TUNISIAN REPUBLIC
*****
MINISTRY OF HIGHER EDUCATION, SCIENTIFIC RESEARCH AND TECHNOLOGY
*****
GENERAL DIRECTORATE OF TECHNOLOGICAL STUDIES
*****
HIGHER INSTITUTE OF TECHNOLOGICAL STUDIES OF CHARGUIA
*****
COMPUTER TECHNOLOGIES DEPARTMENT

END OF STUDIES PROJECT REPORT

Elaborated for the purpose of obtaining a diploma of

applied license in computer technologies

Option

Development of Information Systems

Subject

Improvement of the man-machine interface of existing finance softwares by a chatbot

Elaborated by: Yakoubi Iheb

Layouni Yassine

Pedagogical supervisor: Mme. Jebli khouloud

Company supervisor: Mr. Fayachi Ahmed

Company: PROSOFT INTERNATIONAL

Academic year: 2020/2021

Dedications

We humbly dedicate this work alongside our efforts to

Our fathers

for their support, affection and trust…

Our mothers

for their endless love, encouragement and sacrifice…

Our brothers and sisters

for their unwavering moral support…

To all our dear friends who have made our lives brighter by just being in it…

Yakoubi Iheb

Layouni Yassine

Acknowledgments

First of all, before we start presenting our report, we would like to express our extreme gratitude and thanks to those who have supported us, encouraged us and inspired us from near and far.

For this reason, we want to thank all of our professors for delivering such a great quality of education for the past three years, especially during the pandemic, they always find a way to keep us motivated and pushing us forward.

we would also like to thank our academic supervisor Mme. Jebli Khouloud for her patience, for her time, for always being present whenever we needed her and for her constructive criticisms and her helpful feedbacks.

Also, we would like to thank the Prosoft family for facilitating our integration and providing an amazing experience and atmosphere since day one.

And a special thanks to our professional supervisor Mr. Fayachi Ahmed who was like a mentor to us, always providing us with brilliant ideas and technical support. we feel really lucky to be surrounded by such amazing people. It was really an honor for us to know them and work with them.

And of course, without forgetting the most important and precious people in our lives which are our families and friends for providing love and support since the beginning and never lost faith in us no matter what.

we are who we are today, because of you all. Forever Grateful.

Summary

Contents

List of figures

List of Tables

General Introduction

What differentiates a mature, professional and successful entity from the rest is its ability to communicate effectively with clients. The customer is king and creating customer loyalty means listening to feedback, anticipating their needs or wants, answering their questions and solving their problems with as much little response-time as possible.

Response time is simply the time between when a customer inquiry or contact is received and when it is responded to. Considering this, it's clear that customer service response time is one of the most important factors in determining customer satisfaction.

However, it can be challenging for a growing and thriving business to keep track and respond to every single question and problem especially using time-consuming communication methods such as emails and telephones.

In this context, we were invited to conceive and develop an AI-Powered chatbot that can act as an intermediate between the customers and the financial applications developed by Prosoft international. This chatbot will reduce customer response time and facilitate the user/machine interaction. It will offer customer support, providing information about the application they are using and interpreting user inputs to handle real-time actions manipulating the software automatically.

To achieve this project, we decided to use the Waterfall method.

The content of this report is made up of four chapters:

In the first chapter "Project environment", we are interested in putting our project in its general environment.

In the second part, we start "the requirements specification" where we analyze the functional and non-functional needs of the system and the overall use case diagram.

Then we will continue with the third part "Design" where we view the overall architecture of our project using Diagrams, describing how all the different parts of the system will fit together.

And finally, we will end the report with the last chapter “Realization” illustrating the development process and testing.

Chapter 1
Project Environment

Introduction

In this chapter, we will put the project in its general framework, we will start first by presenting Prosoft international as well as their services, products and clients, afterwards we will take a look at the existing programs evoking their solutions and finally we will choose from certain methodologies the best one that fits with our work.

I. The company Description [N1]

Figure 1: Prosoft International Logo

Our end of studies project was realized within the company Prosoft International, it is a company based in Tunis since 2007. It is specialized in the development of specific software in the field of finance.

Prosoft supports its clients in their IT development projects by offering solutions tailored to their needs and their businesses, adapting an evolutionary approach ensuring that their range of products and services are at the cutting edge of technology and of an unmatched standard.

The company is combined of a strong team made up of a project manager and several highly qualified engineers from major Tunisian schools, with an experienced and well-organized development and maintenance service that will be able to support and advise clients in the realization of their projects.

Services:

- **Business advice / strategy:**

Prosoft supports its clients in developing their projects in order to offer solutions adapted to their needs.

- **Development / technology:**

Prosoft develops tailor-made applications (web, desktop, mobile), the company uses the 3-tier architecture (client, processing (WCF), database (Microsoft SQL Server)).

- **Integration / Maintenance:**

It also provides system integration and provides adaptive and upgradeable maintenance service.

Products:

- **VALORIXZ**

It is the first computer software adapted to the activity of stock market intermediaries: fund management, investment operations, and stock market intermediation.

- **PMS**

PMS is a computer software dedicated to stock market, fund management and investment operations.

Clients:

- TUNISIE VALEURS
- EDC INVESTEMENT CORPORATION
- CGF BOURSE
- PHOENIX CAPITAL MANAGEMENT
- …

II. The Market Research

Before we start our project, it is important to have an idea about the existing projects and their solutions.

According to an online survey created by DRIFT [N2] in 2018, they asked a 1000+ participants to think about what frustrations they experience with online services such as (search engines, messaging applications, product/service websites and mobile applications).

The most common frustrations reported by consumers included:

- Websites being hard to navigate (34%).
- Not being able to get answers to simple questions (31%).
- Not being able to get basic details about a business, such as address, hours of operation, phone number… (28%).

The takeaway is that in the on-demand, real-time world we live in, where everything just seems to be one click away, consumers expect to be able to find the information they're looking for quickly and easily.

1. The International Solutions

- **FAQs (Frequently asked questions)** forum is often used in articles, websites, email lists, and online forums where common questions tend to recur, for example through posts or queries by new users related to common knowledge gaps. The purpose of an FAQ is generally to provide information on frequent questions or concerns.

- **IVR (Interactive Voice Response)** is a technology that allows humans to interact with telephone systems via DTMF (Dual Tone Multi-Frequency) entered via the keypad or by voice recognition.

For example, when you call a business, you hear "type 1 for technical assistance, type 2 for sales assistance, type 0 to speak with an advisor." Once you have made your choice, your call is routed to an extension / call queue, or a voice message can be spoken. You may also be taken to a submenu which is just a second voice server offering more options that you need to select again.

Figure 2: Interactive Voice Response

- **Chatbot:** A chatbot, also called a “conversational agent”, is a computer program capable of simulating a conversation with one or more humans by voice or textual exchange. This tool is now widely used on the Internet by the customer services of brands or online merchants through instant messaging.

History of the chatbot:

Historically, the first chatbot named Eliza was created in 1966 by Joseph Weizenbaum, professor at MIT (*Massachusetts Institute of Technology*), in the United States. The program, which simulated a Rogerian psychotherapist, reformulated most of his interviewer's statements into questions he asked him in return.

