Tunisian republic
Ministry of Education
Superior
and Scientific Research

TEK-UP
Private Higher School of Technology
and Engineering

Engineer in:
Computer science

Final Project Report semester

introduced to

TEK-UP University

by

Selim horri
Badrddine Aidoudi
Imen Toukebri

Project Tracker

Mrs Safa SAOUDI Supervisor

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

Companies seek to develop their technological environments in terms of project management in order to facilitate the smooth running of projects during its phases, limit delay problems, follow-up and control the progress of work by employees through planning.

Thus, to successfully carry out our project successfully, it would be necessary to follow an effective methodical approach for our work in order to meet the needs. This report is divided into four main chapters:

- The first chapter is entitled State of the art, gives a presentation of the general framework, the study of the existing and the methodology to be used to carry out our project.

- The second chapter is entitled Analysis and Speci fi cation of Needs, which represents the functional and non-functional needs, as well as the breakdown of our project with the planning of the various activities which are af fi liated to it and the architecture and technologies used.

- The two chapters which follow constitute the body of our report which will be subdivided in two parts Release 1 and, Release 2.
Each of them first presents the organization of the sprints carried out as well as their functional speci fi cation. Then, we will illustrate the use case diagrams as well as their textual description, and we will describe the different scenarios via the sequence diagrams. Finally we will present some interfaces illustrating our realization.

Chapter

1

Chapter 1: General framework from project

Summary

1.1 INTRODUCTION

In this chapter, we are going to set up objectives to be achieved in the application, starting from what already exists.

1.2 Study of the existing

In this part, we will describe one of the software commonly used in the market, then, focus on the problem and the proposed solution.

MS Planner

MS Planner offers an extremely collaborative environment for effective teamwork. When creating a folder (Plan), it is possible to invite other colleagues to participate in order to assign them tasks and deadlines to be respected. Then the team manager can export everyone's tasks and due dates to their respective Outlook calendar.

Different types of permissions can be assigned to colleagues: administrator, guest, etc. When an administrator makes a change to a task or deadline in Planner or Outlook, all of their teammates receive an automated notification in their inbox. At all times, team members have access to the common task calendar.

1.2.1 Problem

As part of our project, we concluded that after discussing with the manager within our department, we found that the latter does not have any tool that allows us to visualize the progress of the various projects to be managed as well. than to ensure the collaboration of employees in assigned tasks.

1.2.2 Planned solution

By ensuring the technological development of our activities within our department and in order to solve the problem mentioned above, we are going to develop an application

web to create a platform dedicated to the employees of our department, and this for the reason of assigning projects to the employees concerned by ensuring their follow-up by managers.

1.3 Development methodology

1.3.1 Scrum

Scrum presents the most popular agile method. It is characterized as incremental, by its development its short iterations and its reduced formalism. This figure on which SCRUM is based: e illustrates the process

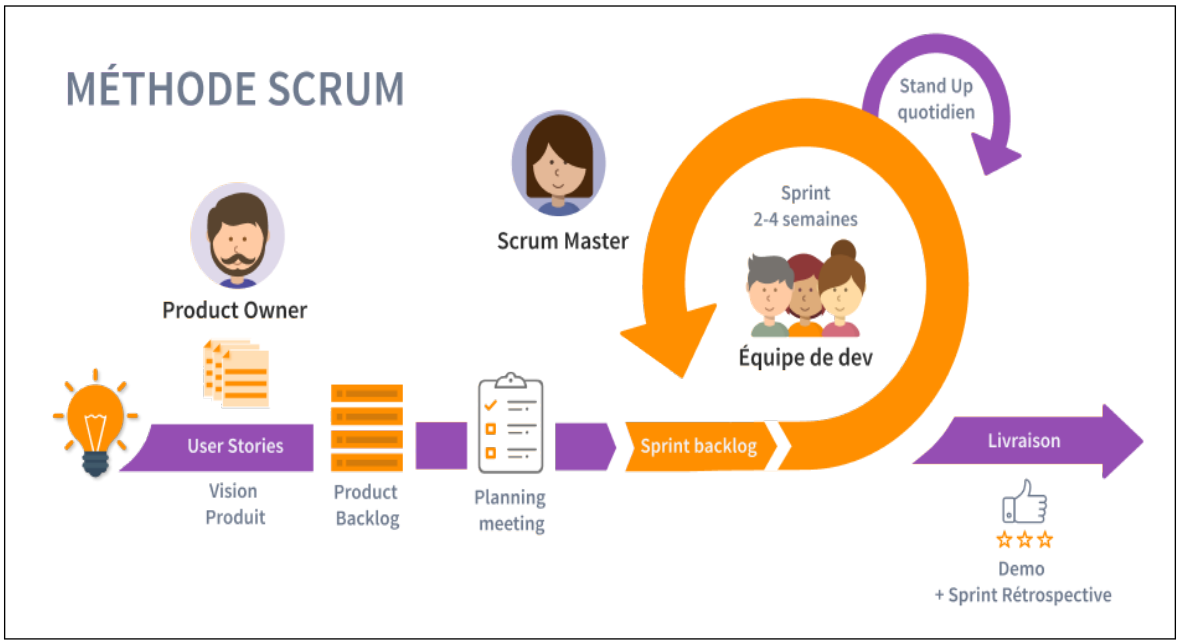

FIGURE 1.1: Scrum process

1.3.2 Roles in scrum

The Scrum method de fi nes only three roles for a business expert a project : Product Owner: one who carries the vision of the product to be achieved. Development team: it is the team responsible for transforming the needs expressed by the Product Owner into usable functionalities. ScrumMaster: who must master Scrum and ensure that it is applied correctly.

1.4 Gantt chart

1.5 Conclusion

In this first chapter, we have described the solutions already existing on the market (MS Planner), presenting the incompatibility of these solutions in our organization. Then, we set out the problematic of our project, while providing criticism and possible solutions. Finally, we explained the choice of the development methodology used and we presented the Gantt chart.

Chapter

2 Analysis and speci fi cation of needs

Summary

2.1 INTRODUCTION

The needs analysis, the first phase of a project, conditions its success insofar as it de fi nes the real needs of those who will use the end result. This is the phase of communication and exchange, it is often the re fl ection of the end result. Suddenly, we will approach the analysis of use cases, the functions offered by the system and the actors with whom it interacts.

2.2 Functional requirements

We will illustrate our functional needs through a list (list of functional needs) and based on a static context diagram and another dynamic with a description of the use cases of each actor in our system. Finally, there will be a good presentation of our general use case diagram and a summary table of the use cases mentioned before. Indeed, the functional needs or the use cases in terms of UML can be de fi ned as follows: “A use case represents a set of sequences of actions carried out by the system and producing an observable result. interesting for a particular actor. "

2.2.1 Static context diagram

This UML diagram is simply used to show the relationship of the different actors with the system. It speci fi es the number of instances of each actor connected to the system at a given time.

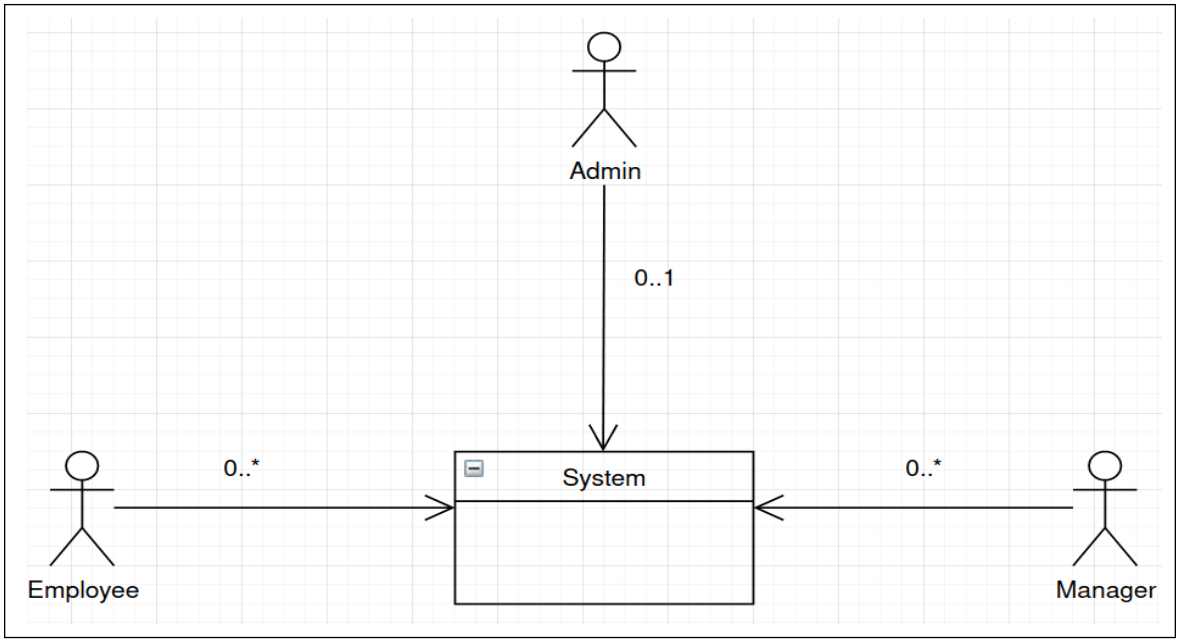

FIGURE 2.1: Static context diagram

2.2.2 Dynamic context diagram

The purpose of the dynamic context diagram is to highlight the roles and responsibilities of each category of actor vis-à-vis the system.

To keep it readable, we cannot detail all the actions of the players linked to the main functionalities of the system.

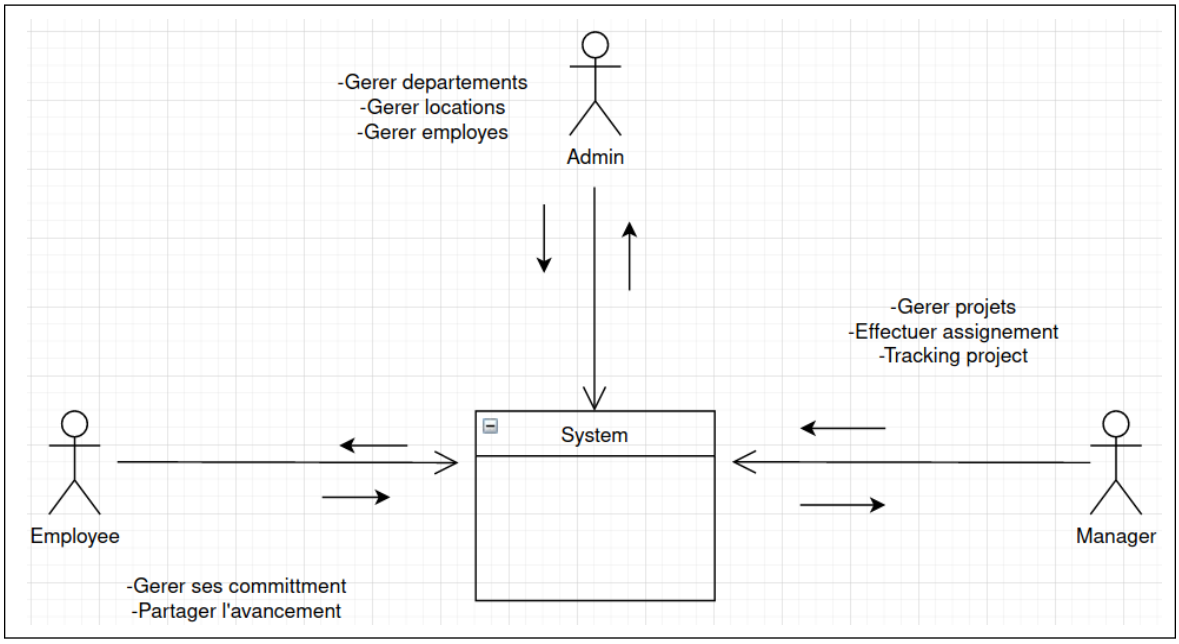

FIGURE 2.2: Dynamic context diagram

2.2.3 Lists of functional requirements

As mentioned before, a use case is a sequence of actions performed by the system in order to respond to a request from a user (ie: actor). In the following, we describe the different functional needs of our system:

1. Employee:

   - Share progress: The employee must share their progress according to a committment date with a description.

   - Consult the affected projects: The employee can list the different projects affected by his manager

   - Consult his committment dates: the employee can view his various committments dates for each project assigned to it.

2. Manager:

   - Consult progress status: The manager must consult the progress status of each project assigned to one or more employees.

   - Consult the assigned projects: The manager can list the different projects assigned to the employees.

   - Ensure follow-up: The manager can ensure the follow-up of each project by adding a comment on the description of the employee.

   - Creation of a project: The manager can create projects.

   - Assign a project: The manager can assign a project to one or more employees.

   - Manage a project: The manager can manage a project / assignment.

3. Admin:

   - Manage rentals: The administrator must manage the rentals of the different departments.

   - Manage departments: The administrator must manage the departments of different employees.

   - Manage employees: The administrator must manage all employees.

2.2.4 Lists of non-functional requirements

Ergonomics and flexibility:

The application must offer a user-friendly and ergonomic interface that can be used by the user by considering all the possible interactions on the screen of the support held.

Speed:

The application must optimize the processing to have a reasonable schema generation time.

Ef fi ciency:

The application must be functional regardless of any circumstances that may surround the user.

2.3 Architecture and technologies

In this part, we describe the architecture of Spring Framework as well as the technologies and the different layers used.

2.3.1 Spring Framework Architecture

Spring is an open source framework for building and de fi ning the infrastructure of a Java application, for which it facilitates development and testing. It provides many functionalities which are sometimes redundant or which can be con fi gured or used in several ways: this leaves the choice to the developer to use the solution which suits him best and / or which meets the needs.

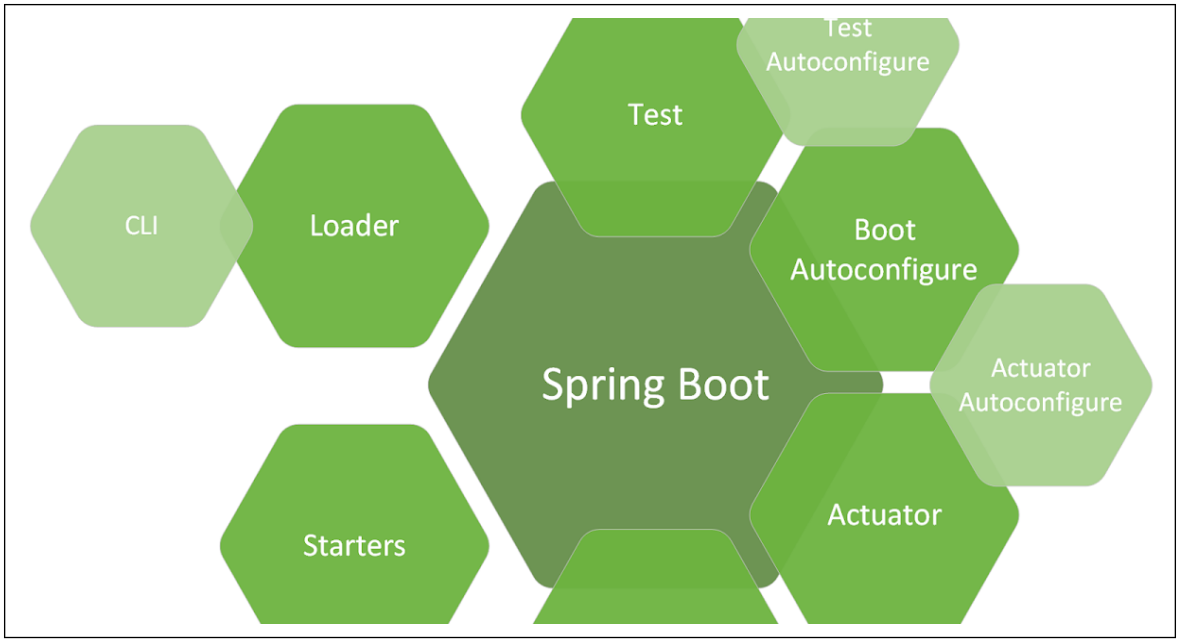

FIGURE 2.3: Spring Modules

Spring Core Framework: Spring Core is based on an IoC type container, ensuring the management of the life cycle of Beans and the injection of dependencies as well as the use of AOP and several other layers.

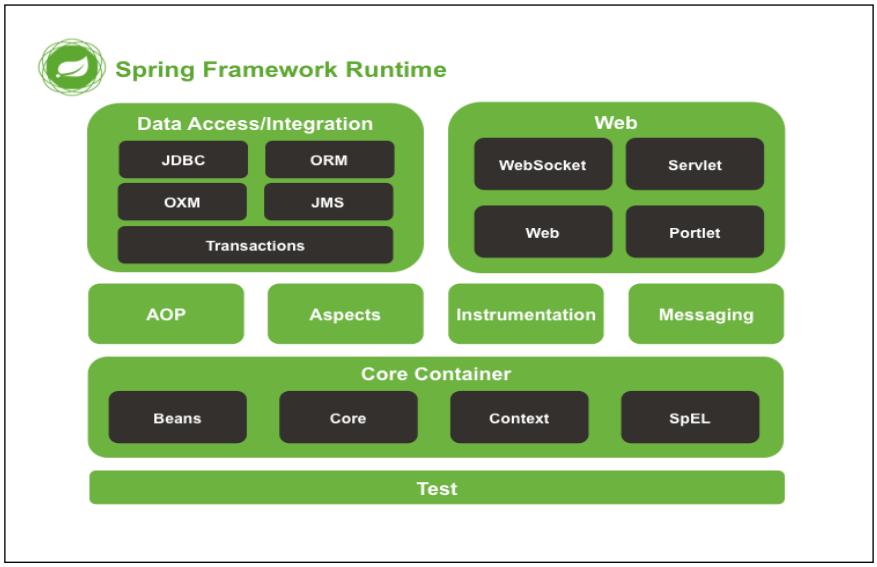

FIGURE 2.4: Spring Core Framework

Spring MVC Architecture

Spring MVC is one of the most widespread Frameworks in the Java world: its popularity has grown thanks to the complexity of Java EE, especially for its versions prior to version 5, but also thanks to the quality and the richness of the functionalities that it offers. he proposes.

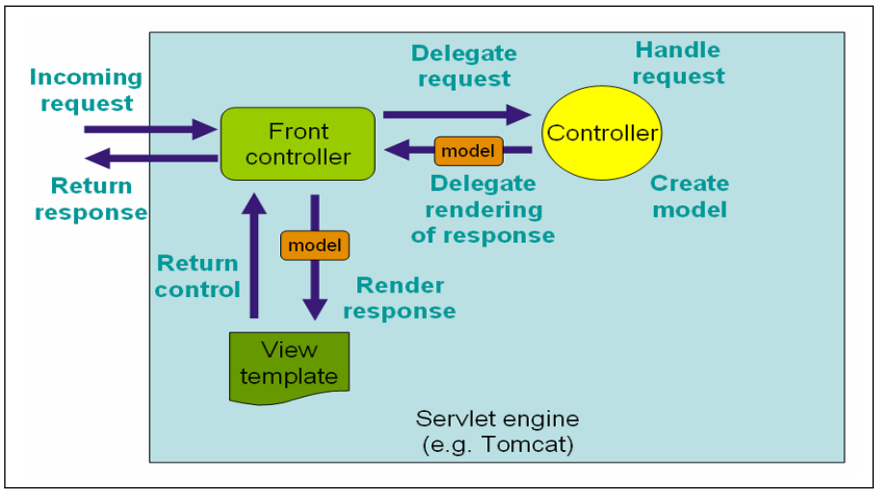

FIGURE 2.5: Spring MVC Architecture

2.3.2 Application architecture

The application to be developed is an MVC web application which will be divided mainly into four Layers:

- Domain Layer.
- Data Access Layer.
- Layer Services.
- Business Layer.

The diagram below will represent the main layers to be developed.

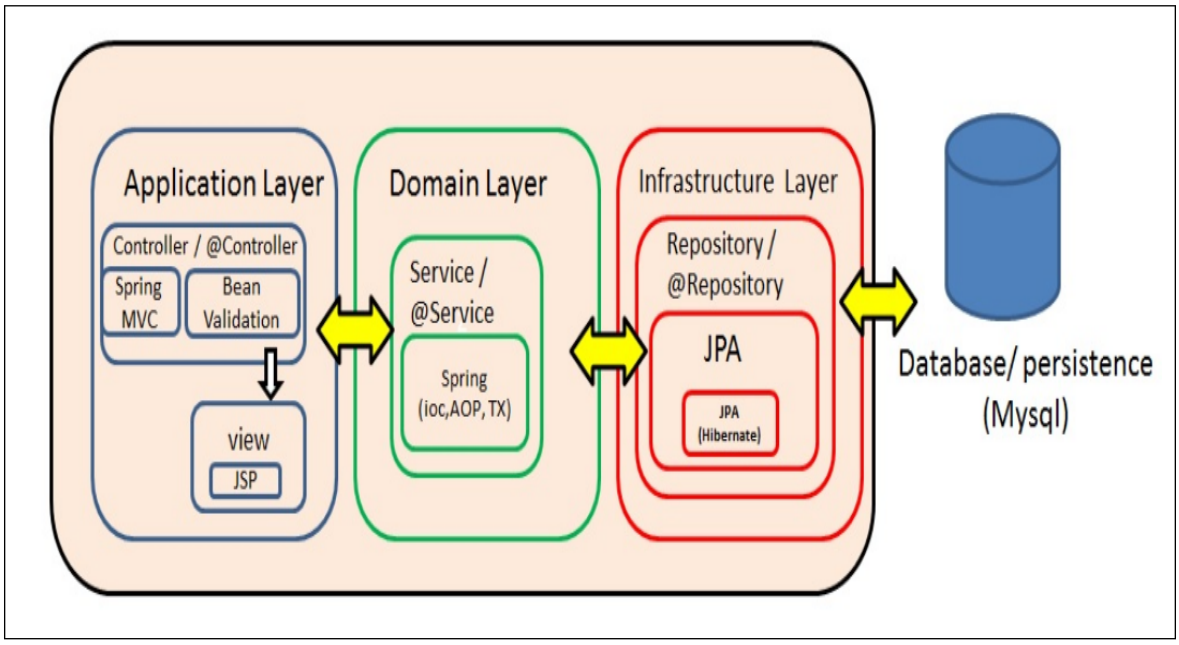

FIGURE 2.6: Spring MVC Architecture

2.4 Work environment

In this part we specify the development environment in which we carried out our project.

2.4.1 Software Environments

- **Java:**

  Our famous Java programming language is an object oriented language created by James Gosling and Patrick Naughton, employees of Sun Microsystems, with the support of Bill Joy (co-founder of Sun Microsystems in 1982), of fi cially presented on May 23, 1995 at SunWorld. The Sun company was then bought in 2009 by the Oracle company which now owns and maintains Java. The peculiarity and central objective of Java is that software written in this language should be very easily portable to several operating systems such as Unix, Windows, Mac OS or GNU / Linux, with little or no modification. For this, various platforms and associated frameworks aim to guide, if not guarantee, this portability of applications developed in Java.

FIGURE 2.7: Java logo

- **Maven:**

  Maven is a tool for automating the management of Java projects. It offers between other the following features:

  Compilation and deployment of Java applications (JAR, WAR). Management of libraries required by the application.

  Execution of unit tests.

  Generation of project documentation (website, pdf, Latex). Integration into different IDEs (Eclipse, JBulder).

FIGURE 2.8: Apache Maven logo

- **Apache Tomcat:**

  Apache Tomcat is a free web container for servlets and JSP. Coming from the Jakarta project, it is one of the many projects of the Apache Software Foundation. It implements the Java Community Process servlet and JSP speci fi cations, is configurable through XML fi les and properties, and includes tools for con fi guration and management. It also has an HTTP server.

