analytics-apim-ingress" created
ingress.extensions "wso2apim-with-analytics-gateway-ingress" created
-e Finished
wso2s-MacBook-Air:scripts wso2$
```

Figure 7.3: Ingress deploying

```
-e Deploying NGINX Ingress Controller...
namespace "ingress-nginx" created
configmap "nginx-configuration" created
configmap "tcp-services" created
configmap "udp-services" created
serviceaccount "nginx-ingress-serviceaccount" created
clusterrole.rbac.authorization.k8s.io "nginx-ingress-clusterrole" created
role.rbac.authorization.k8s.io "nginx-ingress-role" created
rolebinding.rbac.authorization.k8s.io "nginx-ingress-role-nisa-binding" created
clusterrolebinding.rbac.authorization.k8s.io "nginx-ingress-clusterrole-nisa-binding" created
deployment.apps "nginx-ingress-controller" created
service "ingress-nginx" created
-e Finished
wso2s-MacBook-Air:nginx wso2$ kubectl get pods -n wso2
NAME                                                    READY     STATUS    RESTARTS   AGE
wso2apim-with-analytics-apim-646d544f6b-6qmr8           0/1       Running   0          1m
wso2apim-with-analytics-apim-analytics-deployment-5685d45fmdhfv   1/1       Running   0          2m
wso2apim-with-analytics-mysql-deployment-549cf67756-dm778         1/1       Running   0          2m
wso2s-MacBook-Air:nginx wso2$ kubectl get ing -n wso2
NAME                                HOSTS               ADDRESS   PORTS     AGE
wso2apim-with-analytics-apim-ingress      wso2apim                      80, 443   2m
wso2apim-with-analytics-gateway-ingress   wso2apim-gateway              80, 443   2m
wso2s-MacBook-Air:nginx wso2$ kubectl get ing -n wso2 --watch
NAME                                HOSTS               ADDRESS   PORTS     AGE
wso2apim-with-analytics-apim-ingress      wso2apim                      80, 443   2m
wso2apim-with-analytics-gateway-ingress   wso2apim-gateway              80, 443   2m
wso2apim-with-analytics-apim-ingress   wso2apim   34.66.123.80   80, 443   2m
wso2apim-with-analytics-gateway-ingress   wso2apim-gateway   34.66.123.80   80, 443   2m
```

Figure 7.4: wso2 deployment finished

7.3.2 microservices deployment

We start the deployment process by building the project then creating a Docker-File as in he figure 7.5