Figure 3: Eliza the first chatbot

Different Types of chatbots [N3]:

Rule-based Chatbots: With this type of bot, communication is through pre-set rules. User input must conform to these pre-defined rules in order to get an answer. Often with such bots only buttons are used. Since no artificial intelligence is used here, an open conversation with this type of bot is not possible or very limited.

AI Chatbots: Also known as NLP (Natural Language Processing) chatbots, the idea behind this is to make robots “think like humans” using features such as logical reasoning, planning and understanding languages. Understanding languages is especially useful when it comes to chatbots. Unlike the rule-based bots, these bots use algorithms (neural networks) to process natural language.

2. The Current Solution at Prosoft International

The users of Prosoft international’s applications have to navigate and interact with the system directly, and if they have any problems or questions about the software, they are obliged to send an email or phone a Prosoft employee.

3. Criticizing the current solution

The current solution can have a few advantages:

- Direct contact with the company.
- The answers are precise.

However, this solution can also have a few disadvantages:

- Number of Hours of service is limited.
- The response to simple questions is not instant.
- Limited customer interaction.
- The Answers are not consistent.
- Costs time and patience from both the customer and the Prosoft employee.

4. The proposed solution

Our solution is to create an AI-powered chatbot dedicated to the users of Prosoft International's softwares (Bank employees, Stock brokers, …), inspired by one of the international solutions, in order to facilitate the human/machine interactions and reduce customer response time.

To be more flexible and accessible to the users, our solution will have two different versions, mobile and desktop.

Our Project will offer these functionalities:

- Manage user Accounts
- Manage user functions
- Interact with the chatbot in text mode and in French, asking about:
  - Information on the software
  - Manipulating Software automatically (ex: make transaction)
  - Executing queries on the database
  - Generating statistics
- Visualize the result of queries or statistics interpreted by the chatbot
- Manage the rules of use of the chatbot
- Export Data returned by the chatbot
- Consult Chatbot history
- Consult chatbot statistics

III. The design methodology and the modeling language

A project management methodology is defined as a system of methods, principles, and rules for managing a project. It can help standardizing processes, building a common language and understanding how to manage a project.

It helps project managers reduce risks, avoid duplication of efforts and to ultimately increase the impact of the project.

Therefore, in our project, we have traced and followed a meticulous design methodology, which allowed us to better analyze and manage the different phases of realization of the application.

1. The design methodology

Before we choose a specific method of development, first we have to make a comparison between them, analyzing the advantages and disadvantages of each one, then determining the method that is most compatible with the context of our project.

	Definition	Strong Points	Weak Points
V-model [N4]	The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as Verification and Validation model. The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage.	• Easy to use • Simple to use • Testing activities like planning, test designing happens well before coding. • Time saving, quick • Hence higher chance of success over the waterfall model • Proactive defect tracking – that is defects are found at early stage • Avoids the downward flow of the defects • Works well for small projects where requirements are easily understood	• Very rigid • Least flexible • Software is developed during the implementation phase, so no early prototypes of the software are produced • If any changes happen in midway, then the test documents along with requirement documents has to be updated • Risky

Scrum [N5]	Scrum is a lightweight framework that helps people, teams and organizations generate value through adaptive and iterative solutions for complex problems.	• Scrum can help teams complete project deliverables quickly and efficiently • Scrum ensures effective use of time and money • Large projects are divided into easily manageable sprints • Developments are coded and tested during the sprint review • Works well for fast-moving development projects • The team gets clear visibility through scrum meetings • Scrum, being agile, adopts feedback from customers and stakeholders • Short sprints enable changes based on feedback a lot more easily • The individual effort of each team member is visible during daily scrum meetings	• The chances of project failure are high if individuals aren't very committed or cooperative • Adopting the Scrum framework in large teams is challenging • The framework can be successful only with experienced team members • Daily meetings sometimes frustrate t eam members • If any team member leaves in the middle of a project, it can have a huge negative impact on the project • Quality is hard to implement until the team goes through an aggressive testing process

Waterfall [N6]	WATERFALL MODEL is a sequential model that divides software development into pre-defined phases. Each phase must be completed before the next phase can begin with no overlap between the phases. Each phase is designed for performing specific activity during the SDLC phase.	• Before the next phase of development, each phase must be completed • Suited for smaller projects where requirements are well defined • They should perform quality assurance test (Verification and Validation) before completing each stage • Elaborate documentation is done at every phase of the software's development cycle • Project is completely dependent on project team with minimum client intervention • Any changes in software is made during the process of the development	• It is not desirable for complex project where requirement changes frequently • Error can be fixed only during the phase • Testing period comes quite late in the developmental process • Documentation occupies a lot of time of developers and testers • Clients valuable feedback cannot be included with ongoing development phase • Small changes or errors that arise in the completed software may cause a lot of problems

Table 1: Comparison of methodologies

Our choice:

To achieve the objectives of our project, we decided to adopt the Waterfall model. This model seemed to us to be the most suitable solution for our project because:

- Requirements are not changing frequently
- Requirements are clear
- Environment is stable
- Technology and tools are stable
- Resources are available

2. The Presentation of the waterfall model [N7]

Waterfall approach was the first SDLC Model (Software Development Life Cycle Model) to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

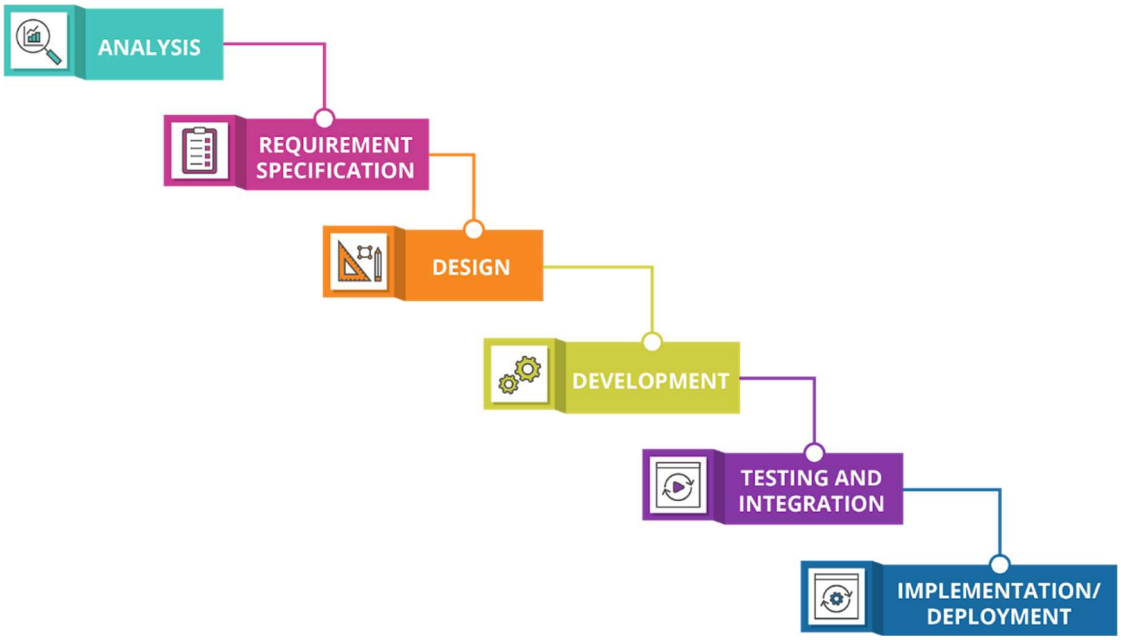

Figure 4: Waterfall model

3. The Modeling Language [N8]

To model and design our project, we need a unified modeling language, we decided to use UML

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing object-oriented software and the software development process.