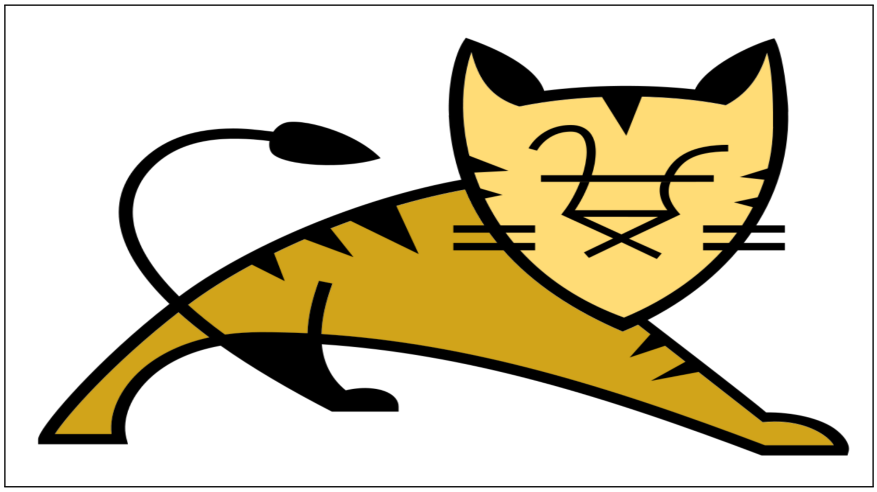

FIGURE 2.9: Apache Tomcat logo

- **Spring Boot:**

  Spring Boot is a JAVA development framework. It is a variation of the classic Spring framework which essentially allows the creation of micro services

FIGURE 2.10: Spring Boot logo

- **Thymeleaf:**

  Thymeleaf is a template engine, licensed under Apache 2.0, written in Java that can generate XML / XHTML / HTML5. Thymeleaf can be used in an environment

web or not web. Its main purpose is to be used in a web environment for view generation for web applications based on the MVC model.

FIGURE 2.11: Thymeleaf logo

- **Docker:**

  Docker is free software for launching applications in software containers. According to industry research firm 451 Research, “Docker is a tool that can package an application and its dependencies in an isolated container, which can be run on any server.

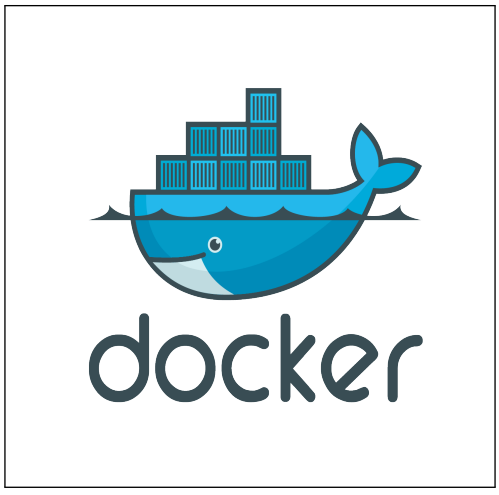

FIGURE 2.12: Docker logo

- **Postman:**

  Postman allows you to build and execute HTTP requests, to store them in a history in order to be able to replay them, but above all to organize them in Collections. This classi fi cation makes it possible in particular to group requests together in a “functional” manner (for example a chain of adding an item to the basket, or else an identification process).

FIGURE 2.13: Postman logo

- **Swagger:**

  Swagger is an open source project launched by a Startup in 2010. The objective is to set up a Framework that will allow developers to document and design APIs, while maintaining synchronization with the code.

FIGURE 2.14: Swagger logo

- **JUnit:**

  JUnit is a unit testing framework for the Java programming language. Created by Kent Beck and Erich Gamma, JUnit is arguably the most successful project in the xUnit series. JUnit de fi nes two types of test fi les

FIGURE 2.15: JUnit logo

- **Mysql:**

  MySQL is a relational database management system (RDBMS). It is distributed under a dual GPL and proprietary license. It is one of the most widely used database management software in the world4, both by the general public (mainly web applications) and by professionals, in competition with Oracle, PostgreSQL and Microsoft SQL Server.

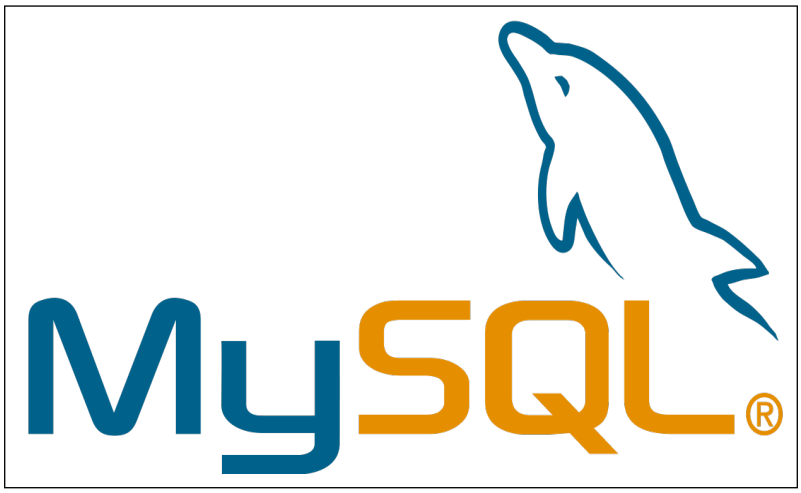

FIGURE 2.16: Mysql logo

- **H2:**

  H2 is a relational database management system written in Java. It can be integrated into a Java application or operate in client-server mode. it supports a subset of the SQL standard. It offers SQL and JDBC programming interfaces (APIs); however it can also use the PostgreSQL ODBC driver and behave as a PostgreSQL server. Tables can be created in RAM or on disk. They can be persistent or temporary. Indexes are hash tables and trees for in-memory tables, and B trees for on-disk tables8. All data manipulation is transactional. Table-level locking as well as multiversion concurrency is implemented.

FIGURE 2.17: Logo of H2

- **Git:**

  Git is decentralized version management software. It is free software created by Linus Torvalds, author of the Linux kernel, and distributed under the terms of the GNU General Public License version 2. As of 2016, it is the most popular version control software that is used by over twelve million people. Similar in this to GitLab, Git does not rely on a centralized server, but uses a peer-to-peer connection system.

FIGURE 2.18: Git logo

- **Spring Tool Suite:**

  Spring has its own Eclipse plugin called STS (Spring Tool Suite). This allows Spring to be efficiently integrated into the development environment.

FIGURE 2.19: STS logo

- **Heroku:**

  Heroku is a company creating server software that enables the deployment of web applications.

FIGURE 2.20: Heroku logo

- **Latex:**

  Is a description language giving users the means of word processing. It allows to design professional quality documents without knowledge of their form.

FIGURE 2.21: Latex

2.5 Project management with scrum

For the management of our project, we will follow the agile SCRUM method. In this part, we identify the scrum team, then we present the product bachklog as well as sprint planning.

2.5.1 Identification of the SCRUM team

In our project, the roles are organized as follows:

Product owner: Ms. Safa Saoudi

Scrum master: Selim HORRI, Badreddine AIDOUDI, Imen TOUKEBRI

Scrum team: Selim HORRI, Badreddine AIDOUDI, Imen TOUKEBRI

2.6 Product backlog

The functionalities to be developed are described in the form of user stories1 in the product backlog which contains all the technical needs. Table 2.3 shows our prioritized product backlog, in the stories are de fi ned by the MOSCOW priority such as:

- M: Must
- S: Should
- C: Could
- W: Won't

Task	Role	Priority
The administrator must manage the rentals of the different departments	Administrator	M
The administrator must manage the departments of different employees	Administrator	S
The administrator must manage all employees	Administrator	S
The employee must share their progress according to a committment date with a description	Employee	S
The employee can list the different projects affected by his manager	Employee	VS

the employee can view his various committments dates for each project assigned to it.	Employee	VS
The manager must consult the progress of each project assigned to one or more employees.	Manager	M
The manager can list the different projects assigned to employees	Manager	M
The manager can create projects	Employee	M
The manager can assign a project to one or more employees.	Manager	VS
The manager can manage a project / assignment	Manager	VS

2.6.1 General Use Case Diagram

In this part, we present the overall use case diagram. This last one is the first UML diagram that captures the behavior of a system. It describes the operation of the system in coherent units which are the use cases. These make it possible to express the needs of the users of a system.

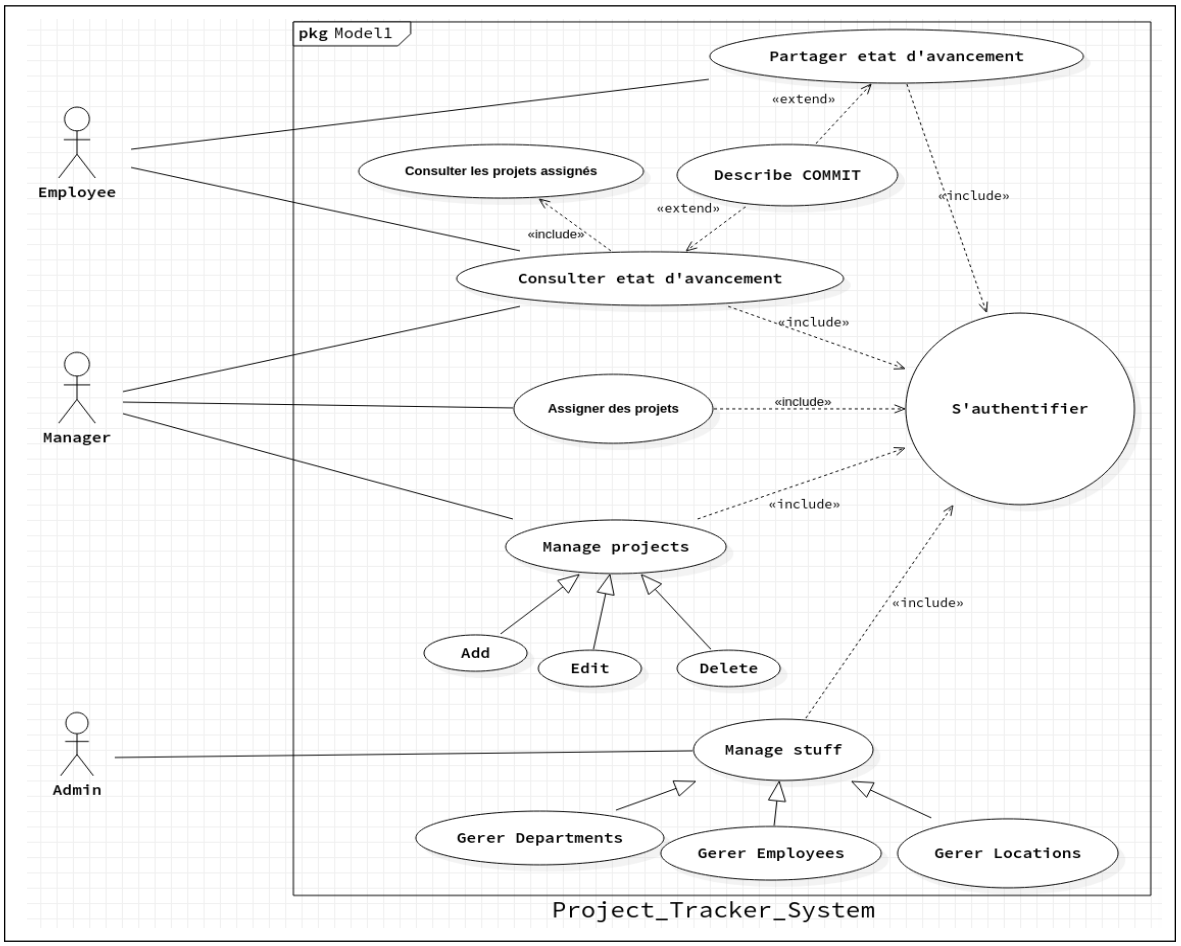

FIGURE 2.22: General use case diagram

2.6.2 Class diagram

The class diagram is a diagram used in software engineering to present the classes and interfaces of systems as well as the different relationships between them. This diagram is part of the static part of UML because it disregards the temporal and dynamic aspects.

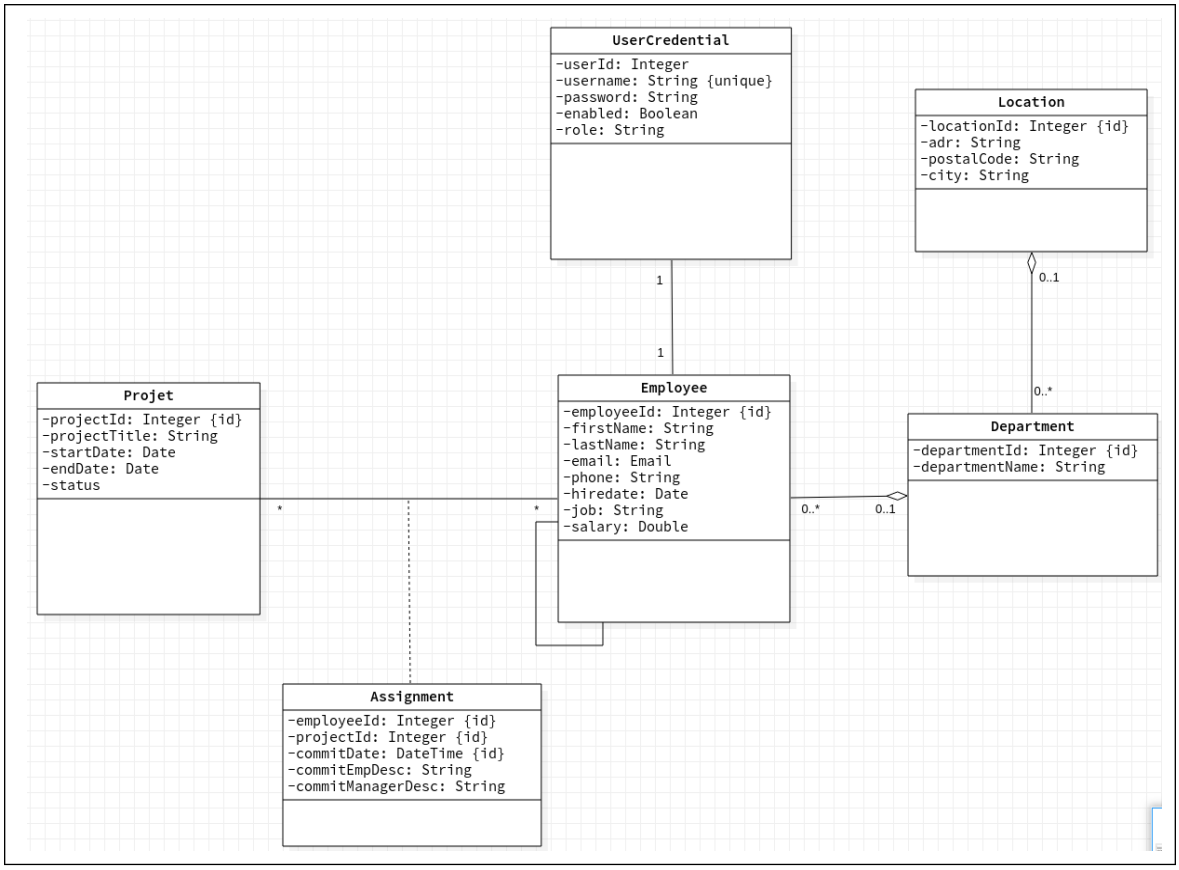

FIGURE 2.23: Class diagram

2.6.3 ERD diagram

Entity relationship diagram, also known as ERD, ER diagram, or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two

important information: the main entities within the scope of the system and between these the relationships entities.

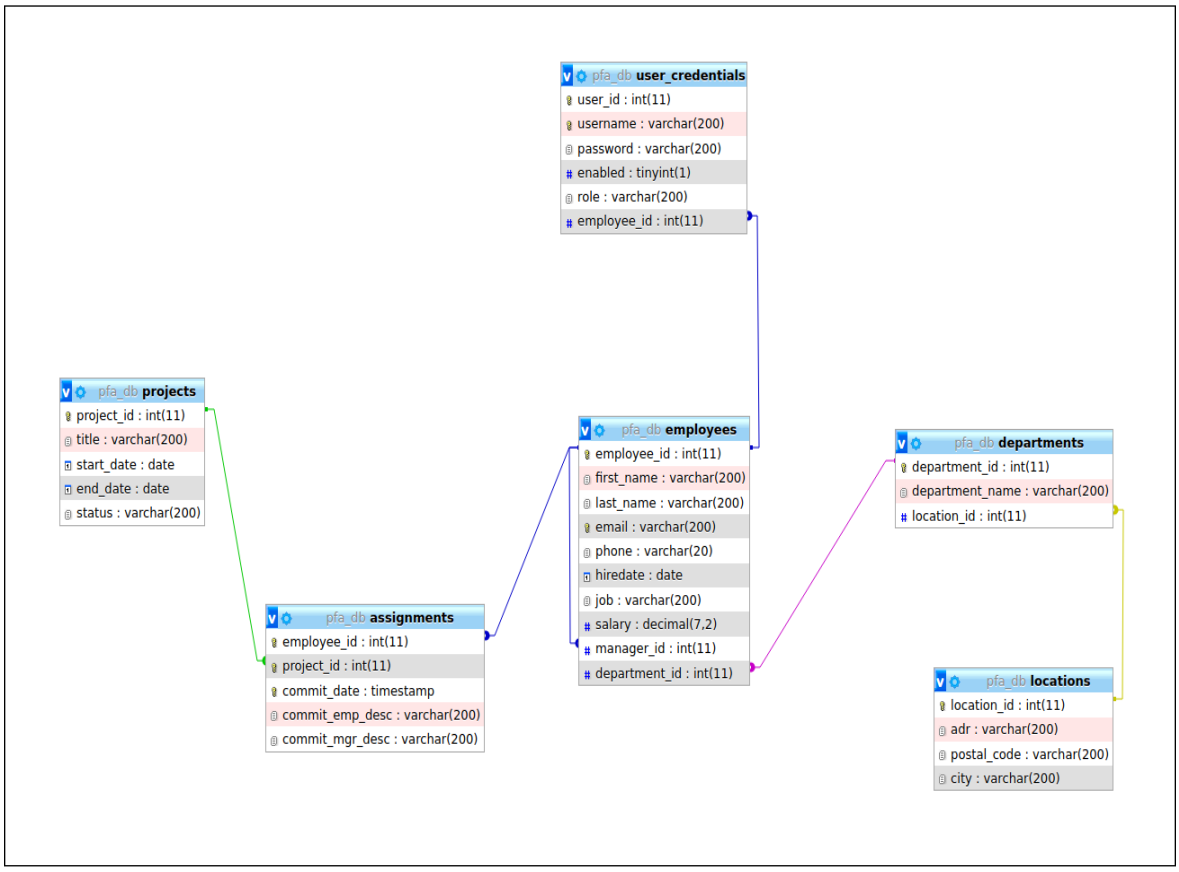

FIGURE 2.24: ERD diagram

Conclusion

Throughout this chapter, we have speci fi ed functional requirements and non-functional requirements. Then, we presented the architecture of the system as well as the technologies used for the realization of the project. Then, we prepared our work plan during which we identified the project team, our backlog and the division of our release into sprints. Finally, we presented the general use case diagram and the global class diagram. In the next chapter, we continue with the presentation of our first release.

Chapter

3

Chapter 3: THE FIRST RELEASE

Summary

3.1 INTRODUCTION

In this chapter, we deal with the user stories from the first release. First, we present all the speci fi cations, then we present the design phases of its sprints as well as the interfaces relating to the realization of the first release.

3.1.1 Sprint backlog:

To fully understand the features of the first release, we reserve this part to present the backlog of the sprints of the latter. The first release is made up of two sprints. The first sprint focuses on the
consultation of all the projects assigned to this employee.

Task	Role	Priority
The employee must to authenticate	Employee	M
The employee can list the different affected projects by his manager	Employee	VS

TABLE 3.1: Backlog sprint 1

The second sprint focuses on the tasks concerning the sharing of progress and the visualization of the various commits.

Task	Role	Priority
The employee must share your state advancement following a date of committment with a description	Employee	S
the employee can visualize his different committment dates for each project that assigned to it.	Employee	VS

TABLE 3.2: Backlog sprint 2

3.2 Analysis and Design of Sprint 1:

In this part, we establish the refinements of the different use cases of the first sprint and some design diagrams in order to deliver a description on the different possible scenarios.

3.2.1 Analysis of the case "The employee must authenticate himself":

The system ensures the identity of the user through his login and password.

3.2.2 Refined Employee Authentication Use Case Diagram must

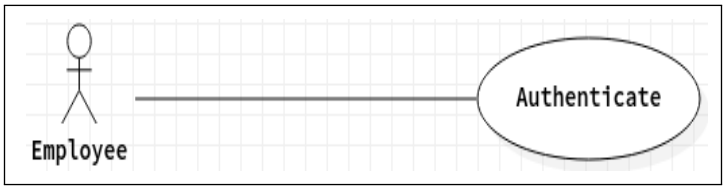

FIGURE 3.1: Refined diagram of use cases: The employee must authenticate himself

3.2.3 Text description

In this part we present the textual description which contains the use case scenario "Employee must authenticate".

SOMMAIRE D'Authentification	
Titre	Authentification
But	Le système s'assure de l'identité de l'employé à travers son login et son mot de passe.
Acteur	Employer
DESCRIPTION DES ENCHAINEMENTS	
Pré conditions	
Chaque employer doit avoir un compte L'employé accède à l'application	
Scénario nominal	
L'authentification : 1. L'employé clique sur « login» 2. Le système afficher la page d'accueil	
Enchainements alternatifs	
1. Les champs login et mot de passe sont non valides et/ou vides a. Le système affiche un message d'erreur.	

FIGURE 3.2: Text description of the "Authentication" scenario.

3.2.4 Sequence diagram of the scenario "The employee must to authenticate "

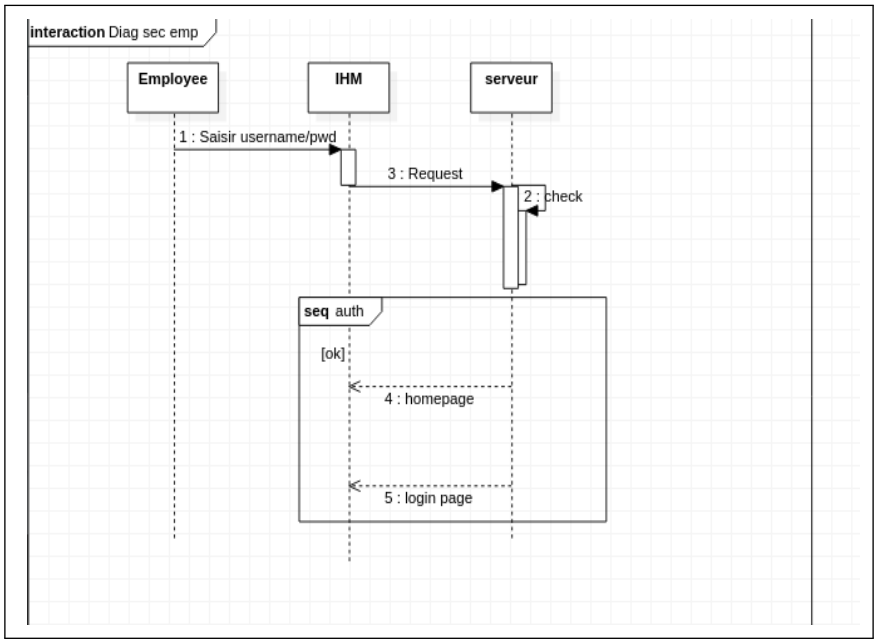

FIGURE 3.3: Sequence diagram of the scenario "The employee must authenticate himself"

3.2.5 Analysis of the case "The employee can list the different projects affected by his manager":

The employee can list the different projects affected by his management

3.2.6 Re fi ned diagram of use cases "The employee can list the different projects affected by his management"

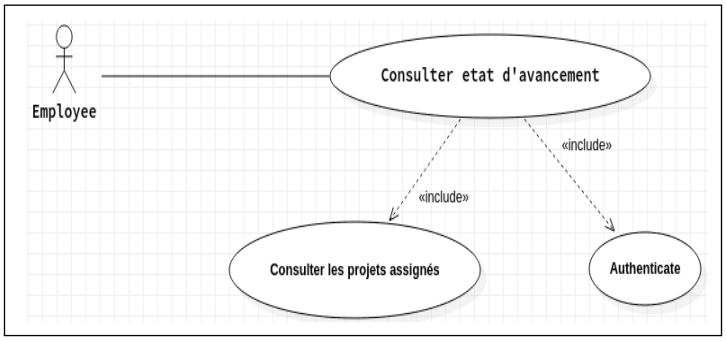

FIGURE 3.4: Refined diagram of use cases: The employee can list the different projects affected by his management

3.2.7 Text description

In this part we present the textual description which contains the use case scenarios "The employee can list the different projects affected by his management".