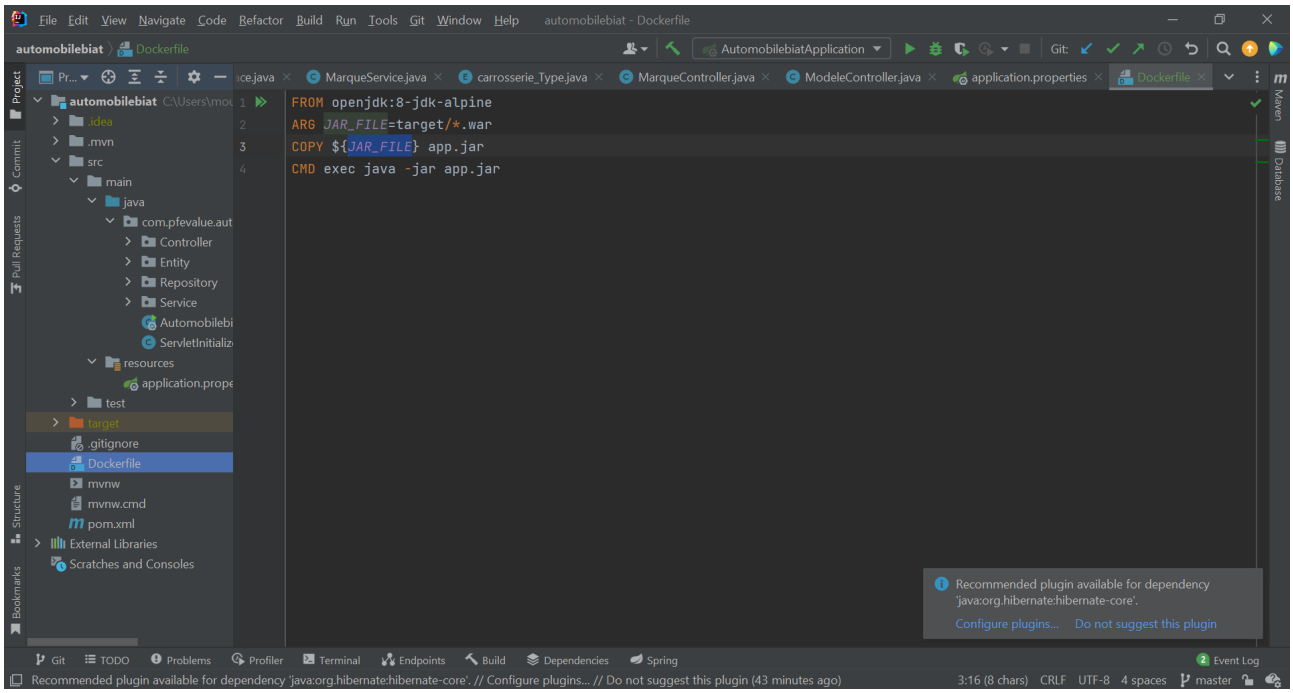


Figure 7.5: Creation of Spring Boot jar

Then we create a docker image with the command "docker build <name< .

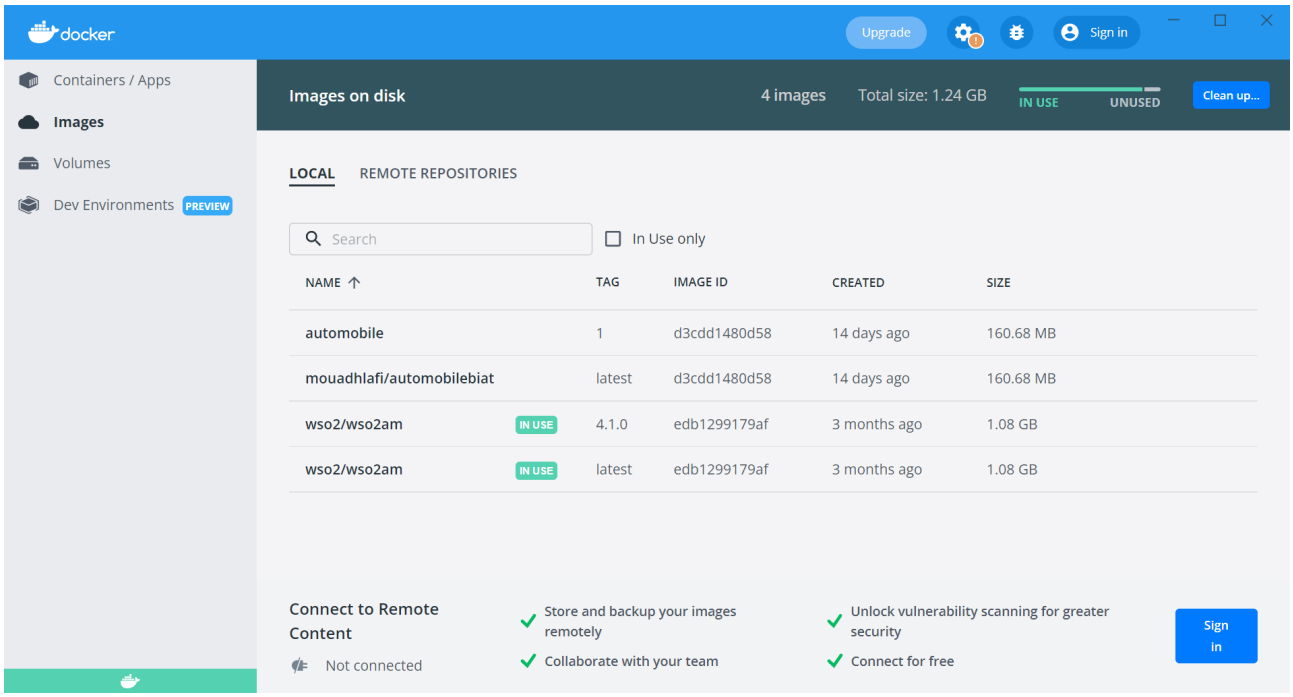


Figure 7.6: Docker Image

then next step is to give the image a tag with the command "docker tag". Once the image is tagged, we can push the image with "docker push"

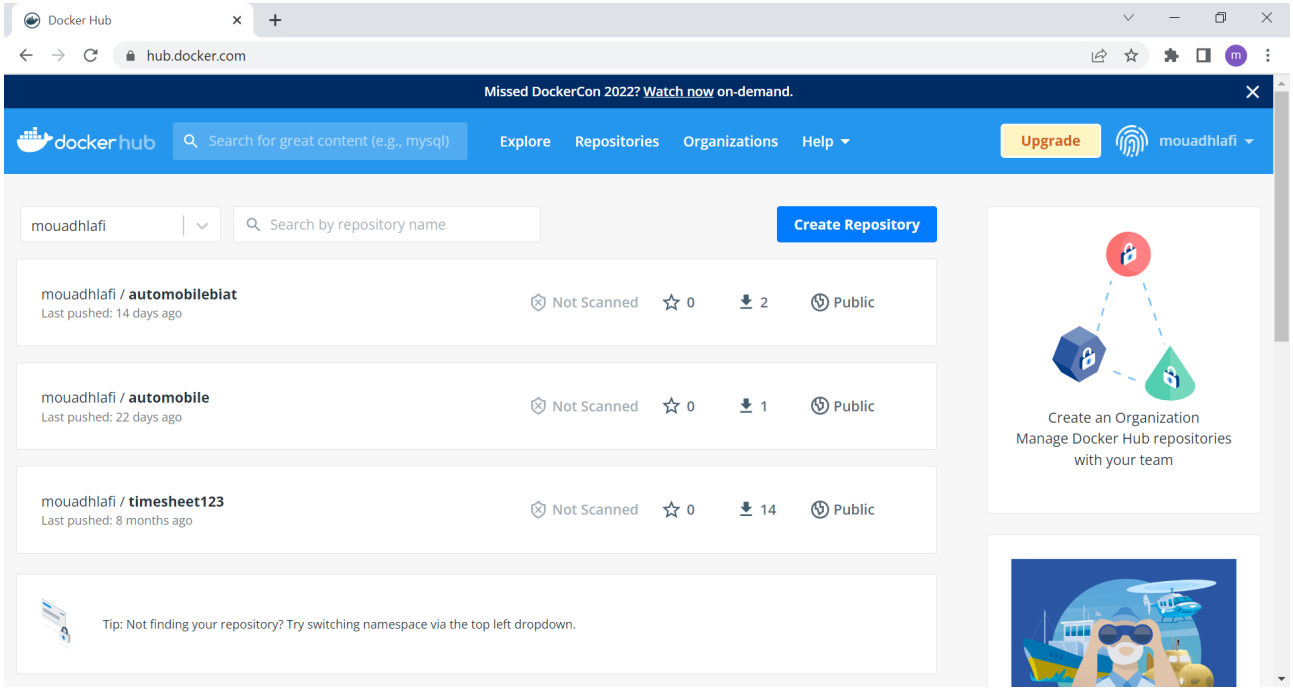


Figure 7.7: Docker Hub

In the next step we will need a Kubernetes Cluster which is in our case AKC(Azure Kubernetes Cluster)

After publishing the Docker image, you can run the image as a pod on Kubernetes. Next we will create a deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: automobile-biat
  labels:
    app: automobile-biat
spec:
  replicas: 3
  selector:
    matchLabels:
      app: automobile-biat
      template:
    metadata:
      labels:
        app: automobile-biat
    spec:
      containers:
        - name: automobile-biat
          imagePullPolicy: IfNotPresent
          image: mouadhlafi/automobilbiat-svc:1.0.0
```