This modeling language uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

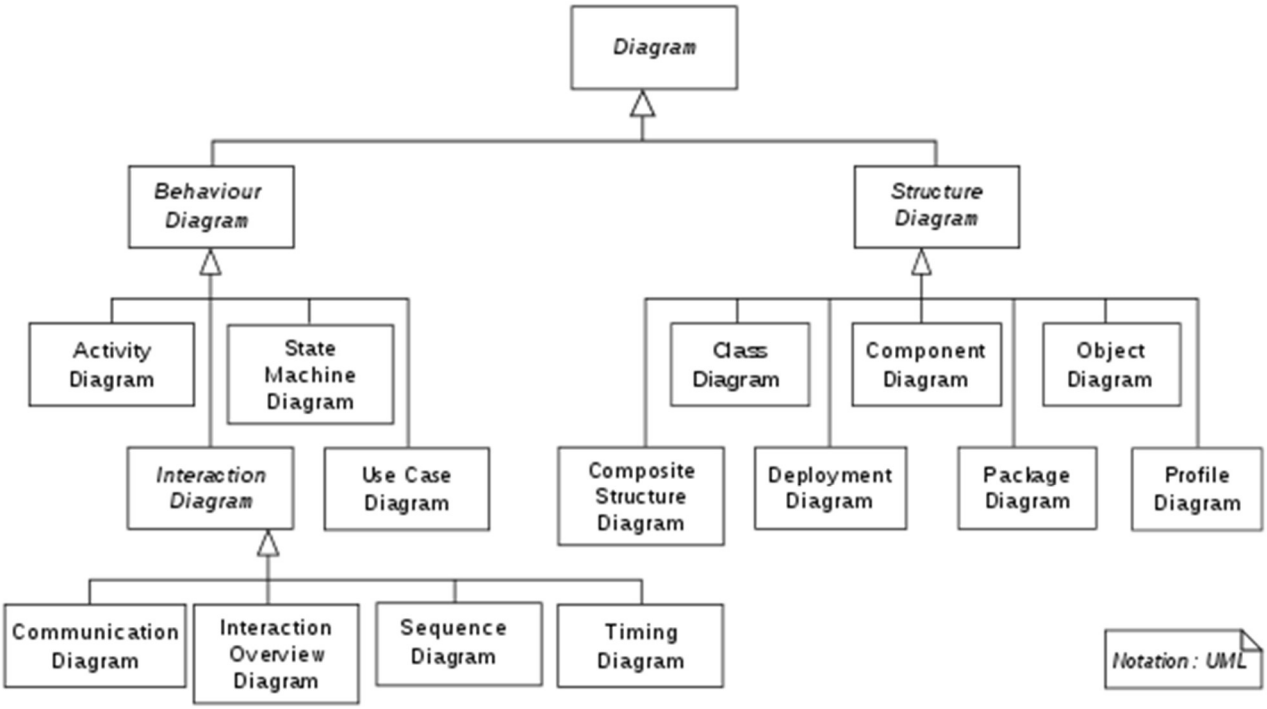

Figure 5: UML Diagrams

IV. The Provisional planning

To properly organize and plan the provisional time consumption of the different phases of our project we decided to use the Gantt chart (using instagantt.com). A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis.

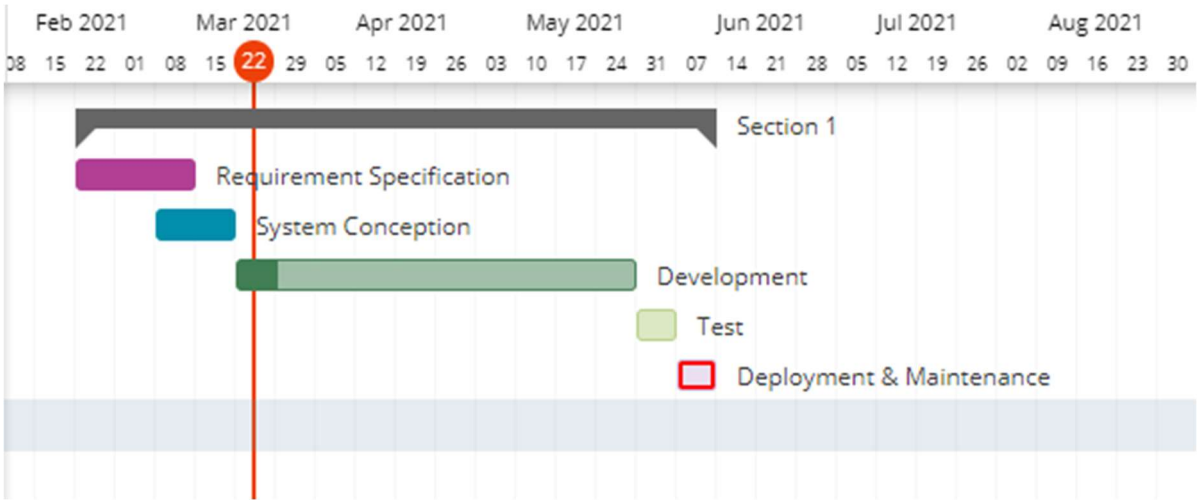

Figure 6: Gantt Chart

Conclusion

In this chapter, we have presented the company, the market research and the chosen methodology for the development of our project. In the following chapter we will describe the different requirements and functions of this project.

Chapter 2
Requirements Specifications

Introduction

To succeed in a project, it is mandatory to determine the analysis and the specification needs. Being the first essential step in the waterfall life cycle of a software, it is fundamental to properly understand the requirements expressed by the user. In order to understand the context of the application we will, during this step, determine the functional needs, non-functional needs as well as the different actors who will interact with our application using the use case diagram.

I. The functional requirements

The functional requirements define all the services expected by the system. In the following, we describe the different functional needs of our system:

1.1. Manage user accounts

The administrator can consult all existing user accounts, update, delete or add a new user account.

The roles for the system user accounts are:

- Administrator
- Employee

1.2. Manage user functions

Each user has a specific function within the organization (project manager, department manager, …).

The administrator can consult all existing user functions, update, delete or add a new user function.

1.3. Interact with the chatbot

The user can interact with the chatbot in text mode and in French.

The user can ask about information on the system they are working on, ask to manipulate the system automatically, to execute queries or to generate statistics.

1.4. Consult the data returned by the chatbot

The user can consult statistics and the result of queries returned by the chatbot.

1.5. Export the data returned by the chatbot

The user can export and download the data returned by the chatbot in multiple formats (Excel, PDF, Word).

1.6. Consult chatbot history

The user can consult all the interactions between him and the chatbot.

1.7. Consult chatbot statistics

The administrator can consult the statistics of the chatbot (number of messages per day, most executed rules…).

1.8. Manage the rules of use of the chatbot

The rules of use are predictions of what the user might want and the chatbot's responses for them.

The rules of use are composed of:

- Title: the title of the rule.
- Trigger(s): what initiates the rule ex:" Show Portfolio".
- Response(s): the chatbot's response to the trigger(s).
- Sequence message(s): a sequence of questions about the parameters of the api (Application Programming Interface) ex: "User: display User", "Chatbot: username?".
- Type: type of the rule (conversation or task)
- Method: method of the api if it is a task (GET, POST, PUT, DELETE)
- Api: api (Application Programming Interface) of the rule if it is a task.
- Response type: the format of the response if it is a task (matrix, statistics, …).