SOMMAIRE Consultation de projets	
Titre	Consultation
But	L'employé peut lister les différents projets affectés par son manager
Acteur	Employé
DESCRIPTION DES ENCHAINEMENTS	
Pré conditions	
Chaque utilisateur doit avoir un compte □ L'employé accède à l'interface d'accueil.	
Scénario nominal	
Consultation : □ Le système affiche les projets assignés à l'employé	
Enchainements alternatifs	
□ Connexion interrompu	

FIGURE 3.5: Text description of the scenario "The employee can list the different projects affected by his management ".

3.2.8 Sequence diagram of the scenario "The employee can list the different projects affected by his management"

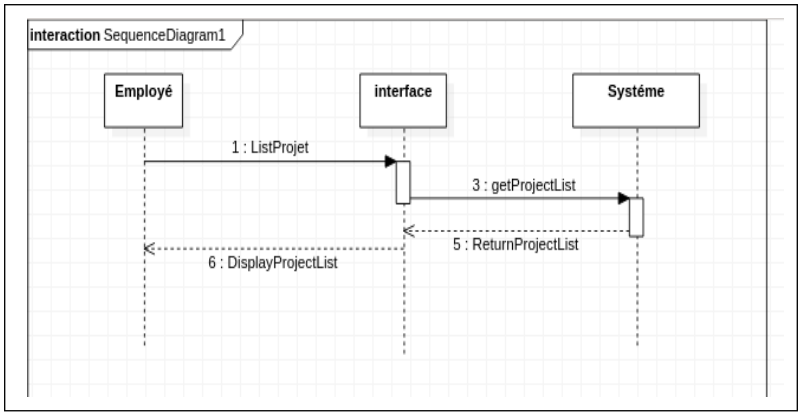

FIGURE 3.6: Re fi ned diagram of use cases: The employee can list the different projects affected by his management

3.3 Sprint Analysis and Design 2

In this part, we establish the refinements of the different use cases of the second sprint in order to deliver a description on the different possible scenarios.

3.3.1 Analysis of the case "The employee must share his progress according to a committment date with a description"

3.3.2 Refined diagram of the use case "Employee must share their progress according to a commit date with a description"

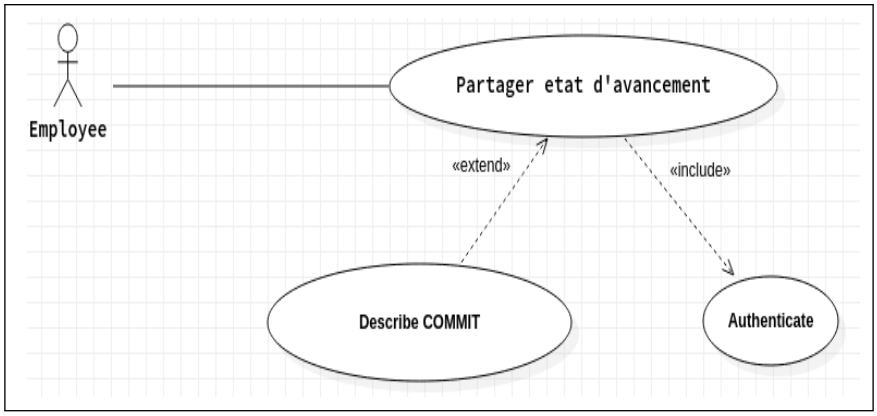

FIGURE 3.7: Use case diagram: The employee must share their progress according to a committment date with a description

3.3.3 Text description

In this part we present the textual description which contains the use case scenario "The employee must share his progress according to a commit date with a description".

SOMMAIRE Etat d'avancement	
Titre	Partager l'état d'avancement
But	Le système doit afficher l'interface dans le quelle l'employer peut ajoutes des commits
Acteur	Employer
DESCRIPTION DES ENCHAINEMENTS	
Pré conditions	
Chaque utilisateur doit avoir un compte • L'employer doit s'authentifier	
Scénario nominal	
Consultation 1. L'employé clique sur « Add commits » 2. Le système affiche un espace de texte 3. L'employé peut ajouter une ou plusieurs commits	
Enchainements alternatifs	
Connexion interrompue	

FIGURE 3.8: Textual description of the scenario "The employee must share his progress according to a committment date with a description".

3.3.4 Sequence diagram of the scenario "The employee must share his progress according to a date of committment description " with a

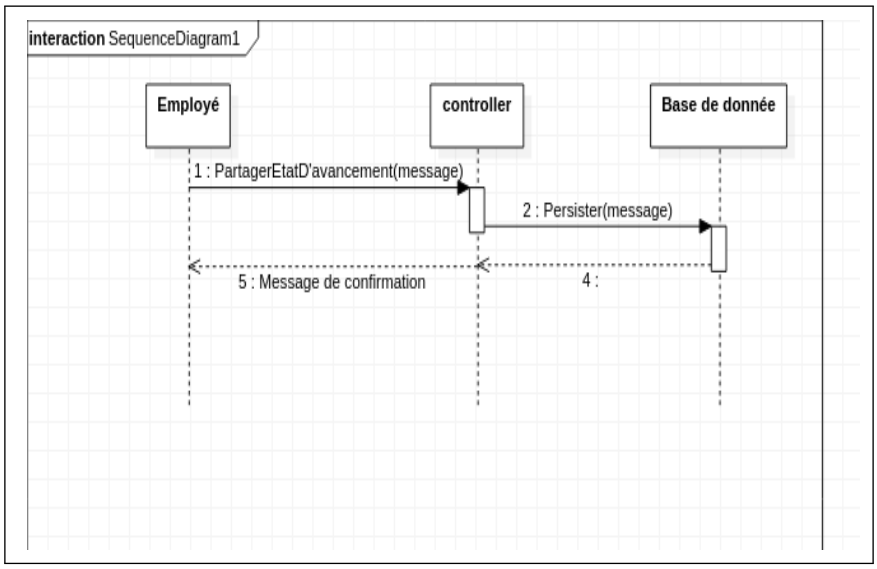

FIGURE 3.9: Sequence diagram of the scenario "The employee must share his progress"

3.4 Analysis of the case "the employee can view his various committment dates for each project assigned to him"

3.4.1 Re fi ned diagram of the use case "the employee can view his various committment dates for each project assigned to him"

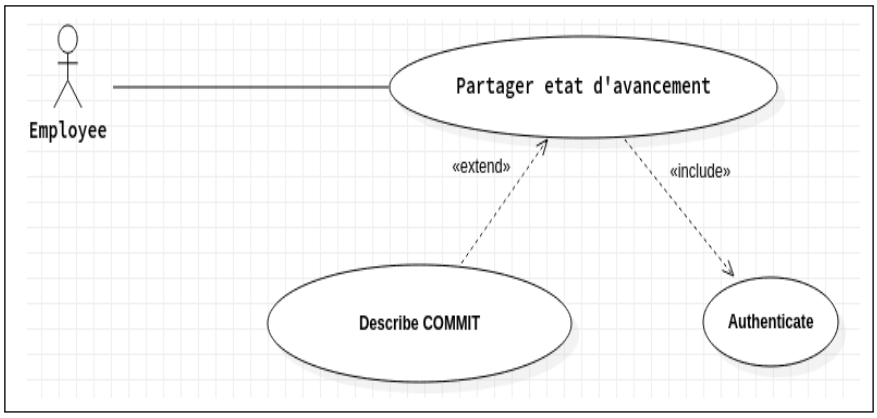

FIGURE 3.10: Use case diagram: the employee can view their different committment dates for each project assigned to them

3.4.2 Text description

In this part we present the textual description which contains the use case scenarios "the employee can view his different committment dates for each project assigned to him".

SOMMAIRE show commit	
Titre	Affichage des commentaire
But	Le système doit afficher les différents commits mis par les employés et les managers
Acteur	Employer
DESCRIPTION DES ENCHAINEMENTS	
Pré conditions	
Chaque utilisateur doit avoir un compte • L'employer doit s'authentifier • L'employé doit ajouter des commits	
Scénario nominal	
Consultation 1. L'employé clique sur « show commits » 2. Le système afficher les différents commits	
Enchainements alternatifs	
Connexion interrompue	

FIGURE 3.11: Textual description of the scenario the employee can view his various committment dates for each project assigned to him ".

3.4.3 Sequence diagram of the scenario "the employee can view his different committment dates for each project assigned to him"

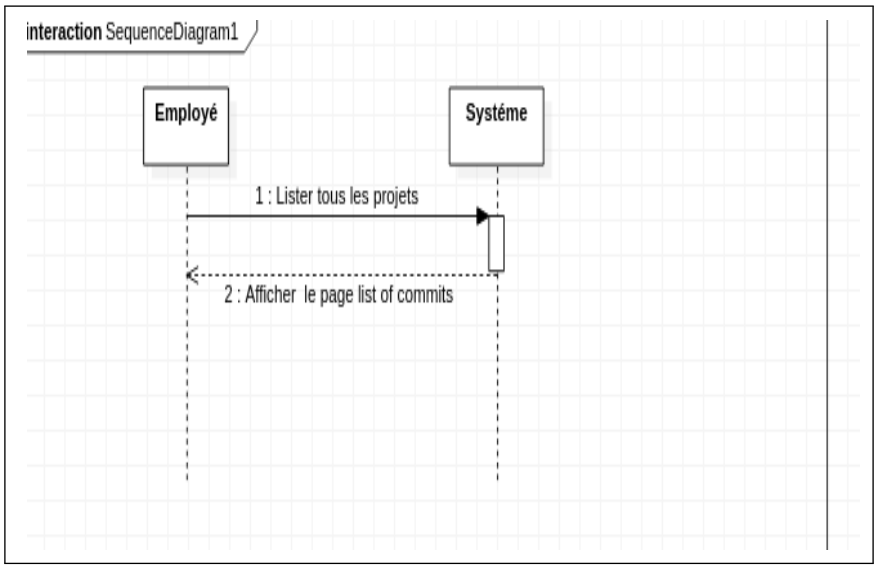

FIGURE 3.12: Sequence diagram of the "employee can view his various committments" scenario

3.5 Realization

In this section, we present the different interfaces relating to the first release.

Home page :

The user can go to the home page and click on the button "Are you an Employee?"

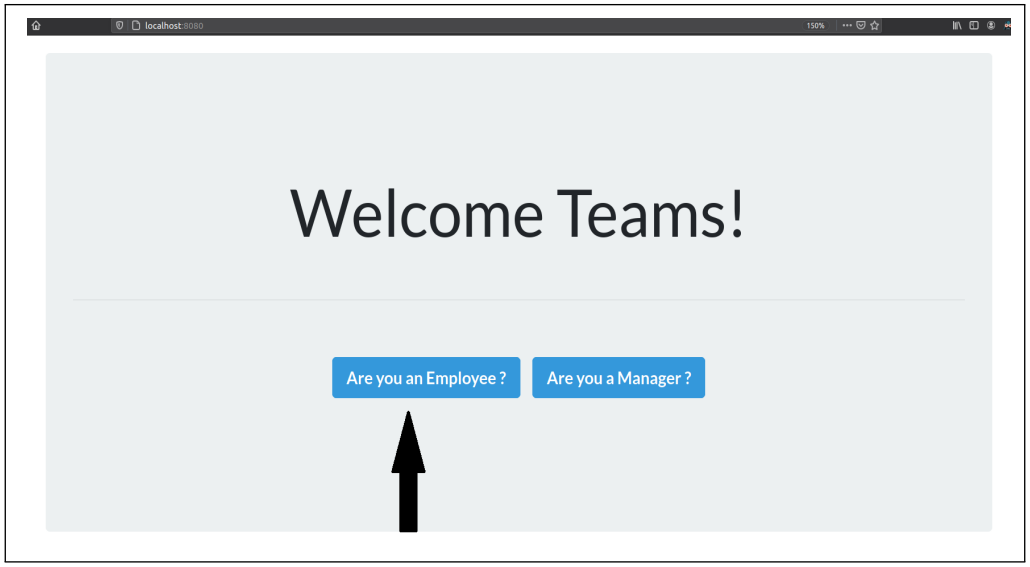

FIGURE 3.13: Home page.

Authentication page:

It is the authentication interface which obliges the employee to authenticate himself before using the application, he must therefore enter his login and password.

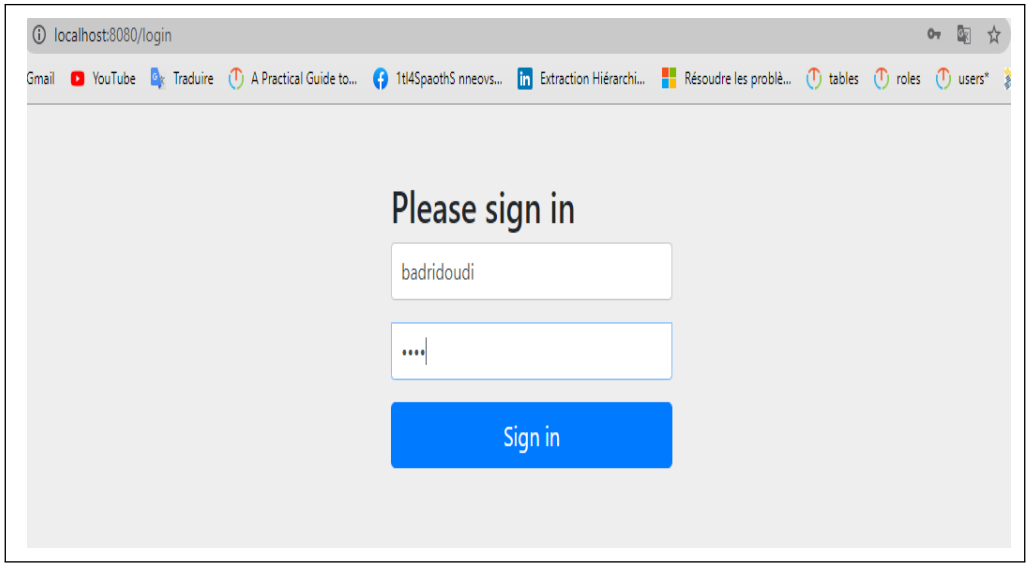

FIGURE 3.14: Authentication

Display of projects assigned to the current user:

The interface that shows all the projects assigned to the logged in employee.

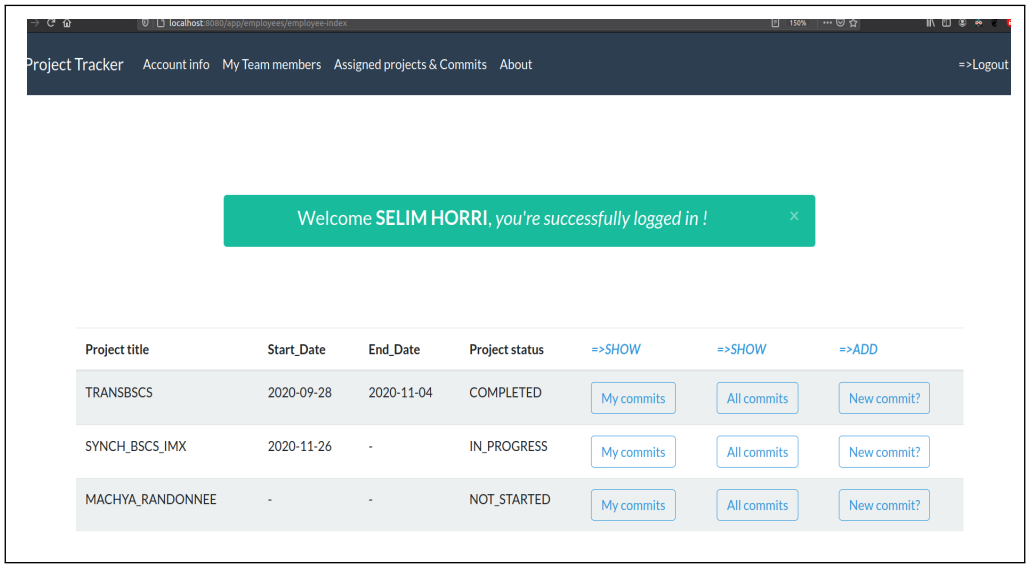

FIGURE 3.15: Display of projects assigned to the user

Interface display add a commit:

The employee can add a commit in a project to which he is assigned.

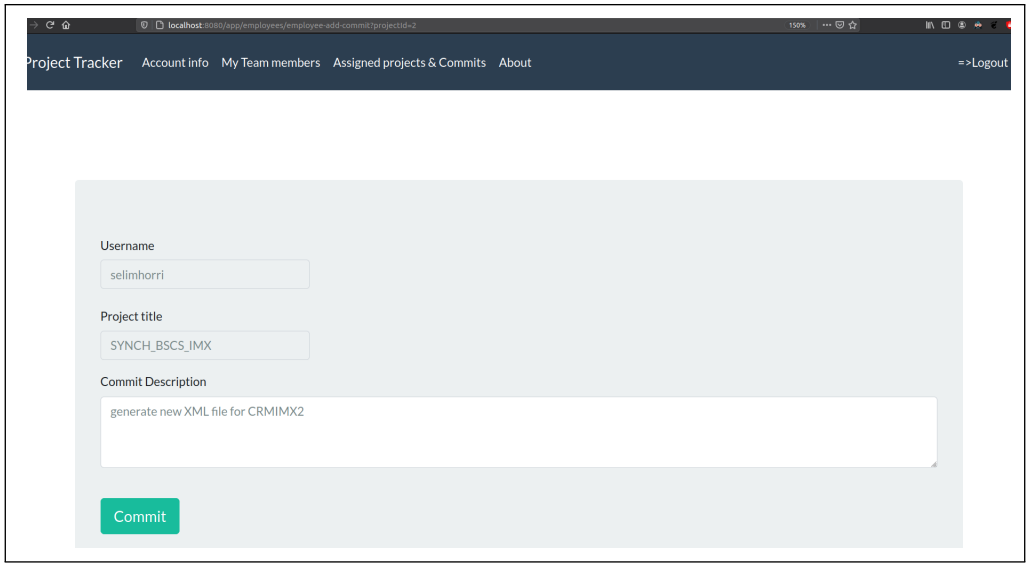

FIGURE 3.16: Interface display add a commit

Confirmation message

A con fi rmation message following a commit.

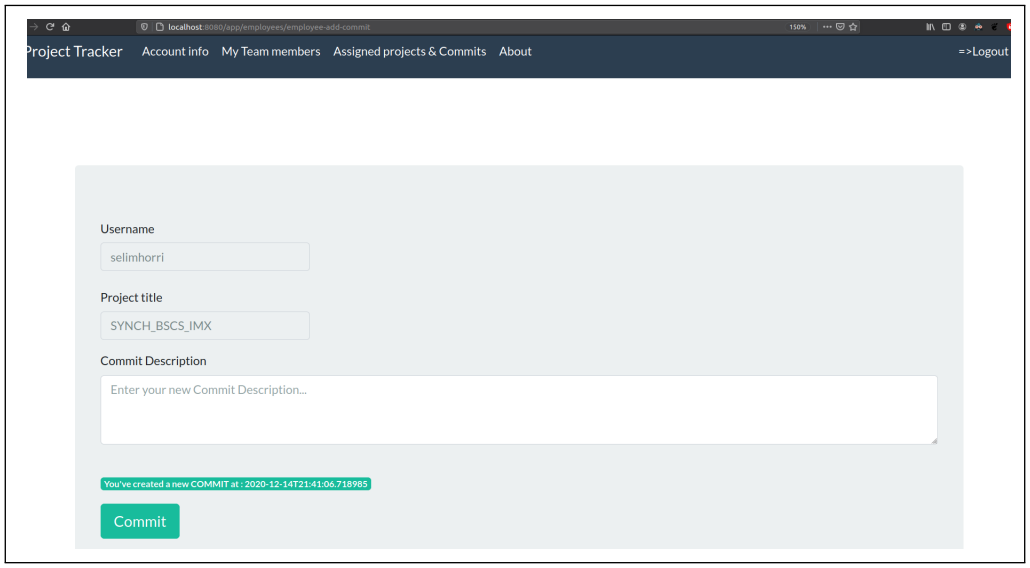

FIGURE 3.17: Display of the con fi rmation message following a commit

Log trace when adding a commit:

```
2020-12-17 15:19:49.241  INFO 25045 --- [nio-8080-exec-6] c.p.pack.controllers.EmployeeController  : COMMIT created successfully at :
2020-12-17T15:19:49.241130
2020-12-17 15:19:52.055  INFO 25045 --- [nio-8080-exec-6] c.p.pack.controllers.EmployeeController  : MAIL successfully sent to
springabcxyzboot@gmail.com
2020-12-17 15:19:53.220  INFO 25045 --- [nio-8080-exec-6] c.p.pack.controllers.EmployeeController  : SMS successfully sent to 22125144
```

FIGURE 3.18: commit log

Receiving an email during a commit:

During addition of a commit, the system send a E-mail To " Springabcxyzboot@gmail.com "

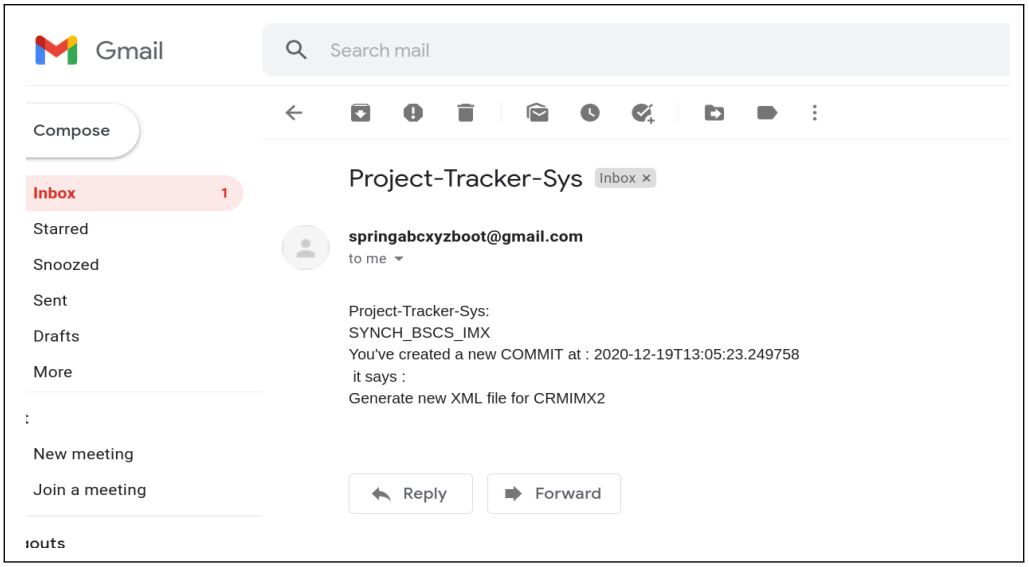

FIGURE 3.19: Receiving an email during a commit

Receipt of an SMS during a commit:

When adding a commit, the system will send an SMS.