Figure 7.8: Deployment File

The above command creates a Kubernetes deployment with three replicas of the pod that matches the labels app: automobile-biat. After the above command is run, Kubernetes pulls the Docker image mouadhlafi/automobilebiat-svc:1.0.0 from the DockerHub repository and creates pods.

Then we will need a ClusterIP service to exposes service on the internal IP of the cluster. We can define a Kubernetes service for the automobilbiat microservices as:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: automobile-biat
spec:
  type: ClusterIP
  selector:
    app: automobile-biat
  ports:
    - port: 80
      targetPort: 8080
```

Figure 7.9: service File

The targetPort is the port exposed by the automobile-biat microservice Docker container, the port is the port of service itself.

To create a Kubernetes service, copy the above definition in service.yaml and run command kubectl apply -f service.yaml.

Finally we will need an Ingress that can exposes HTTP(S) routes, such as /api, from outside the cluster to services within the cluster.

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: app-ingress
spec:
  rules:
    - http:
        paths:
          - path: /api
            pathType: Prefix
            backend:
              service:
                name: automobile-biat
                port:
                  number: 80
```

Figure 7.10: Ingress File

To create ingress, copy the above ingress definition in ingress.yaml and run command kubectl apply -f ingress.yaml

Conclusion

With this chapter, we have completed the fifth and final sprint of our project. This sprint has allowed us to focus on the deployment and azure cloud.

General conclusion and perspectives

Our project was to design and develop a car dealer application while using the wso2 api manager as api centralizer.
The solution supposed to guarantee an easy and simple application to buy cars with credits also it will assure a permanent and fluid communication between the different actors around a credit application but mainly the discovery of wso2 api manager and all it's features.

We have succeeded in developing :

- A web/mobile application to start cars credit and continue the credit process.
- A web application for the admin side to manage brands,models and credits with a handful dashboard.
- Discover all the features of Wso2 Api manager and integrate them to the project

The realization of this project went through three main phases. First the study of the WSO2 API manager during a 4 weeks, next was the development of the microservices and the application as all like what's in the implementation of our Sprints. Lastly was the deployment phase using the Azure architecture.
During the process we encountered different obstacles, The most remarkable ones were related to the WSO2 Api manager, Which is pretty understandable due the lack of sufficient documentation about it also the integration with the other modules of the project is a difficult step due to the high dependency.
Finally, we can say that our project may experience future modifications and enhancements that will increase its usefulness and enhance the user experience.

References

[1] WSO2 https://apim.docs.wso2.com/en/latest/.

[2] WS02-2 https://apim.docs.wso2.com/en/3.0.0/getting-started/overview/

[3] API https://www.seeburger.com/info/apis-from-api-management-to-api-solu

[4] Kubernetes https://azure.microsoft.com/en-us/services/kubernetes-service/#features

[5] WSO2 API Manager Deployment https://apim.docs.wso2.com/en/latest/install-and-setup/setup/deployment-overview/

[6] Kubernetes https://techdozo.dev/deploying-a-restful-spring-boot-microser

[7] Gregg Boer, https://docs.microsoft.com/en-us/devops/plan/what-is-scrum.

[8] Automobiletn, https://automobile.tn/.

[9] Medicalchain, https://medicalchain.com/en/. https://web3js.readthedocs.io

[10] Docker, https://docs.docker.com.