II. The non-functional requirements

These are the requirements that characterize the system in terms of performance, type of material or type of design.

In the following, we describe the different non-functional needs of our system:

- **Availability:**

it is necessary that the application remains always in service to interact with the users at any given moment.

- **Security:**

Since this platform contains confidential data, all access to different areas must be protected by a password and a privileged access, and the users' password must be encrypted before saved in the database. Also, service calls for both the backend and the chatbot require token validation.

- **Performance:**

The fact that the main core of the application is a chatbot, customer response time is crucial, so the interpretation of data must be fast and response time must be fluid.

- **Ergonomics:**

Ergonomics is the use of a simple and easy to understand user interface, so that navigating the application is not ambiguous to any user.

- **Interoperability:**

The user doesn't need to install anything to be able to use this application, just a web browser, in addition it is multiplatform.

- **Reusability:**

The application must provide the ability to modify and reuse the code.

III. The Presentation of the use cases

A use case is a written description of how actors or users will perform tasks on the application. It outlines, from a user's point of view, a system's behavior as it responds to a request. Each use case is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled. [N9]

1. The presentation of the actors

The different actors who interact with our system are divided into 2 main categories:

- Employee: his role is to interact with the chatbot, consult data returned by the chatbot, export data returned by the chatbot and consult chatbot history. He can also manage the rules of use of the chatbot and consult chatbot history (with permission from the administrator).
- Administrator: in addition to all employee functions, he can manage user accounts, consult the entire chatbot history and manage the rules of use of the chatbot.

2. The Textual description of the use cases

The use cases are defined by a textual description, describing the objectives and interactions between the system and its actors. They highlight the roles of its users, and help categorize them, define their expectations (system objectives) and obligations (system management) [N26].

In the following we will illustrate the textual description of a few of the use cases present in our project:

- Textual Description of the use case "Authenticate"

Case	Authenticate
Actors	Employee, Administrator
Objectif	Allows The user to access his account using his username and password
Pre-condition(s)	The user needs to be registered in the database
Post-condition(s)	• User authenticated • The main page is displayed
Nominal Scenario	1. The actor opens the application. 2. The system displays the authentication page. 3. The actor enters the username and password. 4. The system checks the existence of the user in the database. 5. The system displays the home page.
Alternative Scenario	• Authentication error: invalid username or password. This sequence starts at point 4. At point5. The system displays an error message. The scenario resumes at point 2.

	• Empty mandatory fields: This sequence starts at point 4. The scenario resumes at point 2.

Table 2: Textual description of the use case "Authenticate"

- Textual Description of the use case “Add user”

Case	Add User
Actor(s)	Administrator
Objectif	Allows The administrator to add a new user account.
Pre-condition(s)	Authentication successful
Post-condition(s)	User account added
Nominal Scenario	1. The actor selects add a new user. 2. The system displays the form to fill. 3. The actor fills and submits the form. 4. The system checks the data. 5. The system saves the new user in the database. 6. The system displays a success message.

Alternative Scenario	• Mandatory fields not valid or empty or the user already exists: This sequence starts at point 4. At point 5 the system displays an error message. The scenario resumes at point 2.

Table 3: Textual Description of the use case "Add User"

- Textual Description of the use case “Interact with the chatbot”

Case	Interact with the chatbot
Actor(s)	Employee, Administrator
Objectif	Allows The user to communicate with the chatbot.
Pre-condition(s)	Authentication successful
Post-condition(s)	A response is returned or a task is performed.
Nominal Scenario	1. The actor sends a message. 2. The system saves the message in the user’s message history 3. The system analyzes the message. 4. The system returns the appropriate response or performs a task.

Alternative Scenario	• The data to perform a task is wrong: This sequence starts at point 3. At point 4 the system displays an error message. • The user doesn't have permission to execute the specific task: This sequence starts at point 3. At point 4 the system displays an error message.

Table 4: Textual Description of the use case "Interact with the chatbot"

- Textual Description of the use case “Add a rule of use of the chatbot”

Case	Add a rule of use of the chatbot
Actor(s)	Employee, Administrator
Objectif	Allows The user to add a new rule of use of the chatbot.
Pre-condition(s)	Authentication successful
Post-condition(s)	Rule of use added.
Nominal Scenario	1. The actor selects add a new rule of use. 2. The system displays the form to fill. 3. The actor fills and submits the form. 4. The system checks the data. 5. The system saves the new rule in the database. 6. The system saves the new rule in the chatbot data file. 7. The chatbot is trained on the data file. 8. The system displays a success message.
Alternative Scenario	• Mandatory fields not valid or empty or the rule already exists: This sequence starts at point 4. At point 5 the system displays an error message. The scenario resumes at point 2.

Table 5: Textual Description of the use case "Add rule of use of the chatbot"

- Textual Description of the use case “Export Data”

Case	Add a rule of use of the chatbot
Actor(s)	Employee, Administrator
Objectif	Allows The user to export and download the data returned by the chatbot.
Pre-condition(s)	Authentication successful System displays the data (stat, matrix, …)
Post-condition(s)	File downloaded.
Nominal Scenario	1. The actor selects export data as pdf, excel or word. 2. The system generates a file. 3. The file is downloaded.

Table 6: Textual Description of the use case "Export Data"