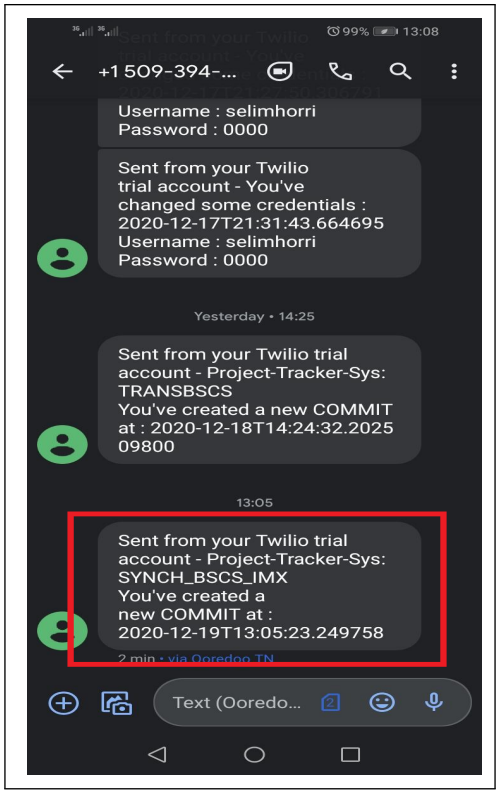

FIGURE 3.20: Receipt of an SMS during a commit

Display of a new commit:

This interface shows the previously added commit.

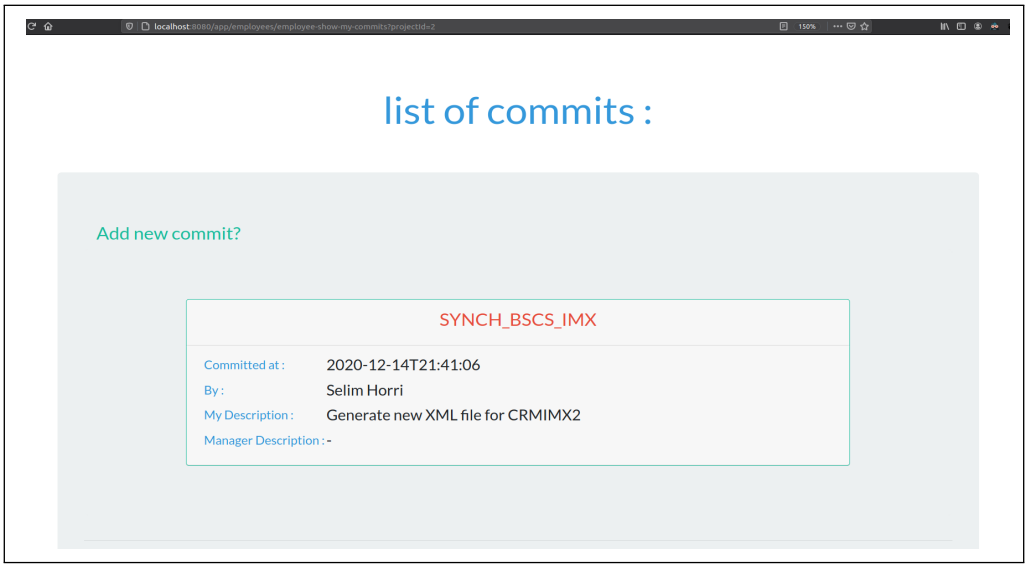

FIGURE 3.21: Display of a new commit

Display of all commits:

This page shows all the commits of the employee as well as those of his colleagues.

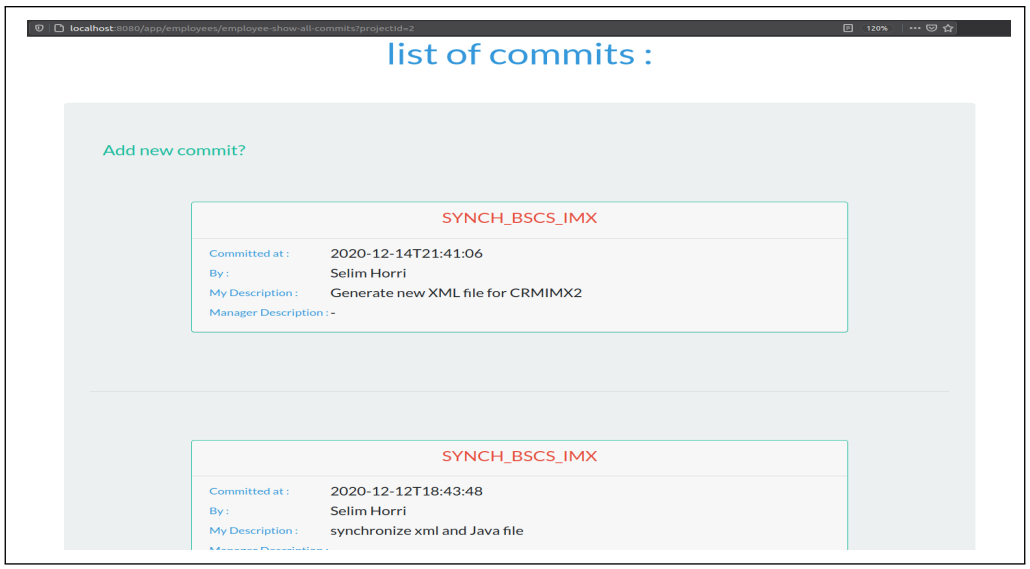

FIGURE 3.22: Display of all commits

3.6 Conclusion

Throughout this chapter, we have presented the Sprint Backlog from the first release. Then, we put the accent on its speci fi cation and its design then its realization.

Chapter

4

Chapter 4: THE SECOND RELEASE

Summary

4.1 INTRODUCTION

In this chapter, we deal with the user stories of the second release. First, we present the organization of the sprints for this release as well as all the speci fi cations, then we present the design phases of its sprints as well as the interfaces relating to their implementation.

4.2 Sprint backlog

To fully understand the features of the second release, we are going to reserve this part to present the backlog of the sprints of the latter. the
first sprint focuses affected. on the authentication and consultation of the projects

Task	Role	Priority
The manager must to authenticate	Manager	M
The manager can consult the affected projects	Manager	VS

TABLE 4.1: Backlog sprint 1

The second sprint focuses on reviewing progress and
monitors projects.

Task	Role	Priority
The manager must consult the state advancement	Manager	M
The manager follow up	Manager	VS

TABLE 4.2: Backlog sprint 2

So the third sprint concerns the creation, assignment and management of a project.

Task	Role	Priority
The manager assign a project	Manager	VS
The manager manages a project	Manager	VS

TABLE 4.3: Backlog sprint 3

4.3 Analysis and Design of Sprint 1:

In this part, we establish the refinements of the different use cases of the first sprint and some design diagrams in order to deliver a description on the different possible scenarios.

4.3.1 Analysis of the "authentication" case:

The system ensures the identity of the user through his login and password.

4.3.2 Refined "Authentication" Use Case Diagram

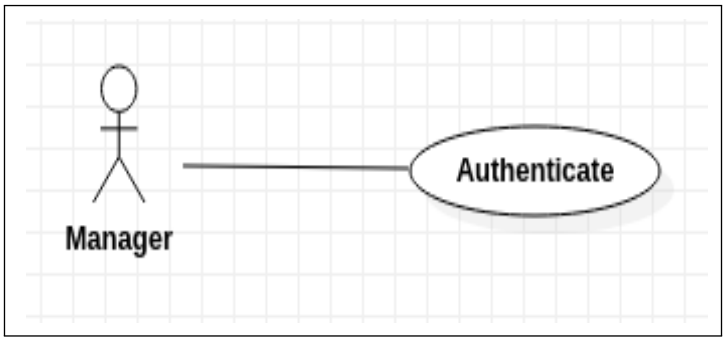

FIGURE 4.1: "Authentication" use case diagram

4.3.3 Text description

In this part we present the textual description which contains the

"Authentication" use case scenarios.

SOMMAIRE AUTHENTIFICATION	
Titre	AUTHENTIFICATION
But	Le système s'assure de l'identité de manager à travers son login et son mot de passe
Acteur	Manager
DESCRIPTION DES ENCHAINEMENTS	
Pré conditions	
Chaque utilisateur doit avoir un compte □ Le Manager accédé à l'application	
Scénario nominal	
L'authentification 1. Le manager clique sur « login » 2. Le système afficher la page d'accueil	
Enchainements alternatifs	
1. Les chams login et mot de passe sont non valide et/ou vide a. Le système affiche un message d'erreur	

FIGURE 4.2: Text description of the "Authentication" scenario.

4.3.4 Sequence diagram of the "Authentication" scenario

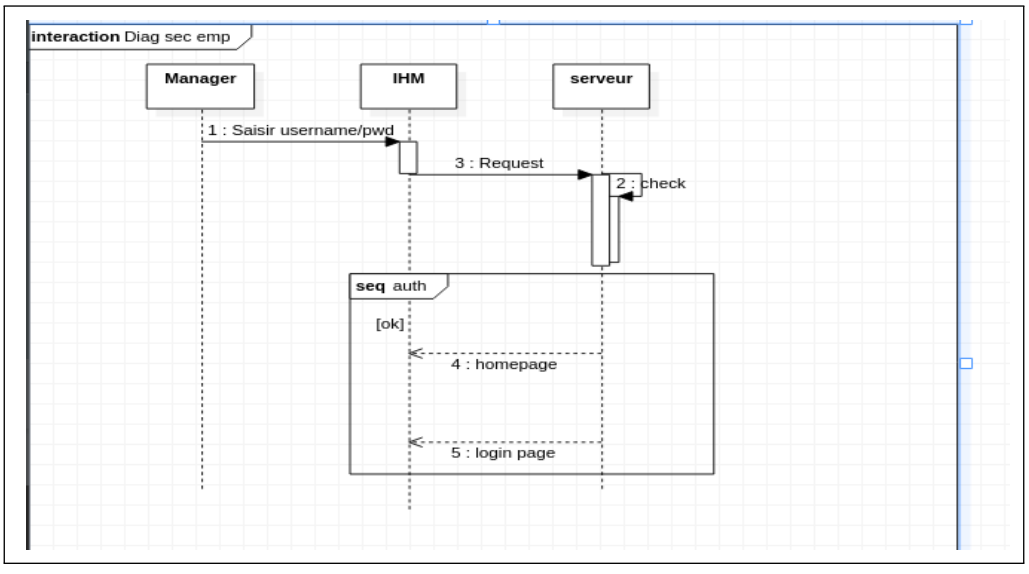

FIGURE 4.3: Sequence diagram of the "Authentication" scenario

4.3.5 Analysis of the case "Consult affected projects":

The manager can list the different projects assigned by himself.

4.3.6 Refined "View Affected Projects" use case diagram

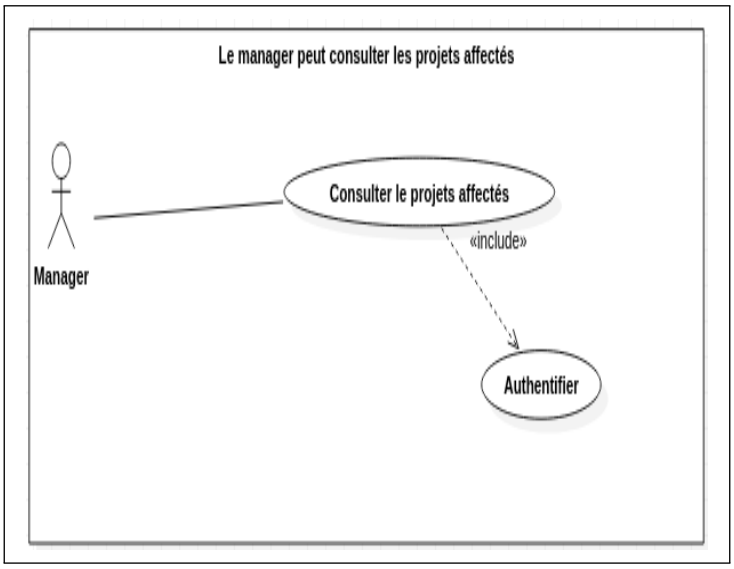

FIGURE 4.4: Text description of the "Consult affected projects" scenario.

4.3.7 Text description

In this part we present the textual description which contains the use case scenarios "Consult affected projects".

SOMMAIRE Consultation de projets	
Titre	Consultation
But	Le Manager peut lister les différents projets affectés par lui-même
Acteur	Manager
DESCRIPTION DES ENCHAINEMENTS	
Pré conditions	
Chaque utilisateur doit avoir un compte Le Manager accède à l'interface d'accueil.	
Scénario nominal	
Consultation : Le système affiche les projets assignés à chaque employé	
Enchainements alternatifs	
Connexion interrompu	

FIGURE 4.5: Text description of the "Consult affected projects" scenario.

4.3.8 Sequence diagram of the "Consult affected projects" scenario

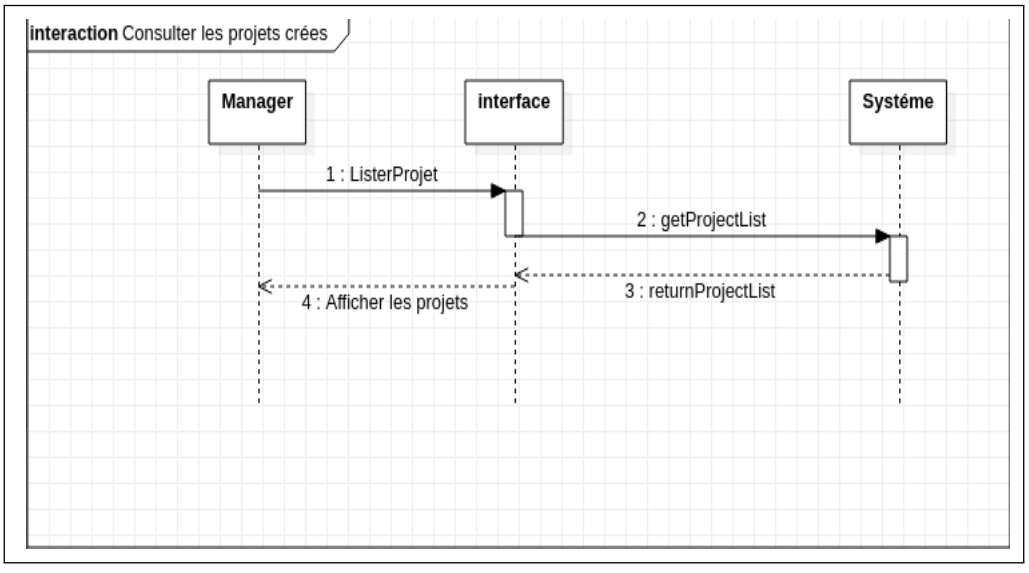

FIGURE 4.6: Sequence diagram of the "Consult affected projects" scenario

4.4 Analysis and Design of Sprint 2:

In this part, we establish the refinements of the different use cases of the second sprint and some design diagrams to finish.

to provide a description of the different possible scenarios.

4.4.1 Refined "Consult progress" use case diagram state

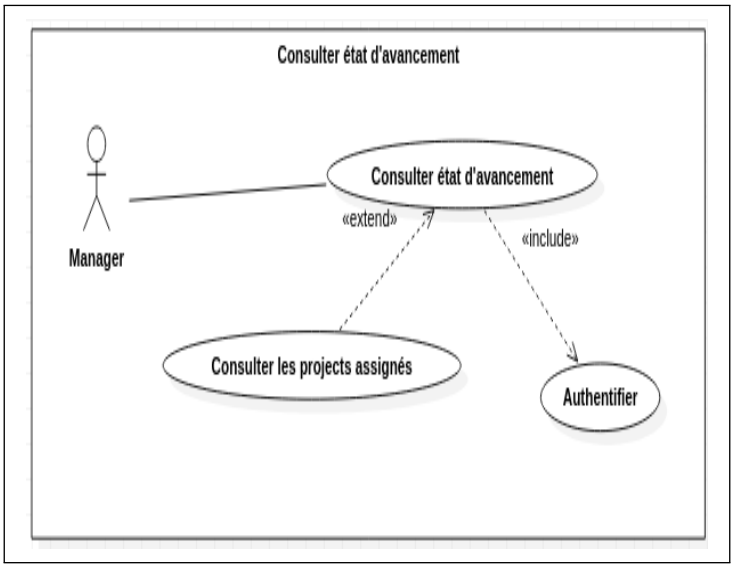

FIGURE 4.7: Text description of the "Consult the progress" scenario.

4.4.2 Text description

In this part we present the textual description which contains the "Consult the progress" use case scenarios.

SOMMAIRE AVANCEMENT	
Titre	Etat d'avancement
But	Le système doit afficher les différents commits effectués par les employés pour chaque projet
Acteur	Manager
DESCRIPTION DES ENCHAINEMENTS	
Pré conditions	
Chaque utilisateur doit avoir un compte Le Manager visualise le projet concerné	
Scénario nominal	
Consultation 1. Le manager clique sur « le nom du projet » 2. Le système afficher les différents commits	
Enchainements alternatifs	
Connexion interrompue	

FIGURE 4.8: Text description of the "Consult the progress" scenario.

4.4.3 Diagram of sequence from scenario "Consult state advancement "

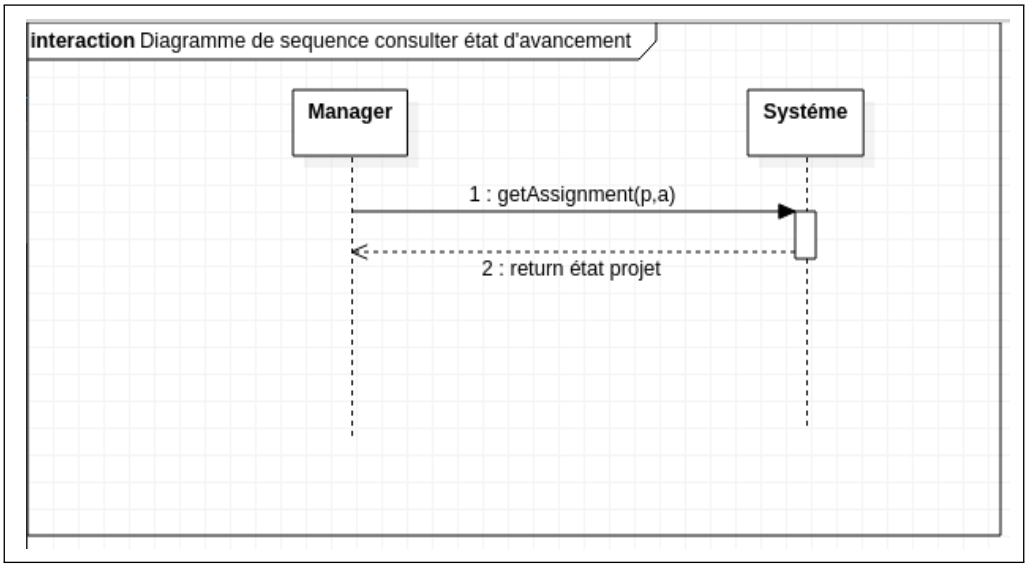

FIGURE 4.9: Sequence diagram of the "Consult the progress" scenario

4.4.4 Analysis of the case "Ensures follow-up":

The system must display to the current manager the various commits performed by the employees.

4.4.5 Refined "Keep Track" Use Case Diagram

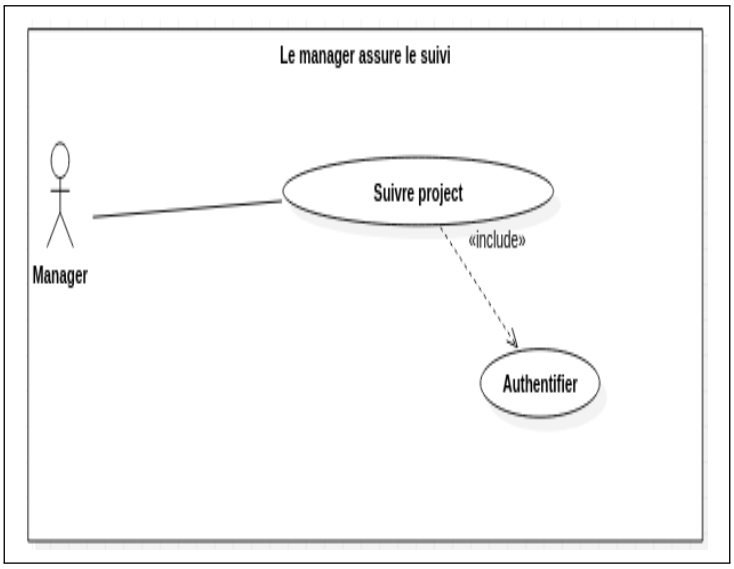

FIGURE 4.10: Text description of the scenario "Ensures the follow-up".

4.4.6 Text description

In this part we present the textual description which contains the "Follow up" use case scenarios.

SOMMAIRE Suivie	
Titre	Assurer le suivi
But	Le système doit afficher les différents commits effectués par les employés pour chaque projet
Acteur	Manager

DESCRIPTION DES ENCHAINEMENTS

Pré conditions

Chaque utilisateur doit avoir un compte

- Le Manager visualise les commits effectués par les employés

Scénario nominal

Consultation

1. Le manager clique sur « show commits »
2. Le système affiche les différents commits
3. Le manager peut ajouter, modifier ou supprimer un commentaire sur le commit de l'employé

Enchainements alternatifs

Connexion interrompue

FIGURE 4.11: Text description of the scenario "Ensures the follow-up".

4.4.7 Follow-up scenario sequence diagram

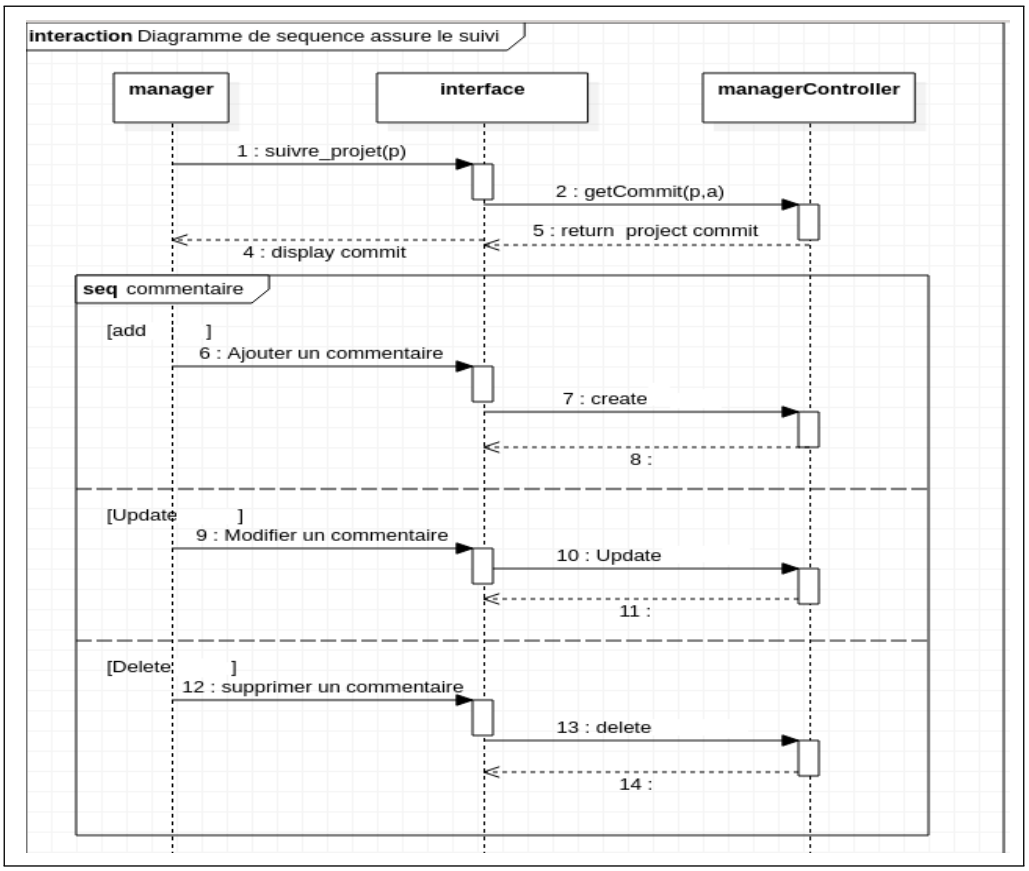

FIGURE 4.12: Text description of the "Ensure follow-up" scenario.