3. The global use case diagram

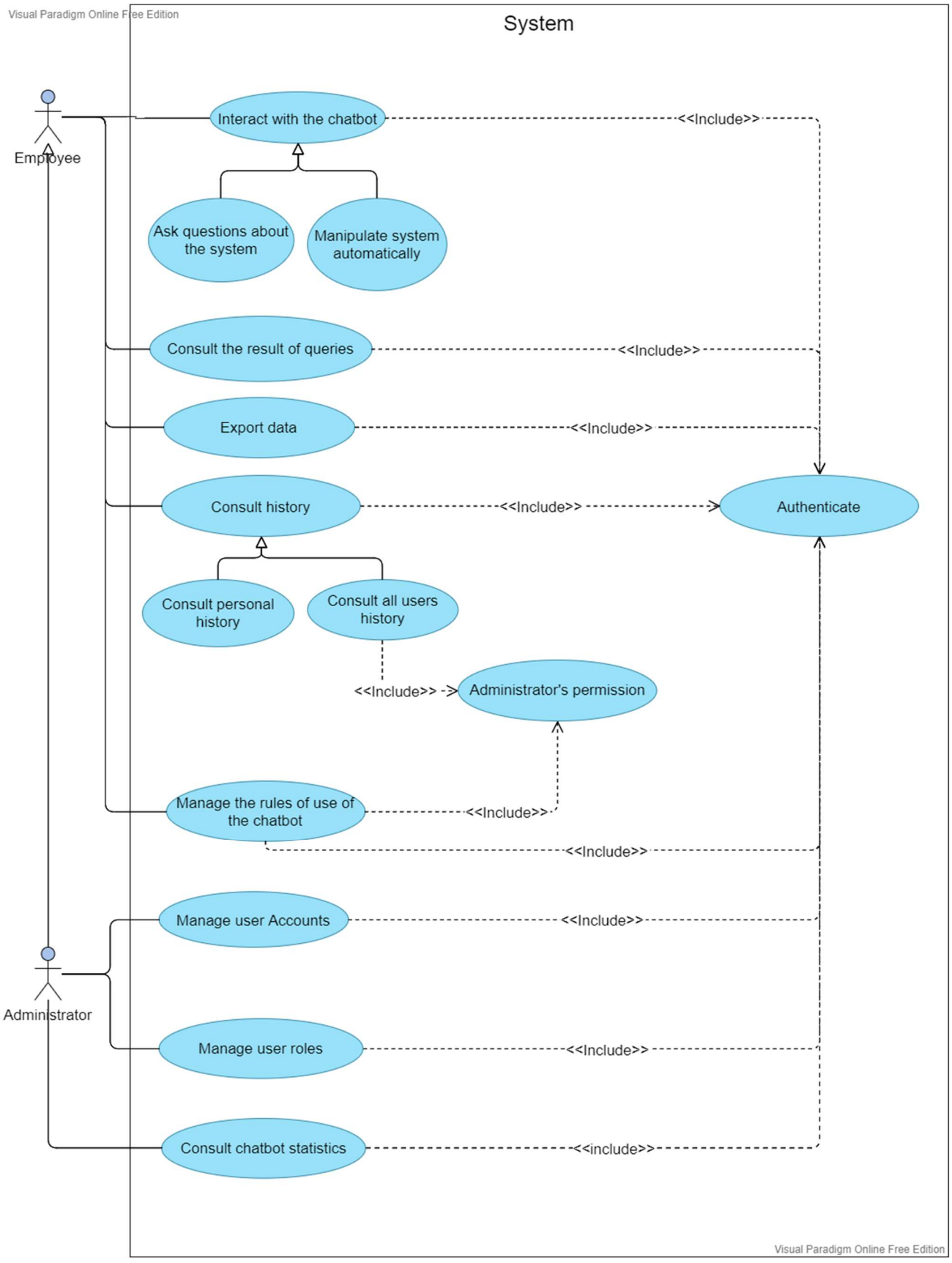

Figure 7 : Global use case diagram

IV. Interface mockups

It is essential before we start the creation of our project that we have a visual idea of what our application would look like. to create the interface mockups, we used **Balsamiq wireframes** which is a user interface design tool for creating wireframes or mockups, mostly used to generate digital sketches of a concept for an application or a website[N21].

- Login interface mockup

this mockup represents the first page of the application, it allows the users to connect to the system.

Figure 8: Login mockup desktop version

- Chatbot interface mockup

this mockup represents the main page of the application, it allows the users to interact with the chatbot.

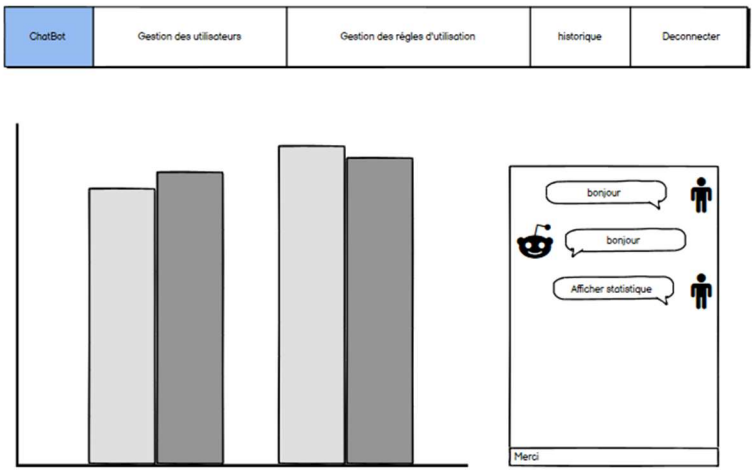

Figure 9: Chatbot mockup desktop version

Figure 10: Chatbot mockup mobile version

- Add rule of use interface mockup

this mockup allows the users to add a new rule of use.

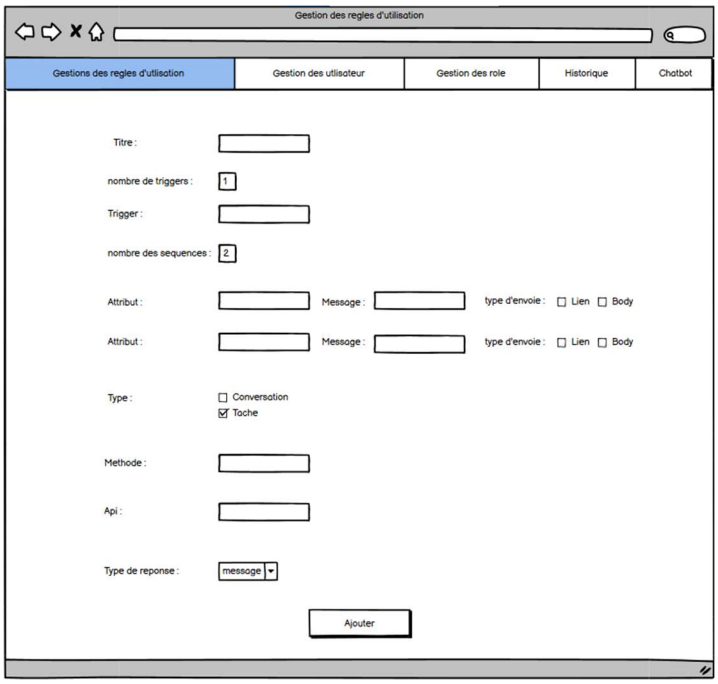

Figure 11: Add rule mockup desktop version

Figure 12: Add rule mockup mobile version

- Update employee interface mockup

this mockup allows the users to update employee accounts.

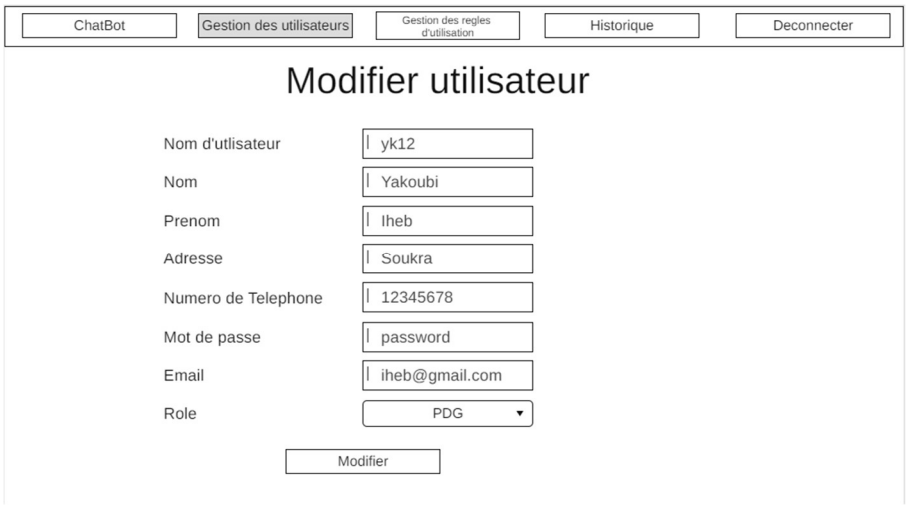

Figure 13: Update user mockup desktop version

Figure 14: Update user mockup mobile version

- Delete employee interface mockup

this mockup allows the users to delete an employee.

Figure 15: Delete user mockup mobile version

- Update function interface mockup

this mockup represents the update function interface, it allows the administrator to update an employee function.

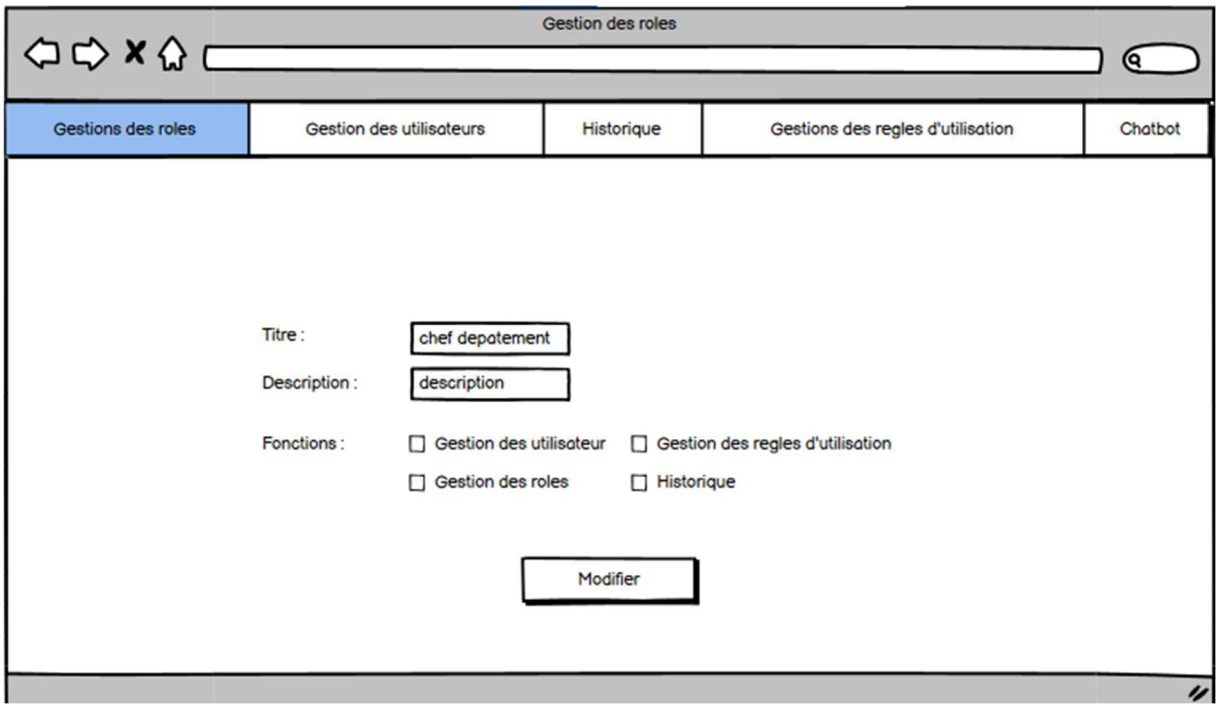

Figure 16: Update function mockup desktop version

Figure 17: Update function mockup mobile version

Conclusion

During this chapter we specified the functional and non-functional requirements of our system and presented the different use cases. In the next chapter we will illustrate our conception for the project.

Chapter 3
Design

Introduction

Now that we have determined the different project requirements, during this chapter we will illustrate how the system will work by presenting our conceptual solution for the project using UML diagrams.

I. The global architecture

1. MVC [23]

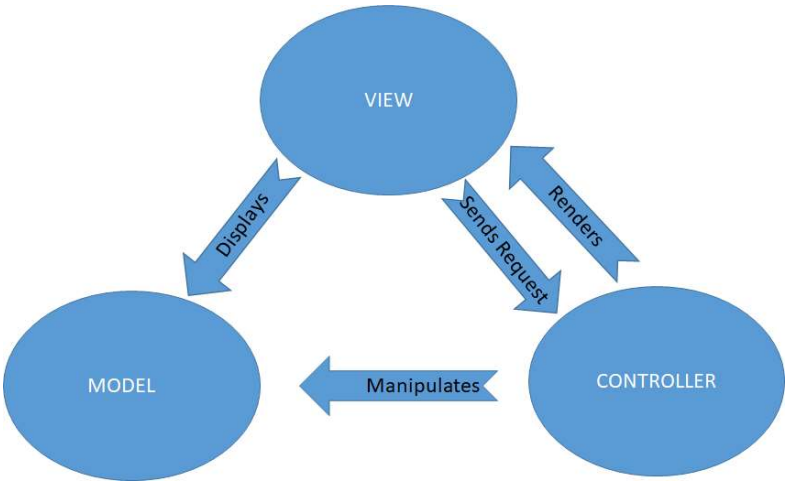

Figure 18: MVC architecture

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

- Model

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data.

- View

The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc.

- Controller

Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output.

2. Physical architecture [N24]

We used a 3-tier architecture:

A 3-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms.

A Presentation Layer that sends content to browsers in the form of HTML/JS/CSS. This might leverage frameworks like React, Angular, Ember, Aurora, ...

An Application Layer that uses an application server and processes the business logic for the application. This might be written in C#, Java, C++, Python, Ruby, ...

A Data Layer which is a database management system that provides access to application data. This could be MSSQL, MySQL, or PostgreSQL, …

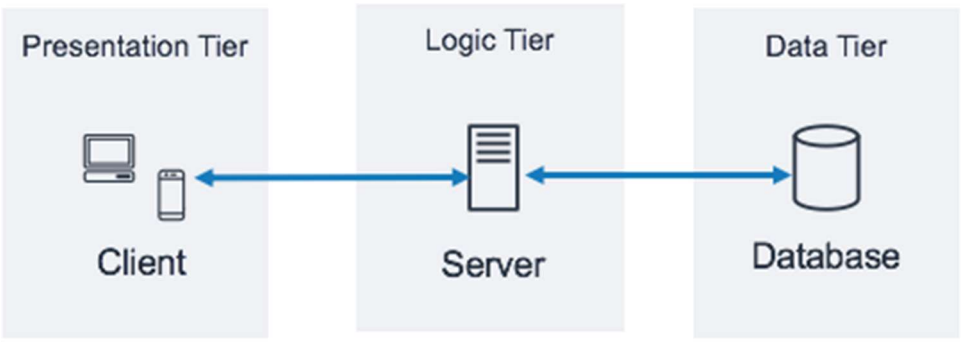

Figure 19: 3-tier architecture

II. The dynamic modeling [N10]

The dynamic model is used to express and model the behaviour of the system over time. It is concerned with the temporal changes and the different states the object passes through during its lifetime.

The sequence Diagrams

Sequence diagrams are used to display the interactions between users, objects and entities within the system. It provides a sequential map of message passing between objects over time. We will illustrate below a few of our project's sequence diagrams:

Add a rule of use

This diagram shows the scenario that occurs when adding a new rule of use.