4.5 Analysis and Design of Sprint 3:

In this part, we establish the refinements of the different use cases of the third sprint and some design diagrams in order to deliver a description on the different possible scenarios.

4.5.1 Analysis of the "Assign a project" case:

The manager can create projects

4.5.2 Refined "Assign Project" use case diagram

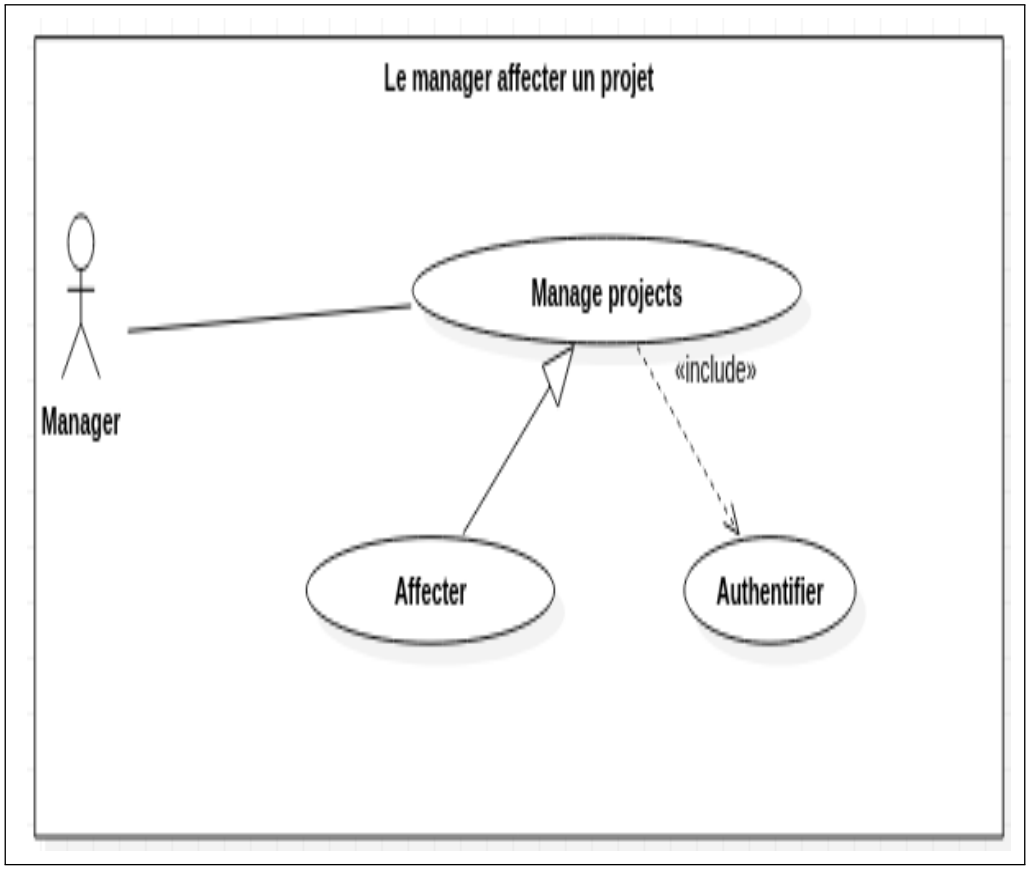

FIGURE 4.13: Text description of the "Assign a project" scenario.

4.5.3 Text description

In this part we present the textual description which contains the "Assign a project" use case scenarios.

SOMMAIRE AFFECTATION	
Titre	Affectation des projets
But	Le Manager doit affecter un ou plusieurs projet à un ou plusieurs employés
Acteur	Manager
DESCRIPTION DES ENCHAINEMENTS	
Pré conditions	
Chaque utilisateur doit avoir un compte - Le Manager doit être authentifie - Le Manager doit créer un projet	
Scénario nominal	
Consultation 1. Le système affiche l'interface concernée 2. Le manager clique sur « Ajouter un Employé » 3. Le projet est affecté aux employés	
Enchainements alternatifs	
Connexion interrompue	

FIGURE 4.14: Text description of the "Assign a project" scenario.

4.5.4 Sequence diagram of the "Assign a project" scenario

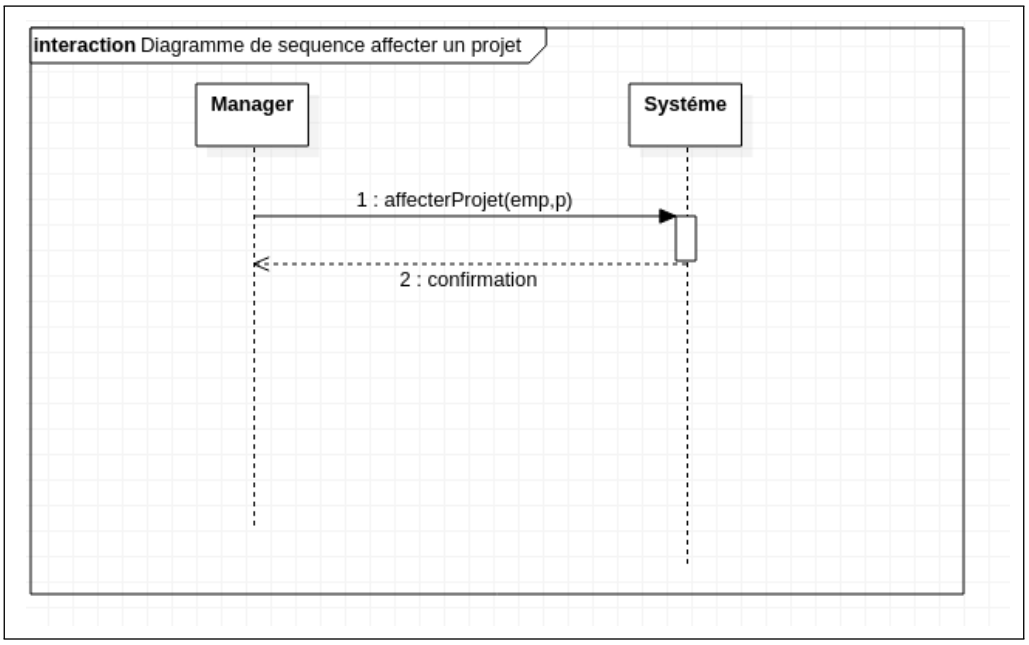

FIGURE 4.15: Sequence diagram of the "Assign a project" scenario

4.5.5 Analysis of the "Manage a project" case:

The manager can manage projects

4.5.6 Refined "Manage a project" use case diagram

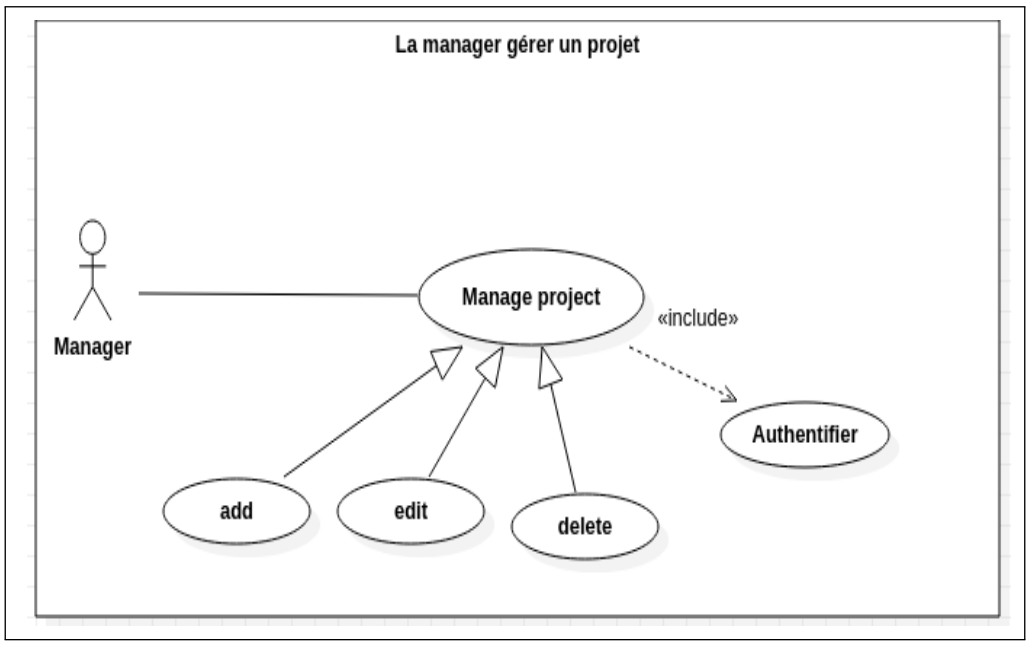

FIGURE 4.16: Text description of the "Manage a project" scenario.

4.5.7 Text description

In this part we present the textual description which contains the "Manage a project" use case scenarios.

SOMMAIRE DE Gérer un projet	
Titre	Gérer un projet
But	Le manager assure la gestion des projets
Acteur	Manager
DESCRIPTION DES ENCHAINEMENTS	
Pré conditions	
✓ Authentification réussite. ✓ Le manager peut accéder à un projet spécifique.	
Scénario nominal	
Gestion d'un projet 1. Le système affiche l'interface concernée 2. Le manager peut Ajouter, Modifier ou supprimer un projet	
Enchainements alternatifs	
Connexion interrompue	

FIGURE 4.17: Text description of the "Manage a project" scenario.

4.5.8 Sequence diagram of the "Manage a project" scenario

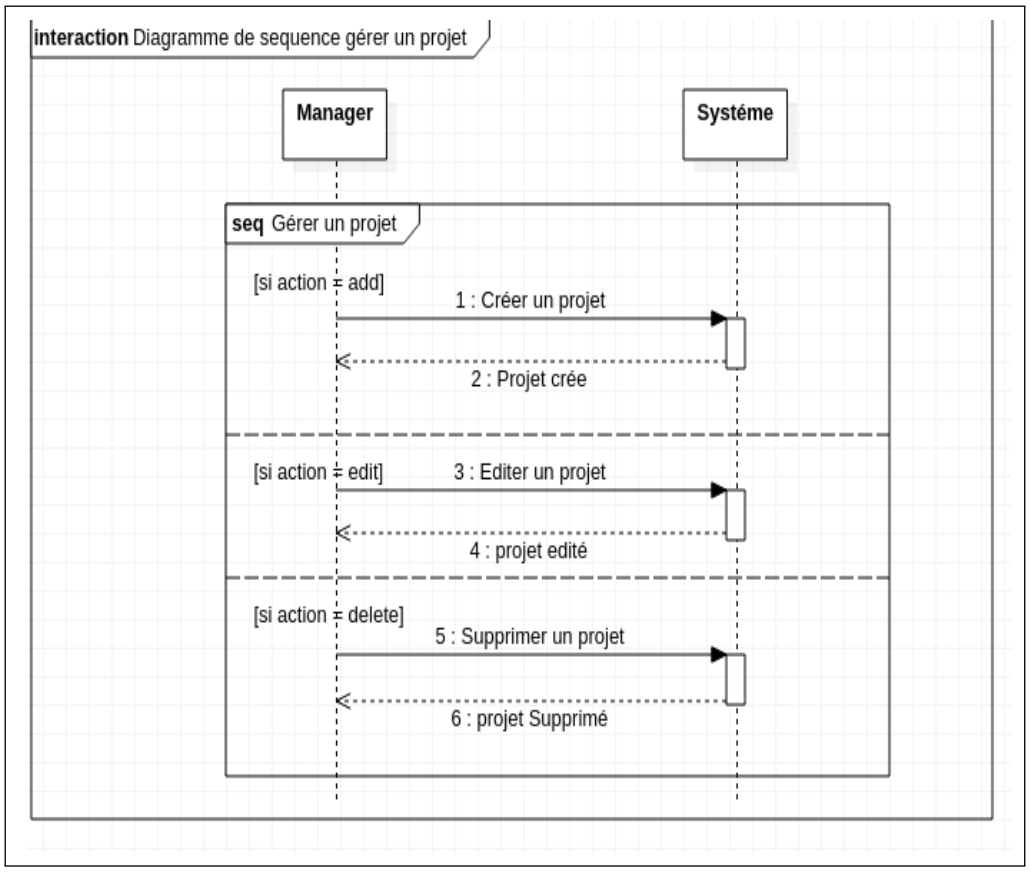

FIGURE 4.18: Sequence diagram of the "Manage a project" scenario

4.6 Realization

In this section, we present the different interfaces relating to the second release.

Home page :

The user can go to the home page and click on the "Are you a Manager?" Button.

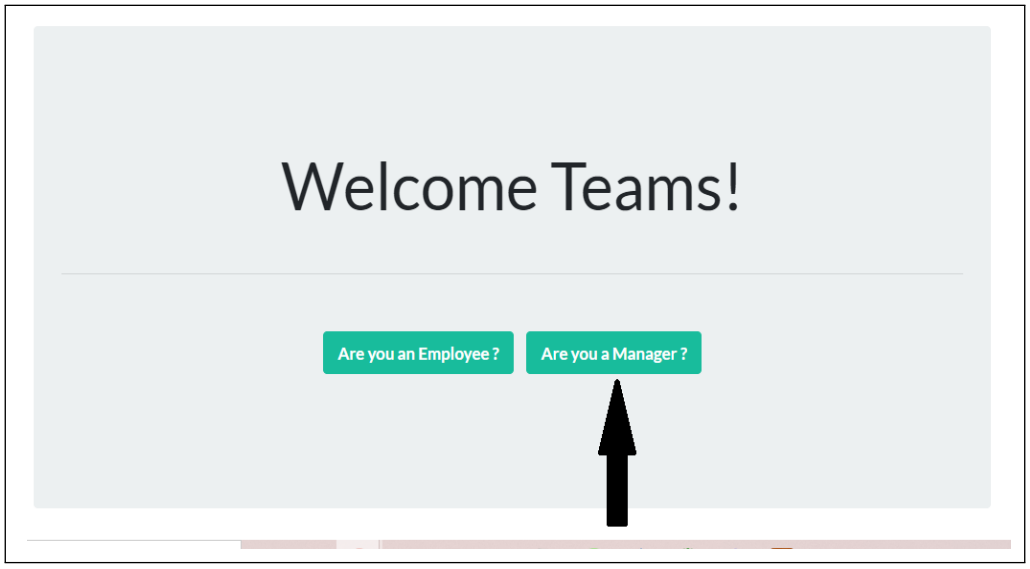

FIGURE 4.19: Home page.

Authentication page:

It is the authentication interface which obliges the manager to authenticate himself before using the application, he must therefore enter his login and password.

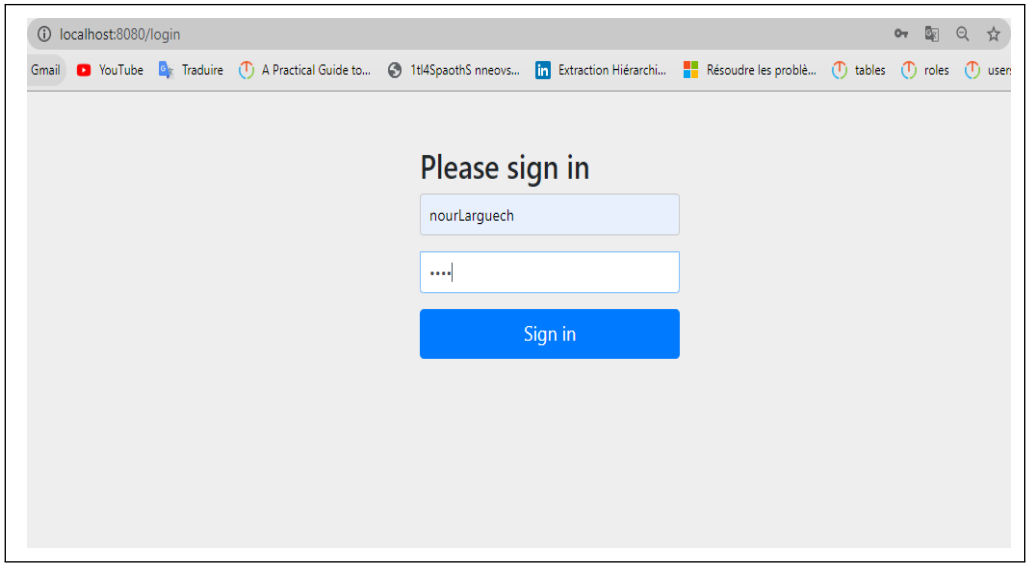

FIGURE 4.20: Authentication page.

Manager home page.

The interface that shows all the projects created by the manager.

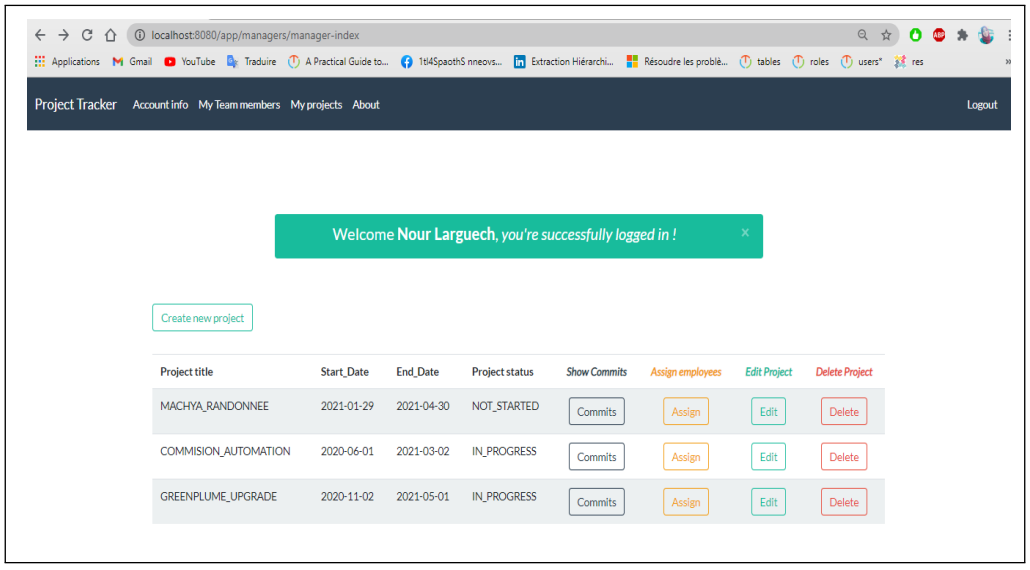

FIGURE 4.21: Manager home page.

Display of a commit:

The manager must click on one of the "commit" button as speci fi c. a project

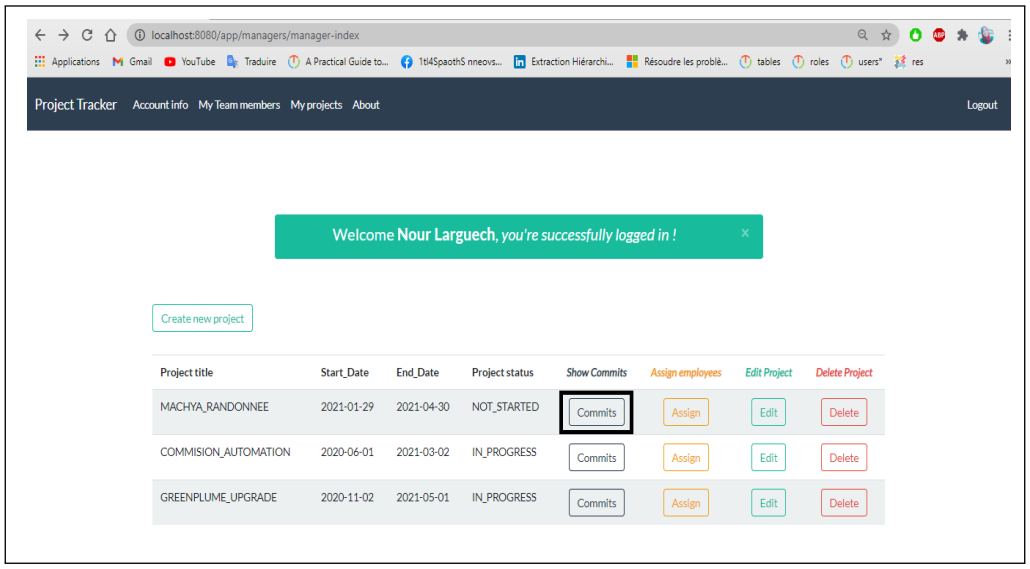

FIGURE 4.22: Display of a commit.

Display of a commit:

This interface demonstrates commits created by employees.

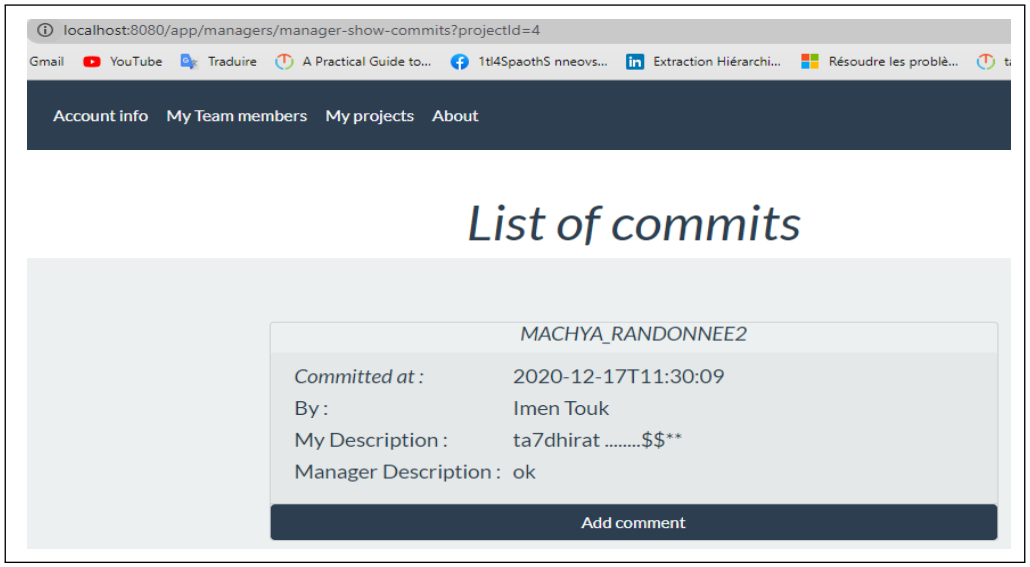

FIGURE 4.23: Display of a commit.

Delete a project:

In order to delete a project, the manager must click on "Delete".

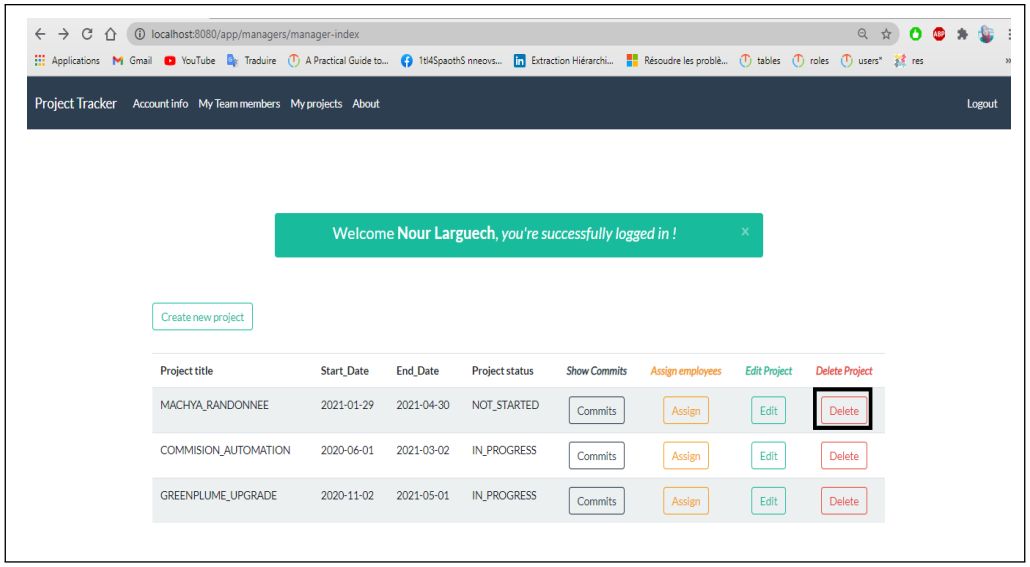

FIGURE 4.24: Delete a project.

Confirmation message:

After clicking on the delete button, a con fi rmation message is displayed to the manager to con fi rm the action.

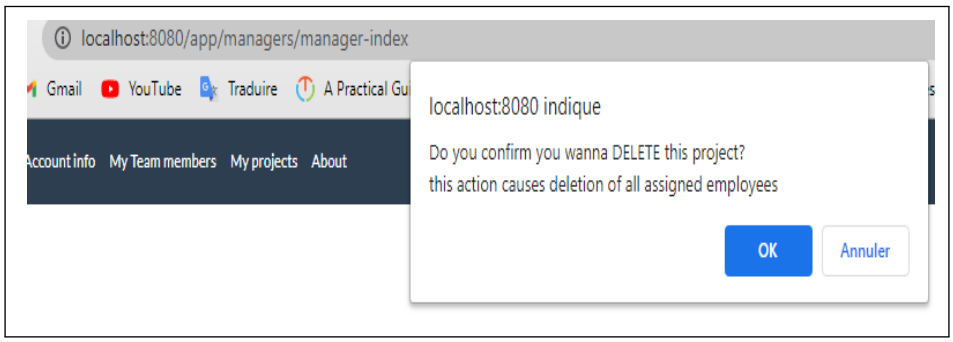

FIGURE 4.25: Con fi rmation message.

Creation of a new project:

To create a project, the manager must click on the button "Create new project"

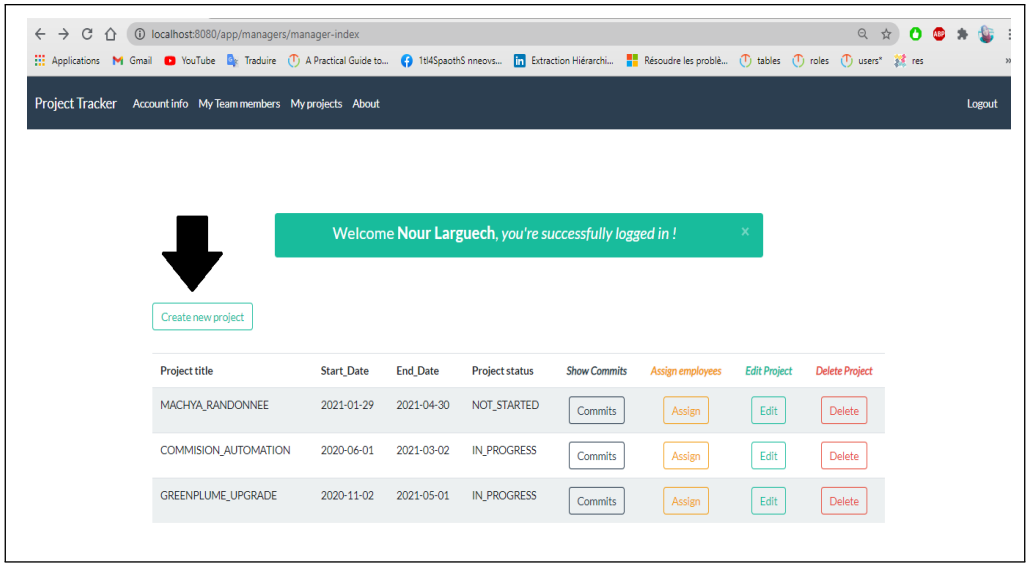

FIGURE 4.26: Creation of a new project.

Creation of a new project:

The manager is invited to fill in the different fields of a project and assign it to one or more employees.

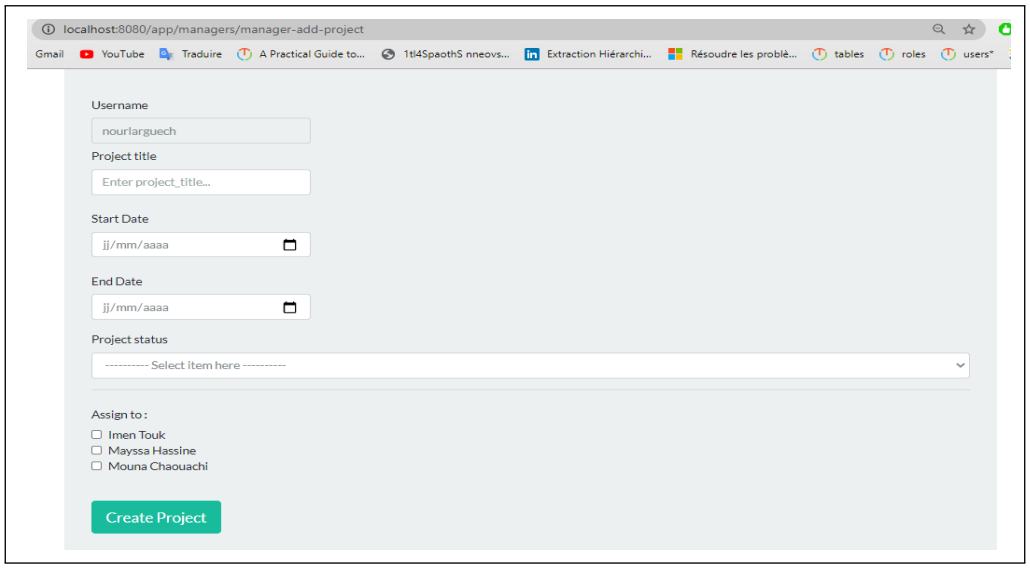

FIGURE 4.27: Creation of a new project.

Confirmation message display:

A con fi rmation message indicates the creation of the project.

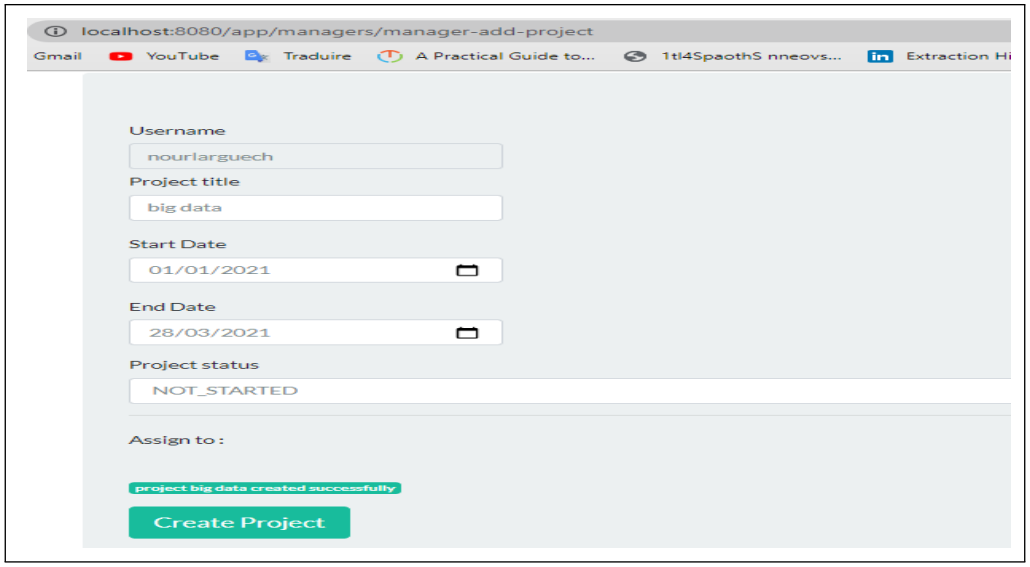

FIGURE 4.28: Display of con fi rmation message.

Modify a project:

In order to modify a project, the manager must click on the "Edit" button.

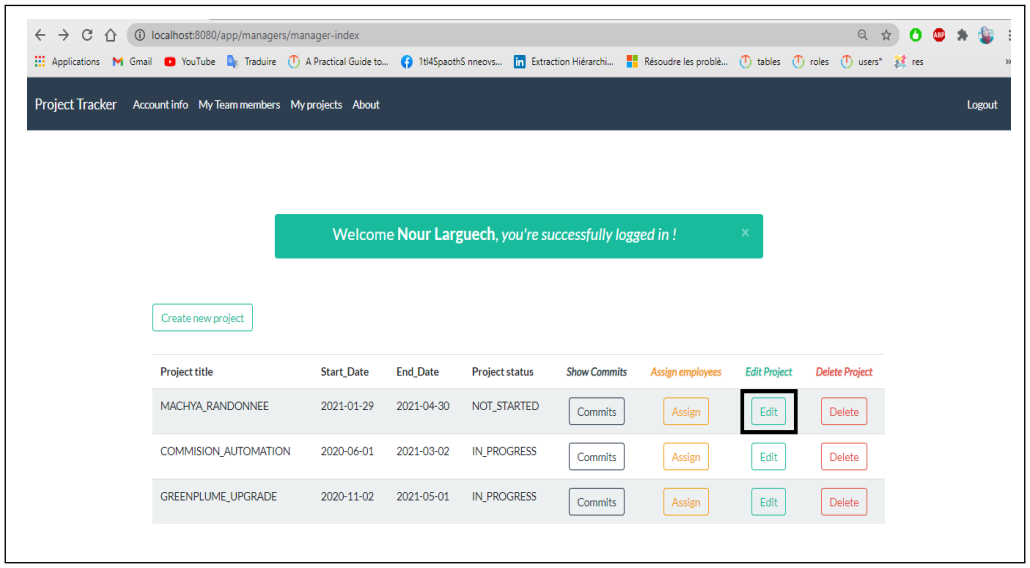

FIGURE 4.29: Modify a project.

Modify the end date of a project:

The manager can modify any field.

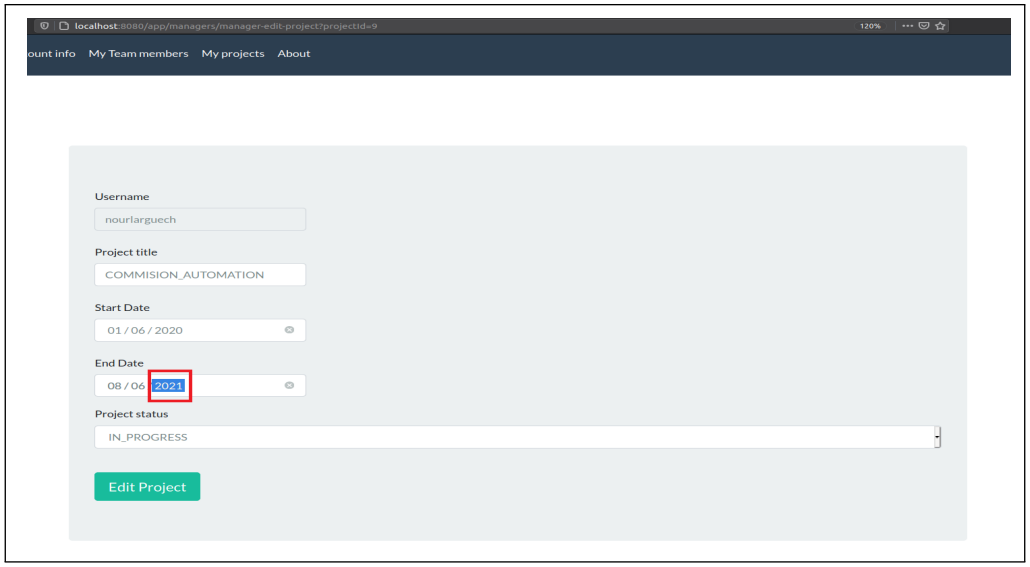

FIGURE 4.30: Modify the end date of a project.

Confirmation message display:

A con fi rmation message indicating the modification of the project.

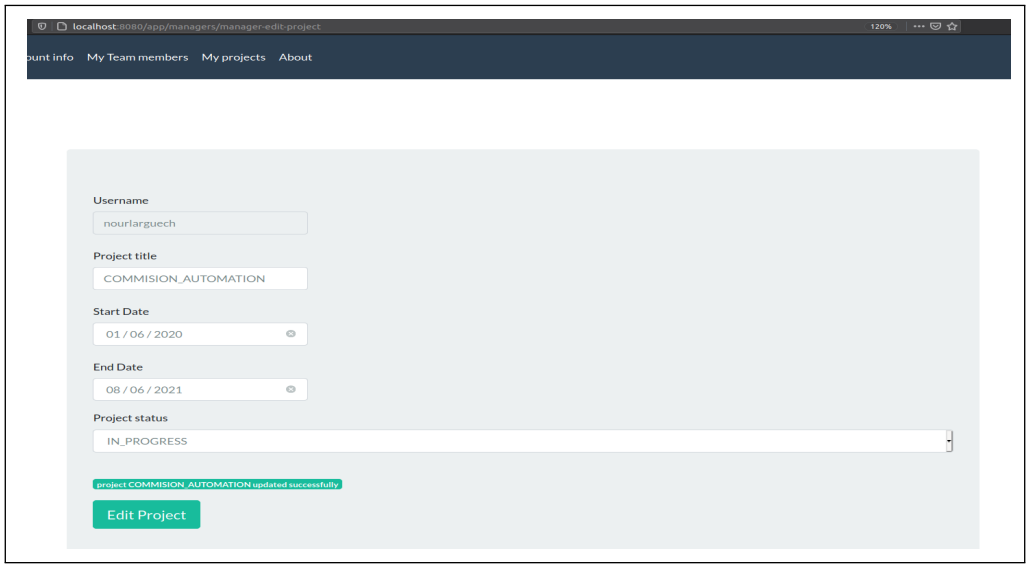

FIGURE 4.31: Display of con fi rmation message.

Assign a project to an employee:

The manager can assign one or more members to a project.

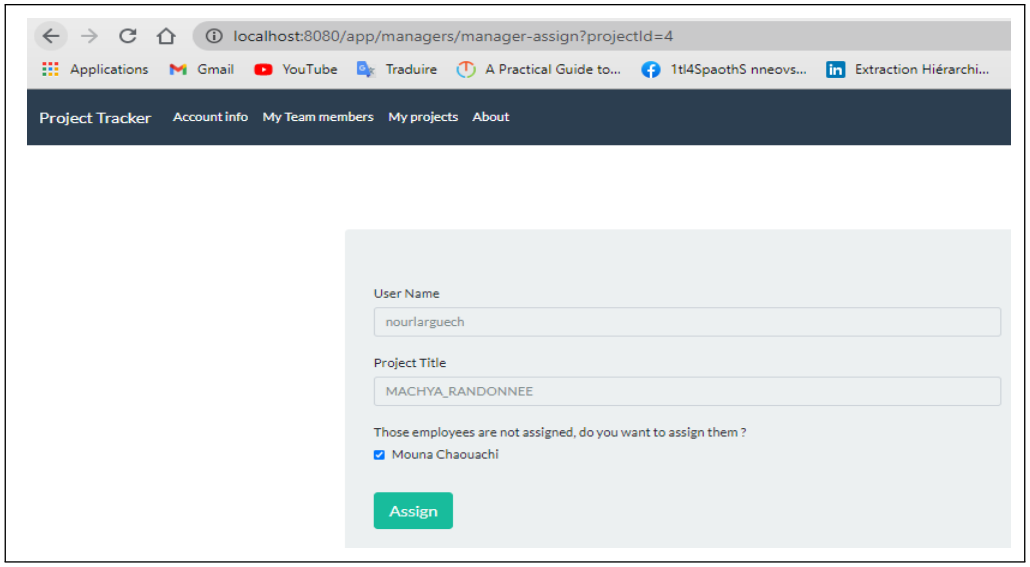

FIGURE 4.32: Assign a project to an employee.

Confirmation message:

The display of the con fi rmation message following the assignment.

FIGURE 4.33: Con fi rmation message.

Adding a comment on a project:

The manager can add a comment on a project.

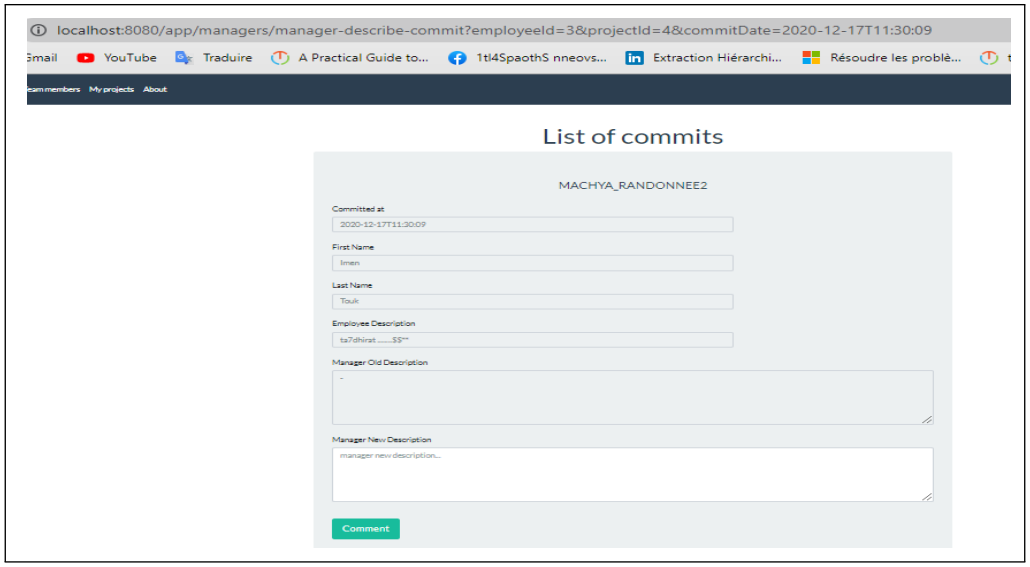

FIGURE 4.34: Adding a comment on a project.

Confirmation message:

The con fi rmation message after adding the comment.

FIGURE 4.35: Con fi rmation message.

List of collaborators:

The interface that shows the manager's team.

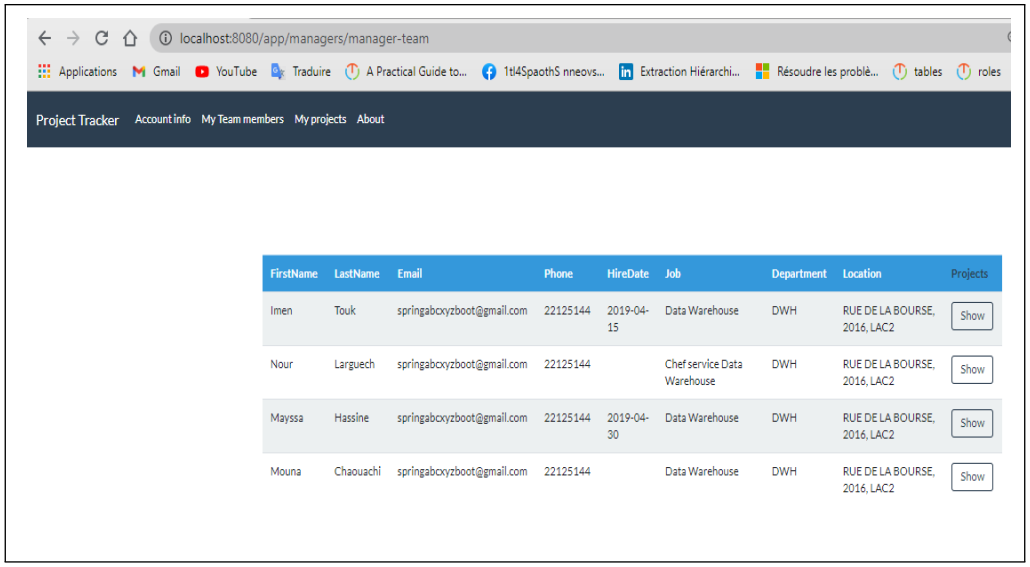

FIGURE 4.36: List of collaborators.

Manager Information:

The display of information about the manager.

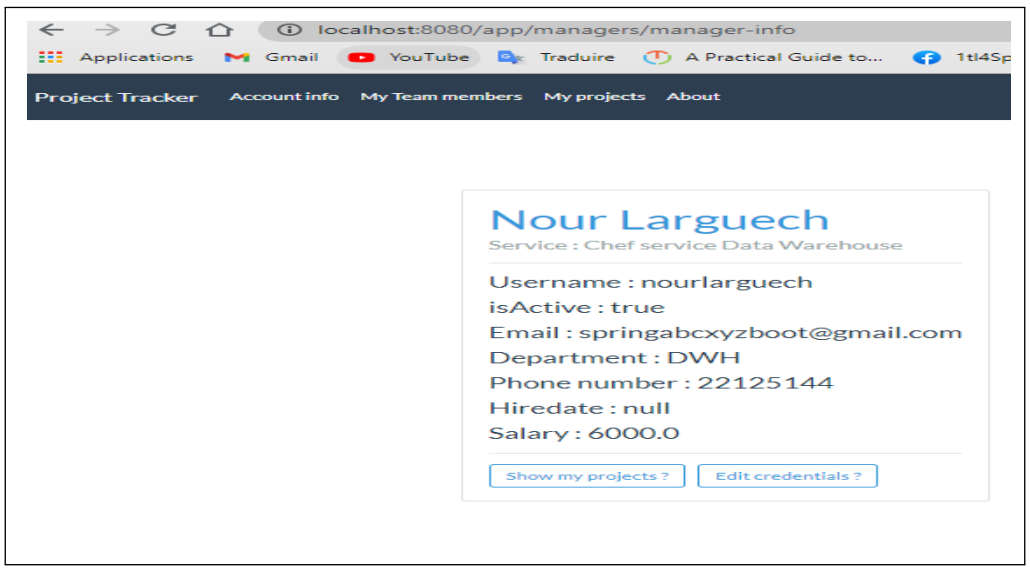

FIGURE 4.37: Manager information

4.7 Conclusion

Throughout this chapter, we have presented the Sprint Backlog for the second release.

Then, we put the accent on its speci fi cation and its design then its realization.

Chapter

5

Chapter 5: THE THIRD RELEASE

Summary

5.1 INTRODUCTION

In this chapter, we deal with the user stories of the third release. First, we present all the speci fi cations, then we present the design phases of its sprints as well as the interfaces relating to the realization of the third release.

5.1.1 Sprint backlog:

To fully understand the features of the third release, we are going to reserve this part to present the backlog of the sprints of the latter. The first sprint focuses on employee management.

Task	Role	Priority
The administrator must manage employees	Administrator	M

TABLE 5.1: Backlog sprint 1

The second sprint focuses on department management.

Task	Role	Priority
The administrator must manage departments	Administrator	S

TABLE 5.2: Backlog sprint 2

The third sprint focuses on rental management.

Task	Role	Priority
The administrator must manage rentals	Administrator	S

TABLE 5.3: Backlog sprint 3

5.2 Analysis and Design of sprint 1:

In this part, we establish the refinements of the different use cases of the first sprint and some design diagrams in order to deliver a description on the different possible scenarios.

5.2.1 Analysis of the "Managing employees" case:

The system ensures the identity of the user through his login and password.