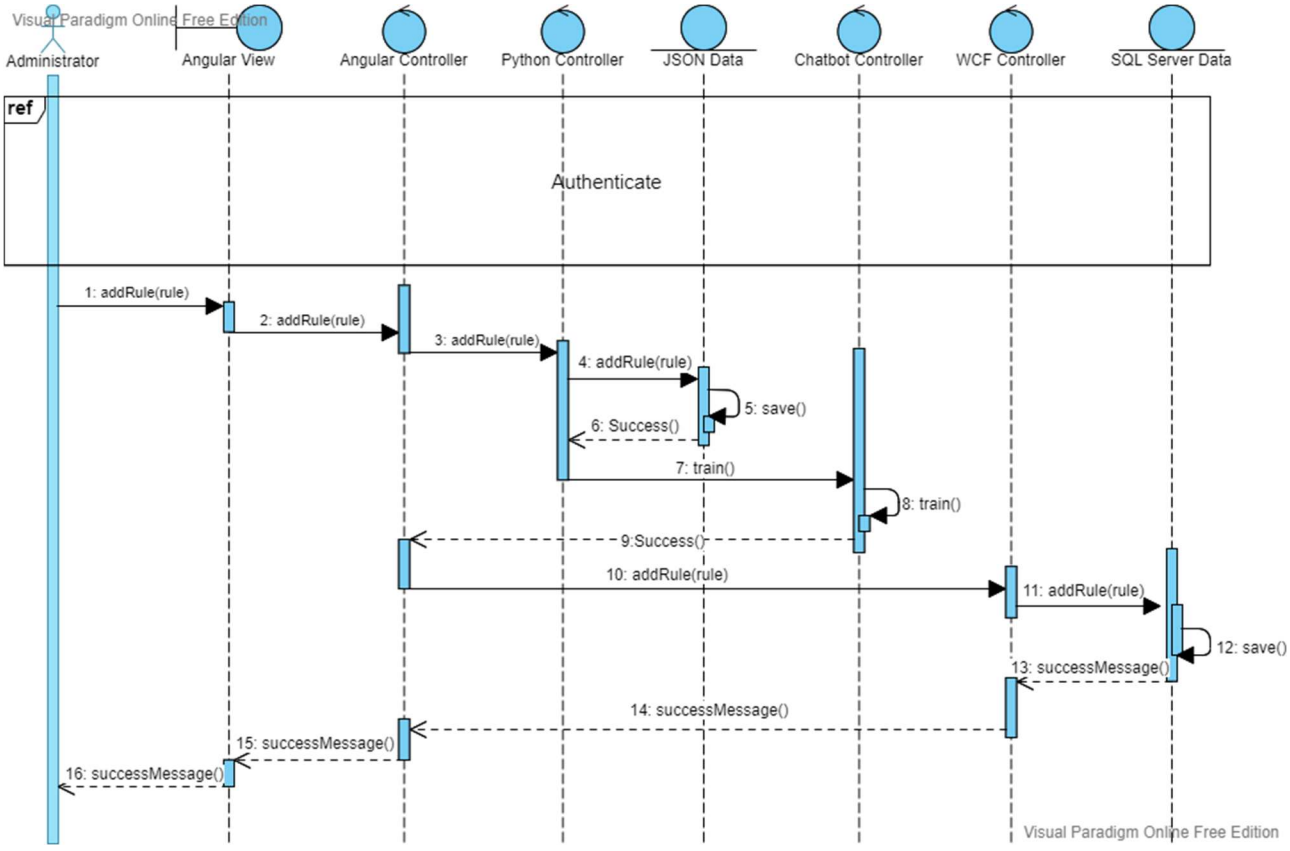

Figure 20: Sequence Diagram "Add rule of use"

Delete user account

This diagram shows the scenario that occurs when deleting a user account.

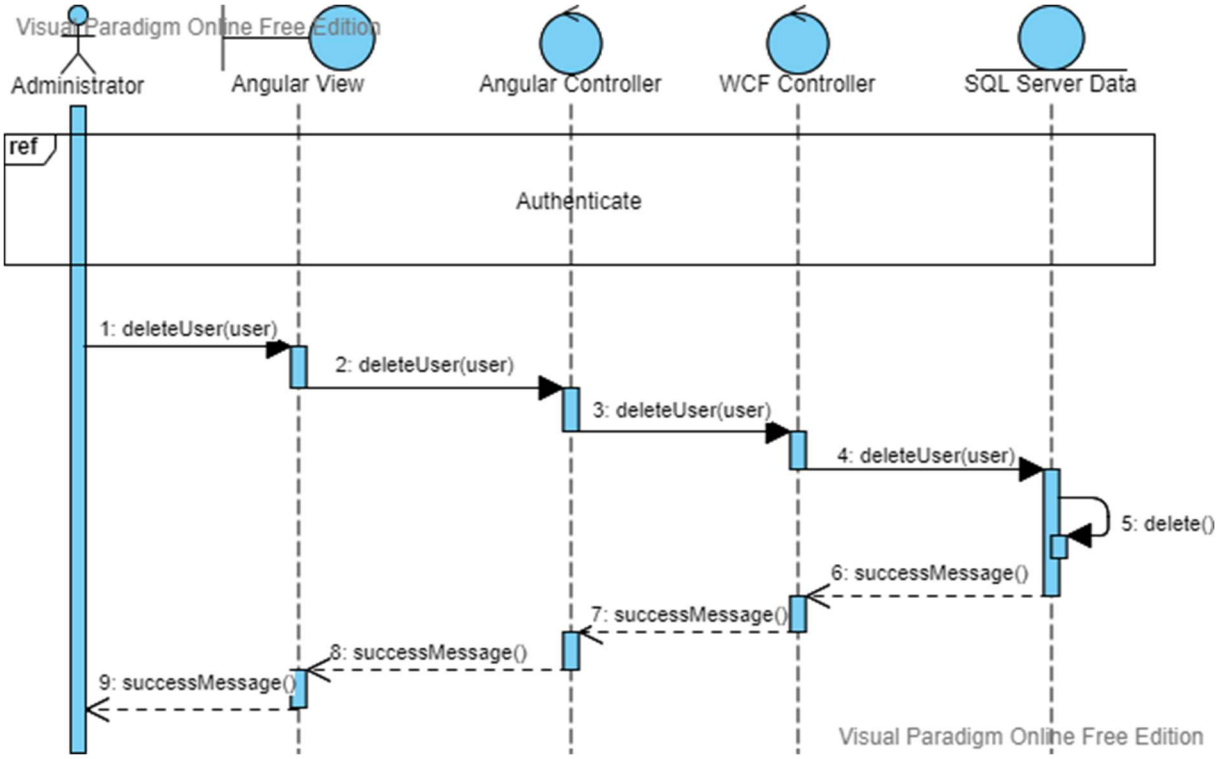

Figure 21: Sequence Diagram "Delete user account"

Consult chatbot history

This diagram shows the scenario that occurs when consulting the chatbot history.