5.2.2 Refined "Manage Employees" Use Case Diagram

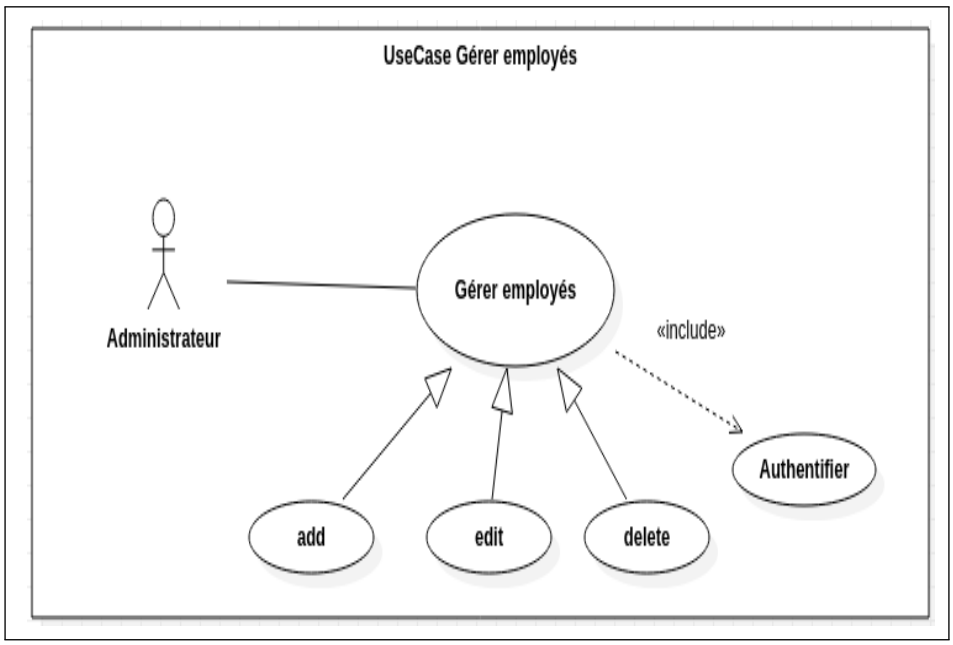

FIGURE 5.1: Refined Use Case Diagram: Manage Employees

5.2.3 Text description

In this part we present the textual description that contains the "Manage Employees" use case scenarios.

SOMMAIRE Gérer les employés	
Titre	Gérer les employés
But	Le système doit gérer les employés
Acteur	Administrateur
DESCRIPTION DES ENCHAINEMENTS	
Pré conditions	
Chaque utilisateur doit avoir un compte • L'administrateur doit s'authentifier	
Scénario nominal	
Consultation 1. L'administrateur clique sur « Manage employees » 2. Le système affiche les différents employés 3. L'administrateur peut ajouter, modifier ou supprimer un employé	
Enchainements alternatifs	
Connexion interrompue	

FIGURE 5.2: Refined Use Case Diagram: Manage Employees

5.2.4 Sequence diagram of the "Manage employees" scenario

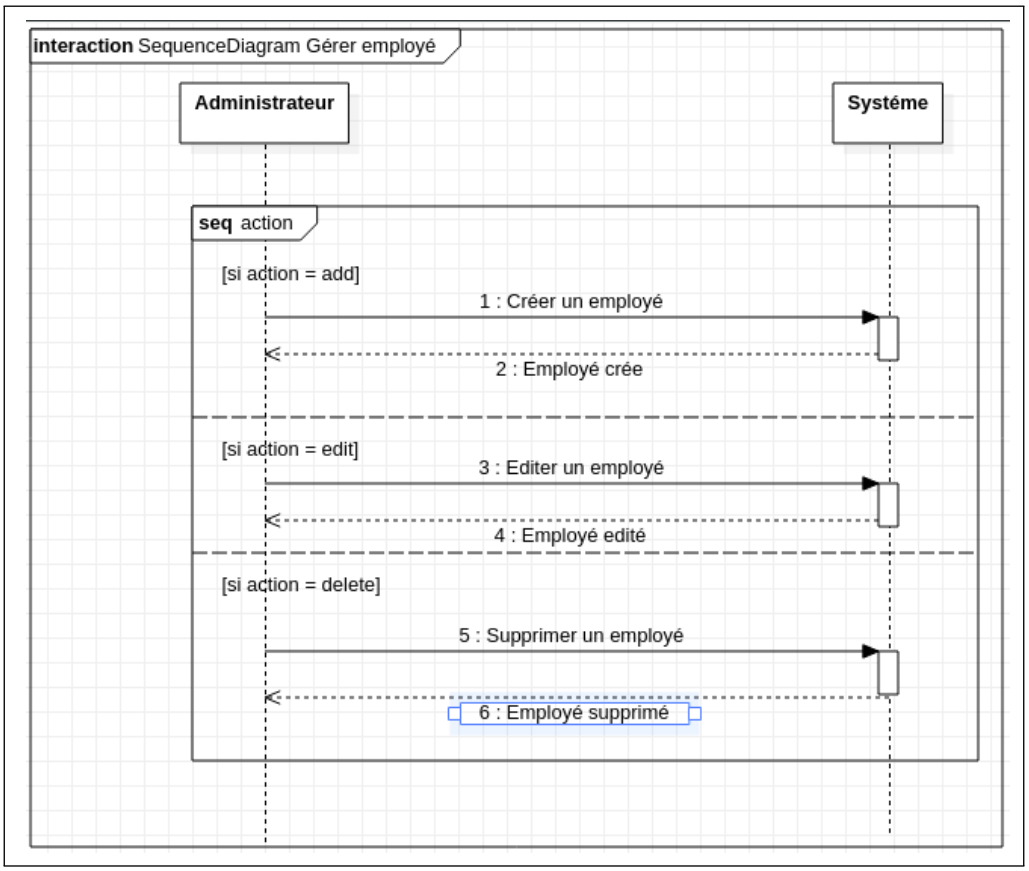

FIGURE 5.3: Sequence diagram of the "Manage employees" scenario

5.3 Analysis and Design of Sprint 2:

In this part, we establish the refinements of the different use cases of the first sprint and some design diagrams in order to deliver a description on the different possible scenarios.

5.3.1 Analysis of the "Manage departments" case:

The system ensures the identity of the user through his login and password.

5.3.2 Refined "Manage Departments" Use Case Diagram

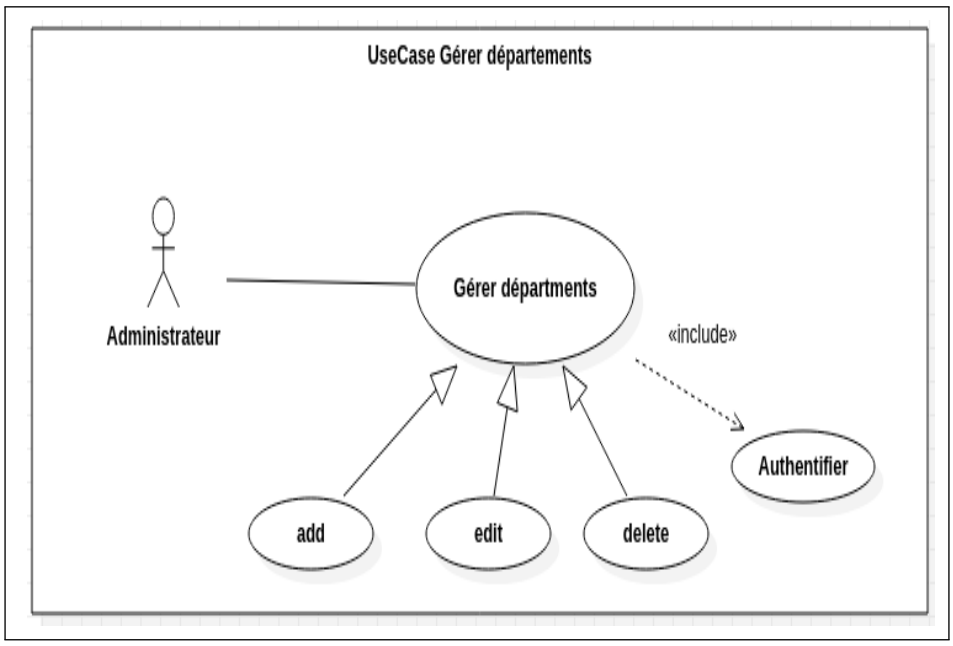

FIGURE 5.4: Refined Use Case Diagram: Manage Departments

5.3.3 Text description

In this part we present the textual description which contains the "Manage all employees" use case scenarios.

SOMMAIRE Gérer les départements	
Titre	Gérer les départements
But	Le système doit gérer les départements
Acteur	Administrateur
DESCRIPTION DES ENCHAINEMENTS	
Pré conditions	
Chaque utilisateur doit avoir un compte • L'administrateur doit s'authentifier	
Scénario nominal	
Consultation 1. L'administrateur clique sur « Manage departments » 2. Le système affiche les différents départements 3. L'administrateur peut ajouter, modifier ou supprimer un département	
Enchainements alternatifs	
Connexion interrompue	

FIGURE 5.5: Refined Use Case Diagram: Manage Departments

5.3.4 Sequence diagram of the "Manage departments" scenario

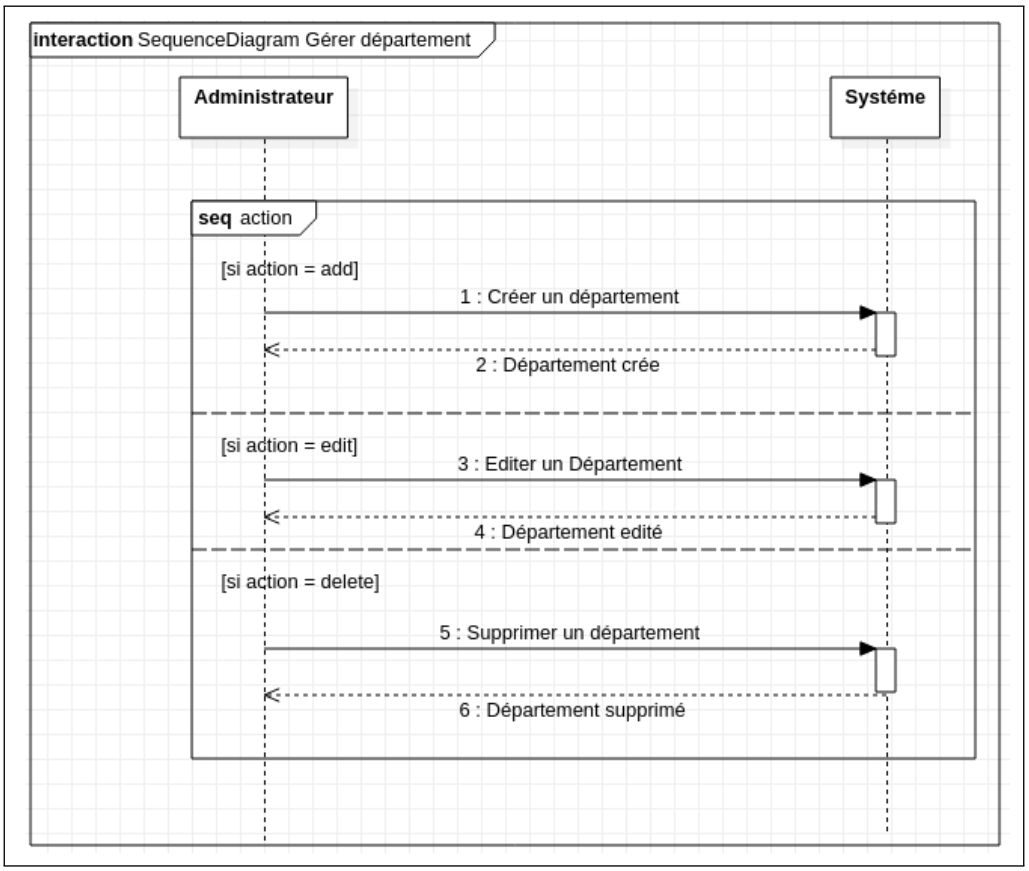

FIGURE 5.6: Sequence diagram of the "Manage departments" scenario

5.4 Analysis and Design of Sprint 3:

In this part, we establish the refinements of the different use cases of the first sprint and some design diagrams in order to deliver a description on the different possible scenarios.

5.4.1 Analysis of the "Manage rentals" case:

The system ensures the identity of the user through his login and password.

5.4.2 Refined "Manage Rentals" Use Case Diagram

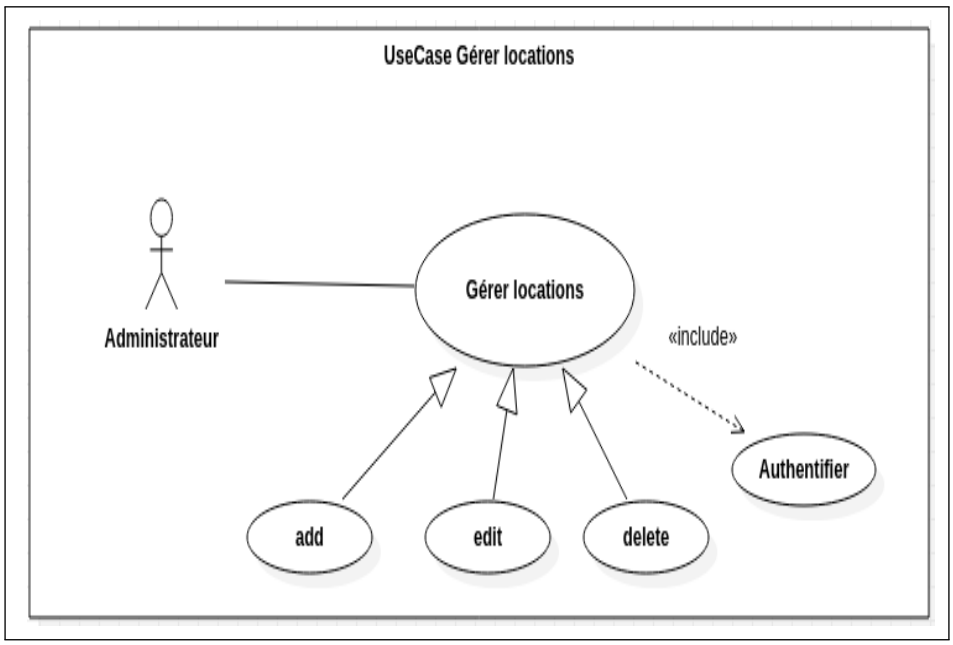

FIGURE 5.7: Refined Use Case Diagram: Manage Rentals

5.4.3 Text description

In this part we present the textual description that contains the "Manage Rentals" use case scenarios.

SOMMAIRE Gérer les locations	
Titre	Gérer les locations
But	Le système doit gérer les locations
Acteur	Administrateur

DESCRIPTION DES ENCHAINEMENTS

Pré conditions

Chaque utilisateur doit avoir un compte

- L'administrateur doit s'authentifier

Scénario nominal

Consultation

1. L'administrateur clique sur « Manage locations »
2. Le système affiche les différentes locations
3. L'administrateur peut ajouter, modifier ou supprimer une location

Enchainements alternatifs

Connexion interrompue

FIGURE 5.8: Refined Use Case Diagram: Manage Rentals

5.4.4 Sequence diagram of the "Manage tenancies" scenario

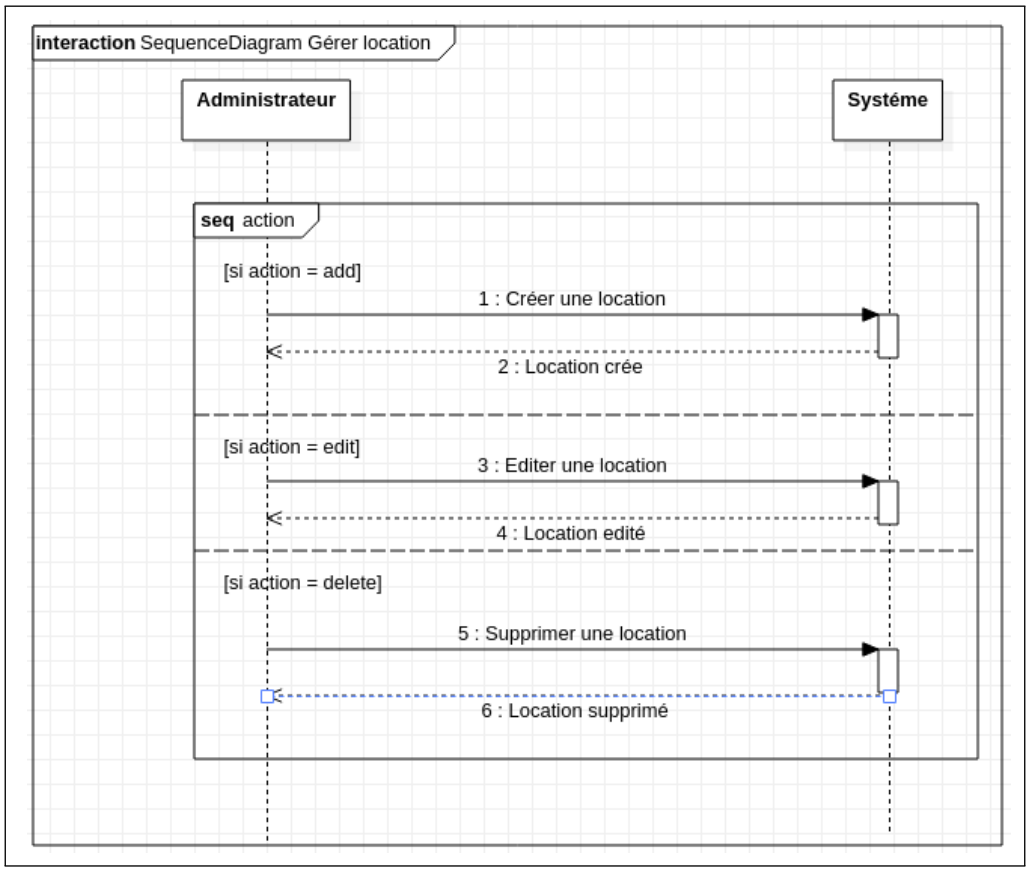

FIGURE 5.9: Sequence diagram of the "Manage rentals" scenario

5.5 Realization

In this section, we present the different interfaces relating to the third release.

Home page :

The user can go to the home page and click on the button "Are you an admin?"

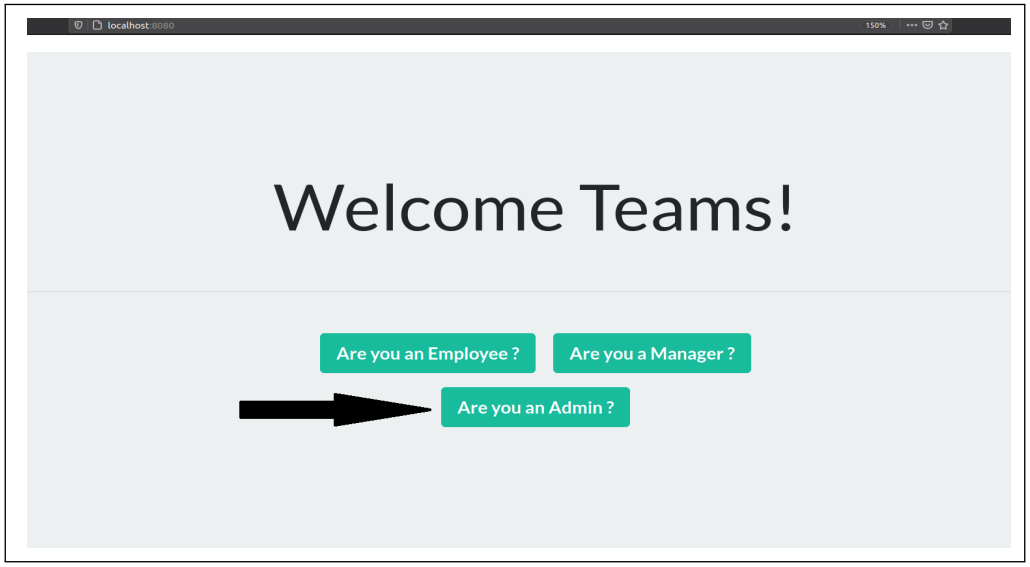

FIGURE 5.10: Home page.

Home page :

The administrator can manage departments, locations or employees.

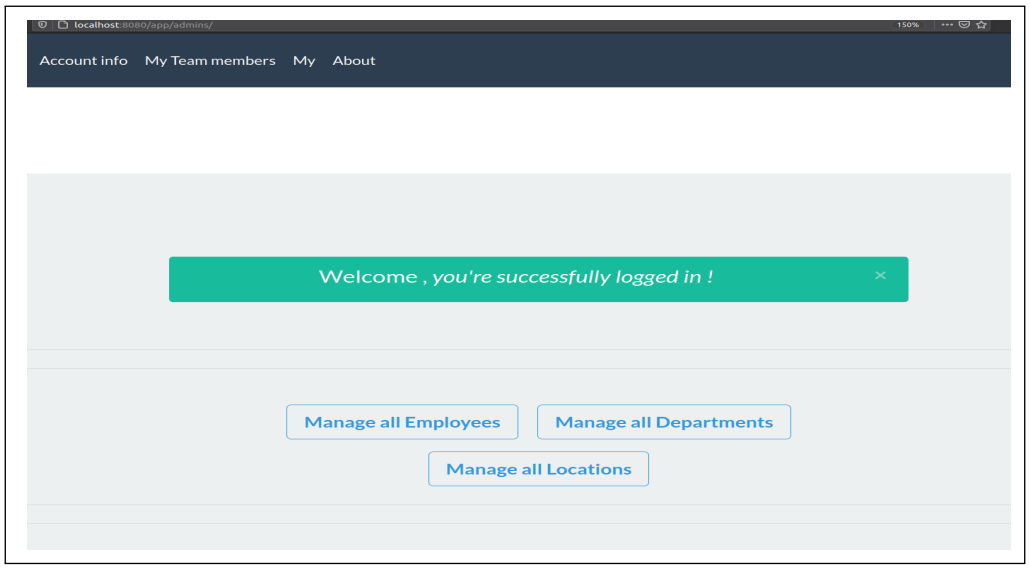

FIGURE 5.11: Home page.

List of rentals:

The interface that shows the locations and the actions to be taken for each location.

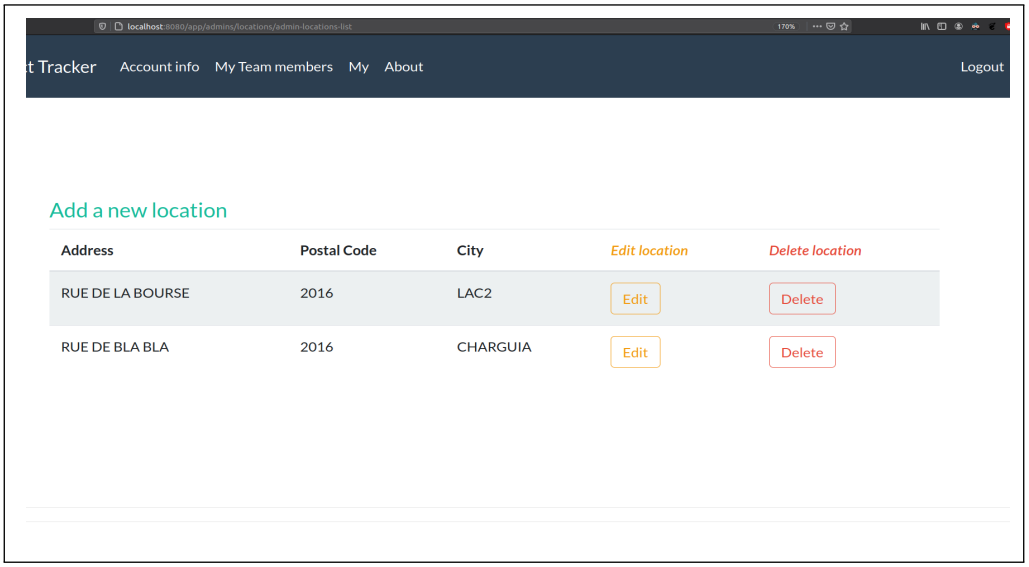

FIGURE 5.12: List of rentals.

Add a location:

The administrator clicks on the button "Add new location" then fill in the information of the location to add. a message returned by the system con fi rms the addition.

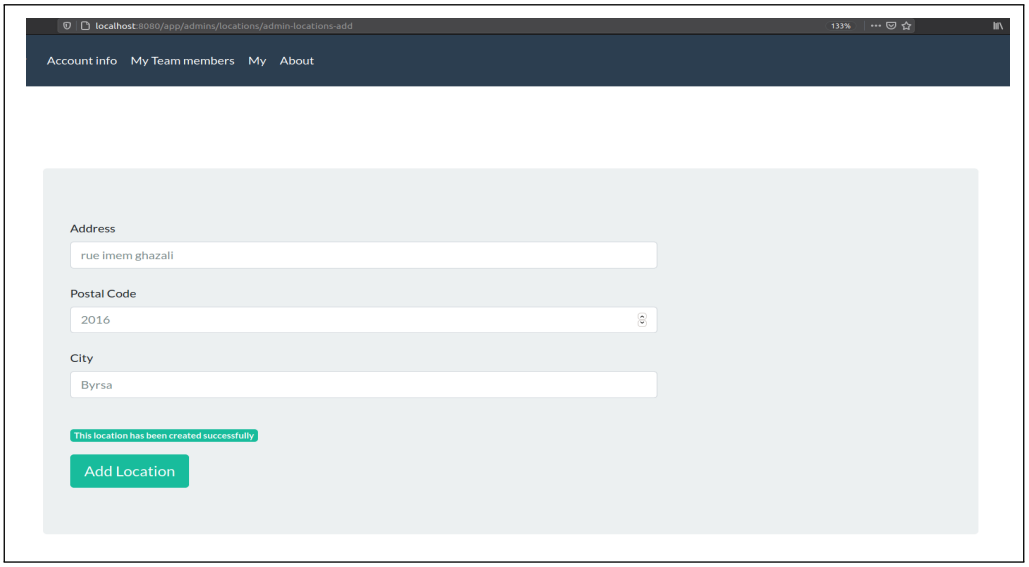

FIGURE 5.13: Add a location.

Modify a rental:

The administrator clicks on the "Edit" button then modi fi es the information of a location. a message returned by the system con fi rms the modification action.

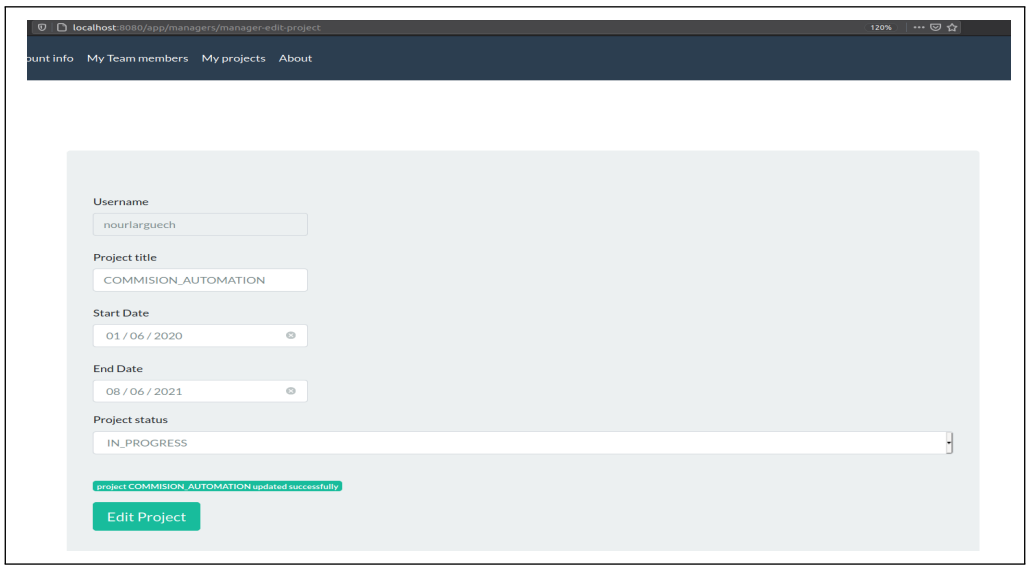

FIGURE 5.14: Modify a rental.

Delete a rental:

The administrator clicks on the "Delete" button, a pop-up is displayed for con fi rmation.

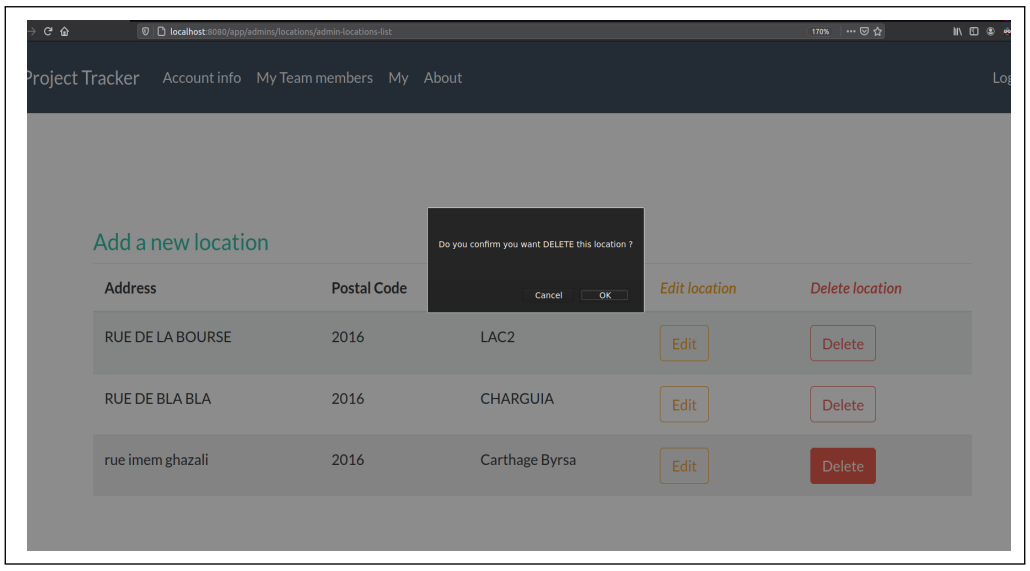

FIGURE 5.15: Delete a rental.

List of departments:

The interface that shows the departments and the actions to be done for each department.

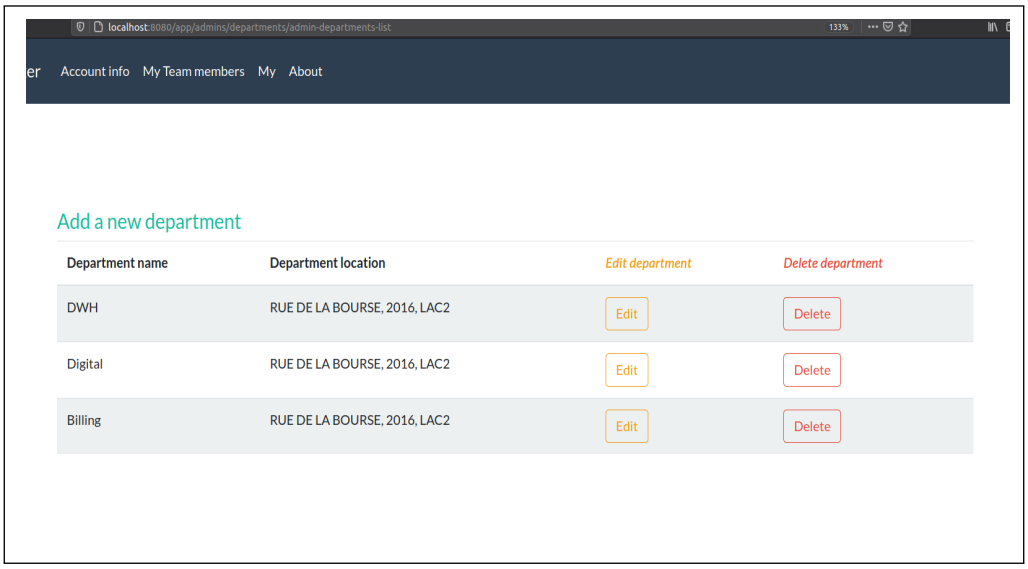

FIGURE 5.16: Home page.

Add a department:

The administrator clicks on the button "Add new department" then fill in the information of the department to add. a message returned by the system con fi rms the addition.

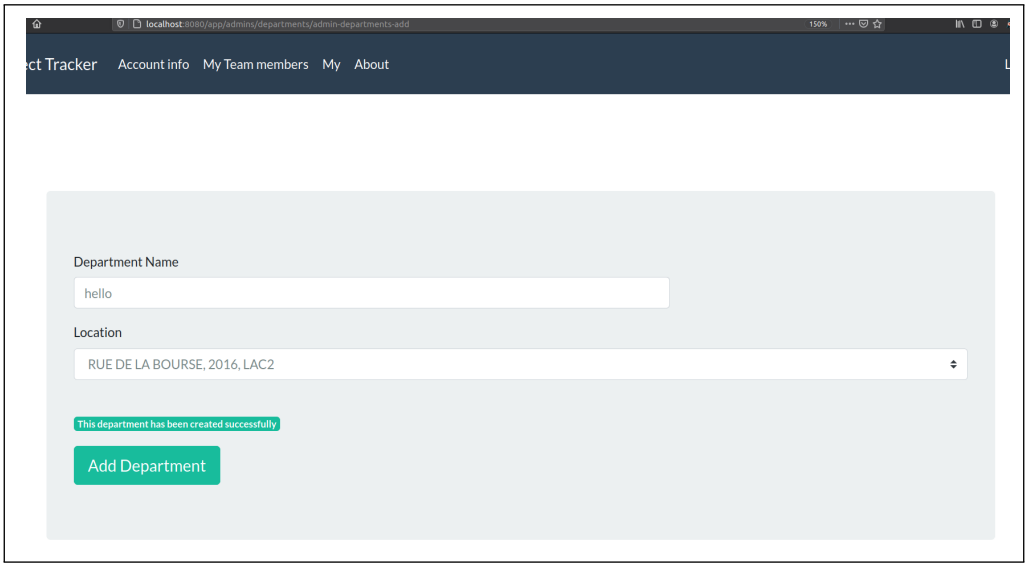

FIGURE 5.17: Add a department.

Modify a department:

The administrator clicks on the "Edit" button then modi fi es the information of a department. a message returned by the system con fi rms the modification action.

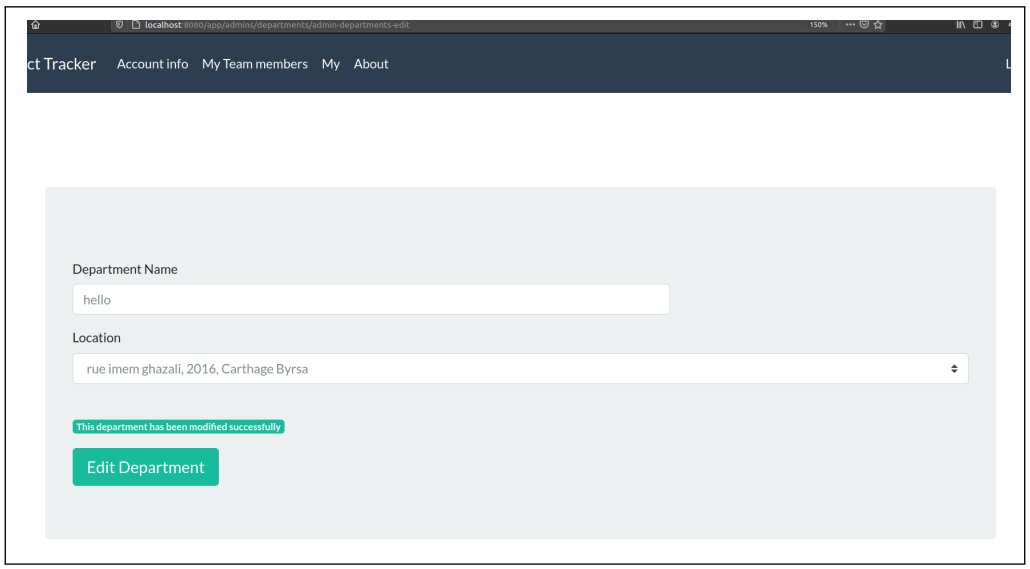

FIGURE 5.18: Modify a department.

Delete a department:

The administrator clicks on the "Delete" button, a pop-up is displayed for con fi rmation.

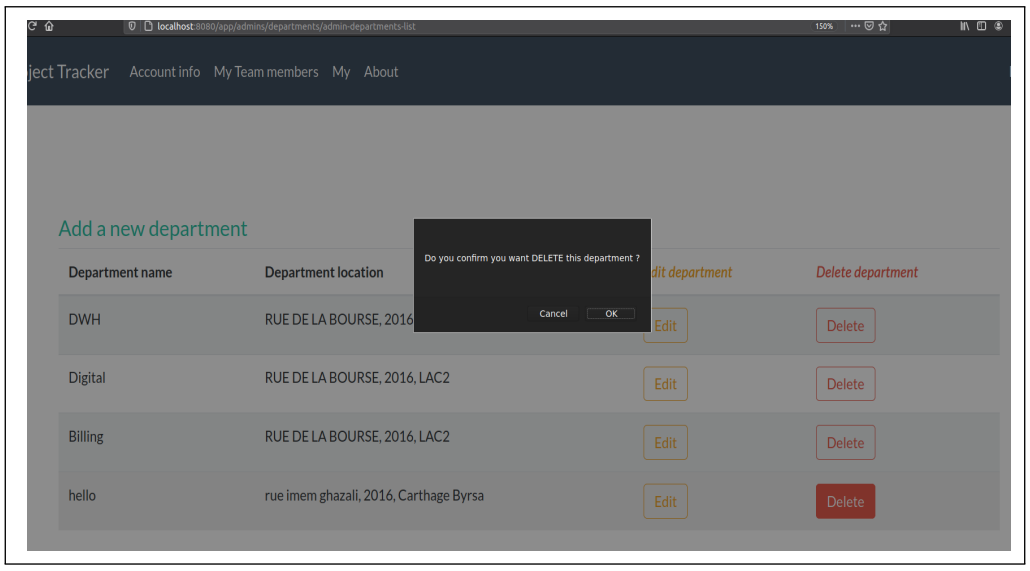

FIGURE 5.19: Home page.

List of employees:

The interface that shows the employees and the actions to be done for each employee.

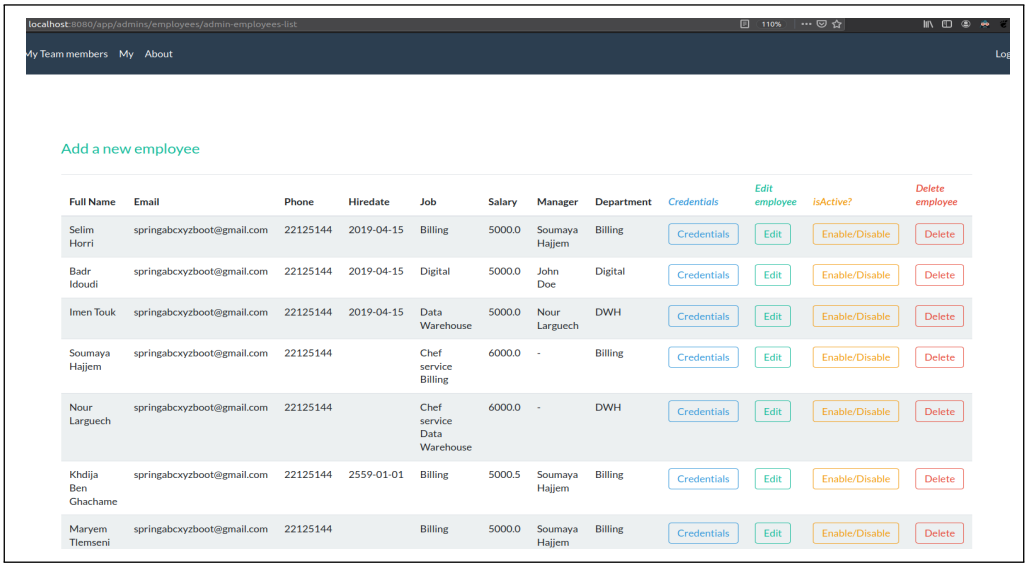

FIGURE 5.20: List of employees.

Add an employee:

The administrator clicks on the button "Add new employee" then fill in the information of the employee to add. a message returned by the system con fi rms the addition.

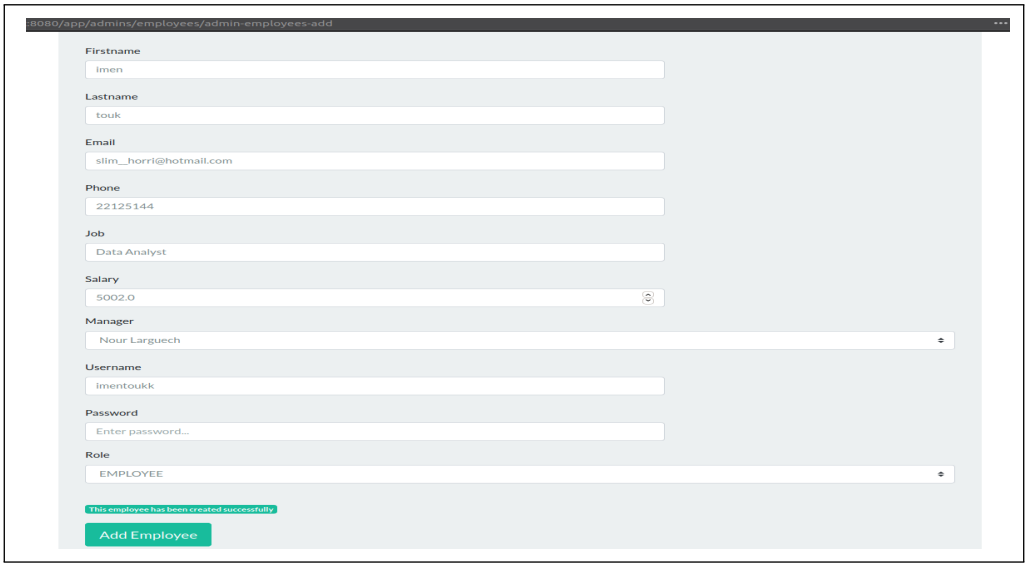

FIGURE 5.21: Add an employee.

Modify an employee:

The administrator clicks on the "Edit" button then modi fi es the information of an employee. a message returned by the system con fi rms the modification action.

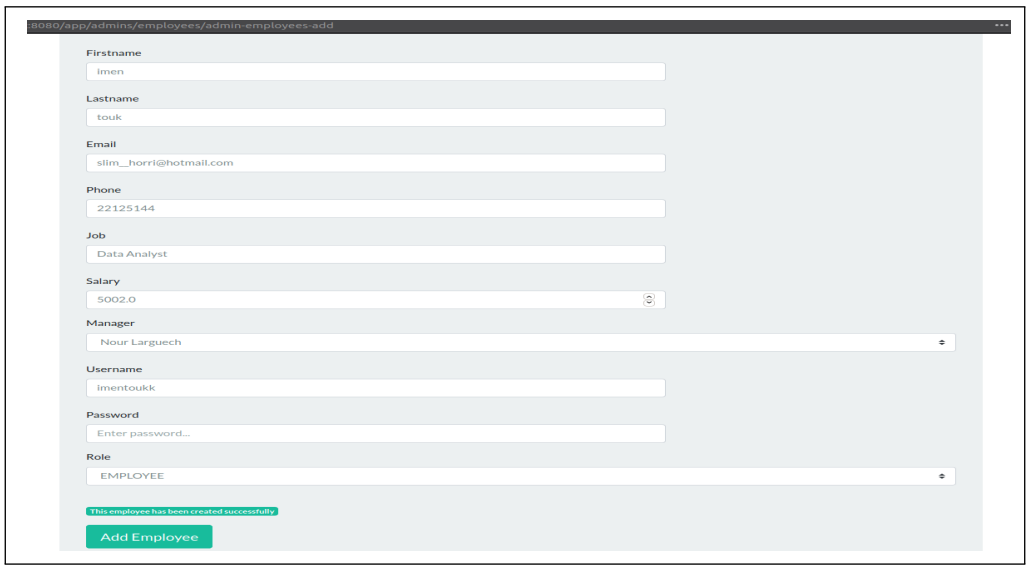

FIGURE 5.22: Add an employee.

Deactivate an employee:

The administrator clicks on the "Enable / Disable" button, a pop-up is displayed for con fi rmation.

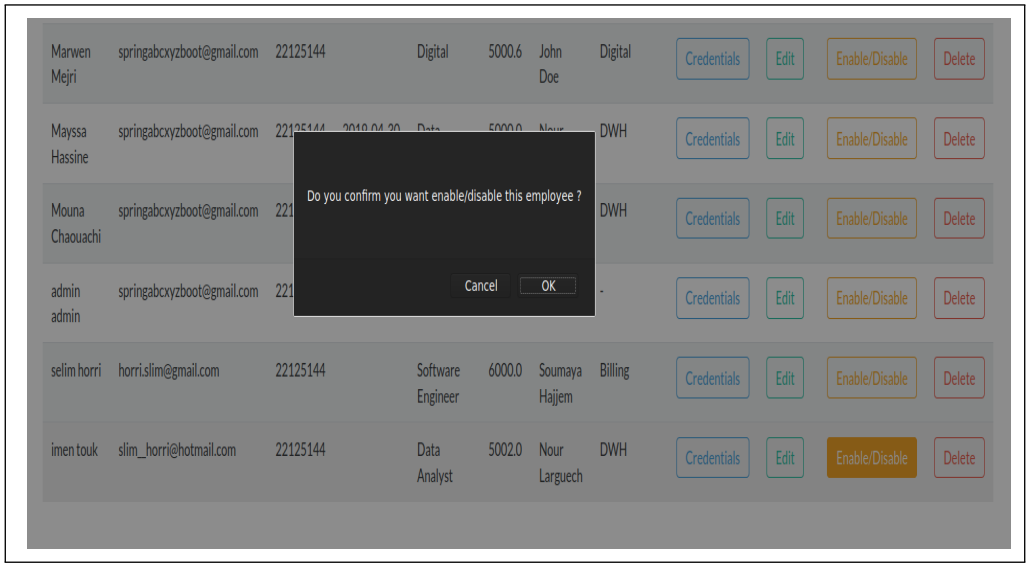

FIGURE 5.23: Home page.

Remove an employee:

The administrator clicks on the "Delete" button, a pop-up is displayed for con fi rmation.

FIGURE 5.24: Home page.

5.6 Conclusion

Throughout this chapter, we have presented the Third Release Sprint Backlog. Then, we put the accent on its speci fi cation and its design then its realization.

GENERAL CONCLUSION

This report presented the work carried out within the framework of an end of semester report. This project focuses on the design and implementation of the "Project Tracker" platform

For the realization of this system, a preliminary study of the applications offering the same service was very useful to identify the functionalities to be implemented and to define the speci fi cations of our application.

The UML method, used in the design phase, allowed us to model our system well so that its understanding becomes easy.

On a technical level, this project was an opportunity for us to learn and master the Spring Boot framework.

BIBLIOGRAPHY

[1] **Java.** Learning Java . Available on : https: //www.oreilly.com/library/view/learning-java/1565927184/

[2] **Design Patterns.** Head First Design Patterns. Available at https: //www.oreilly.com/ : library/view/head- fi rst-design / 0596007124 / :: text = Using

WEBOGRAPHY

[1] **Java** https: //www.java.com/fr/

[2] **Tomcat** http: //tomcat.apache.org/

[3] **Spring Boot** https: //spring.io/projects/spring-boot

[4] **Apache Maven** https: //maven.apache.org

[5] **JUnit** https: //www.jmdoudoux.fr/java/dej/chap-junit.htm

[6] **H2** https: //www.h2database.com/

[7] **Docker** https: //www.docker.com/

[8] **Mysql** https: //www.mysql.com/fr/

[9] **Git** https: //git-scm.com/