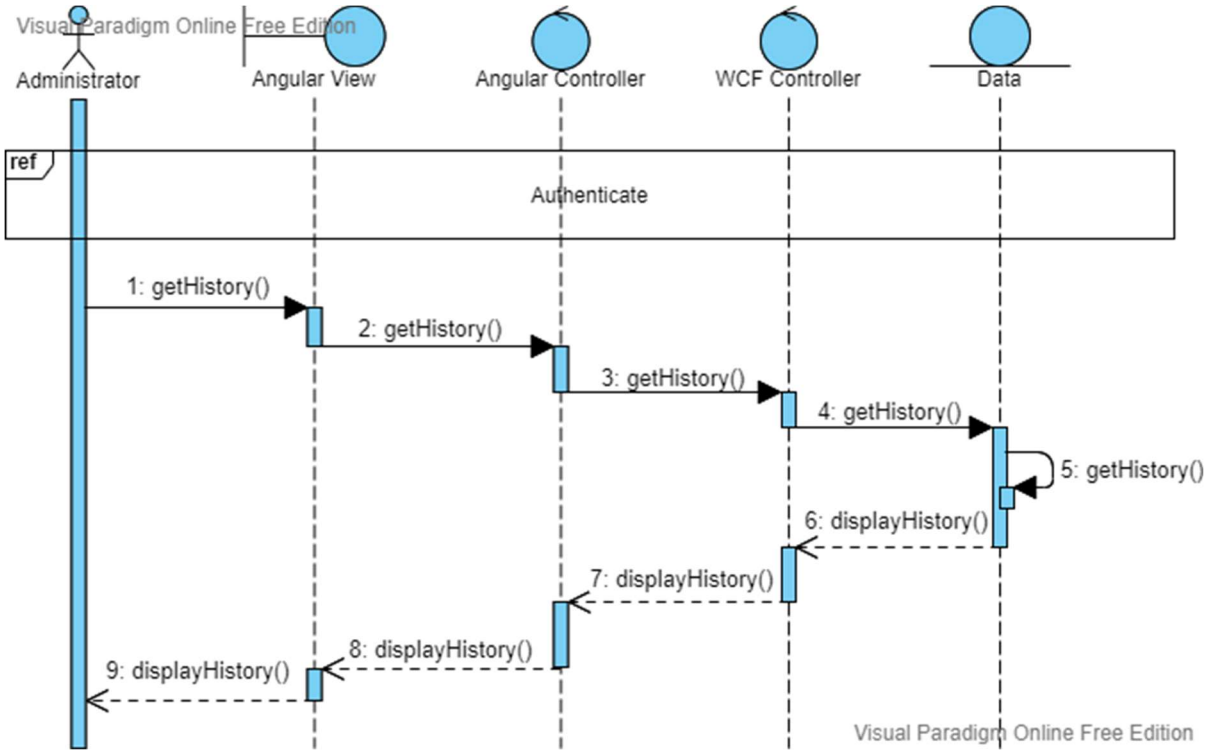

Figure 22: Sequence Diagram "Consult chatbot history"

Interact with the chatbot

This diagram shows the scenario that occurs when interacting with the chatbot.

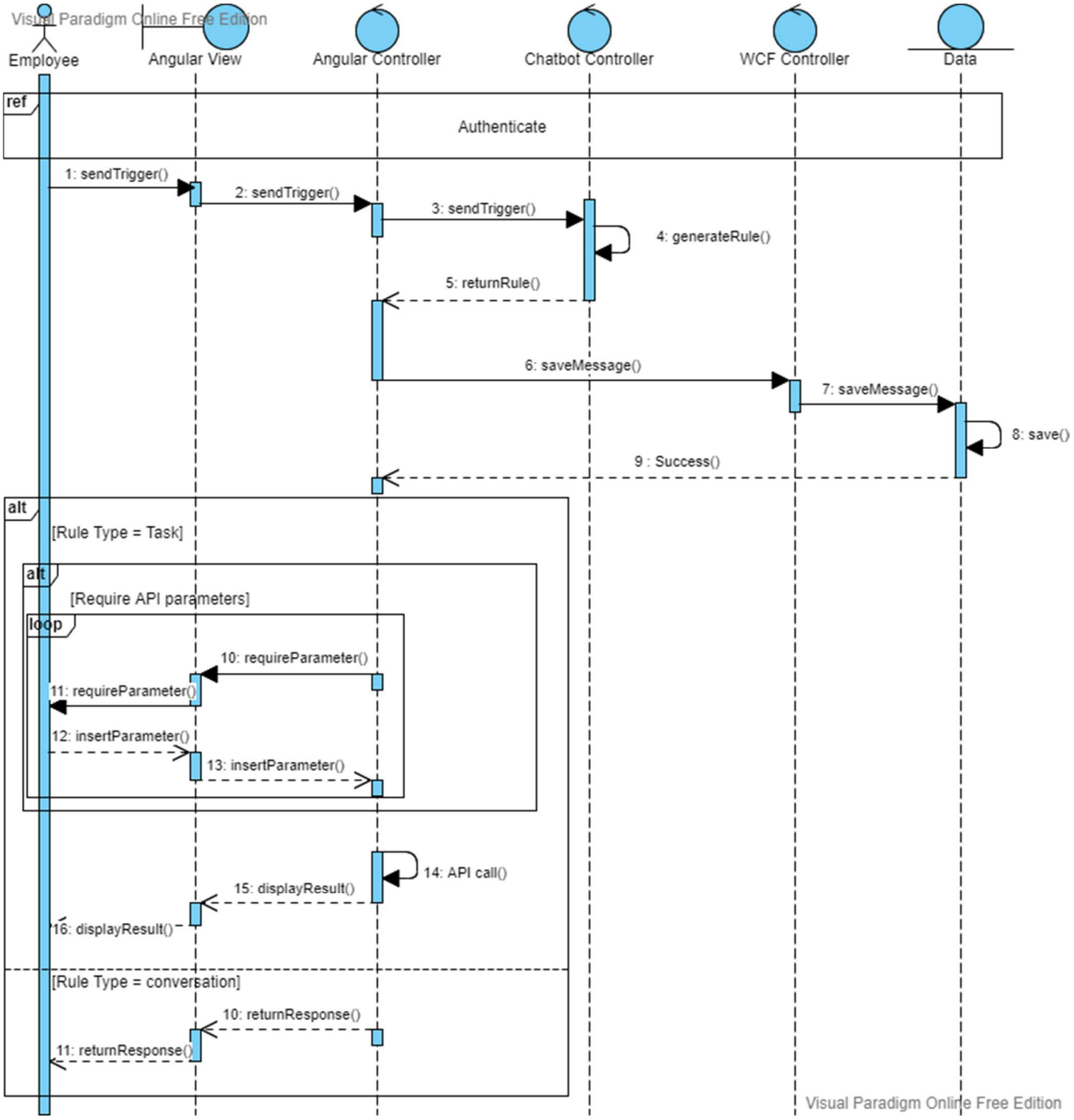

Figure 23: Sequence Diagram "Interact with the chatbot"

III. The static modeling [N10]

A logical (static) model is a static view of the objects and classes that make up the design/analysis space, focusing on the structure level of entities and their relationships within the system.

1. The Class Diagram

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Entities	Description
MessageHistory	Represents the entire history of the interactions between the user and the chatbot.
Function	Represents the employee's function within the organization ex: project Manager.
Permission	Represents what authorization a specific function has. Ex: consult history.
Rule	Represents a set of triggers and responses.
Trigger	Represents a message that initiates a specific rule.
SequenceMessage	Represents a set of questions returned by the chatbot to get the api parameters.
Conversation	Represents a rule that returns a message(s).
Task	Represents a rule that returns an api.
TaskResponseType	Represents the type of response to display to the user (matrix, statistics, value).

Table 7: Entities description

Figure 24: The class diagram

2. The Relational model

In relational model, the data and relationships are represented by collection of inter-related tables. Each table is a group of column and rows, where column represents attribute of an entity and rows represents records.

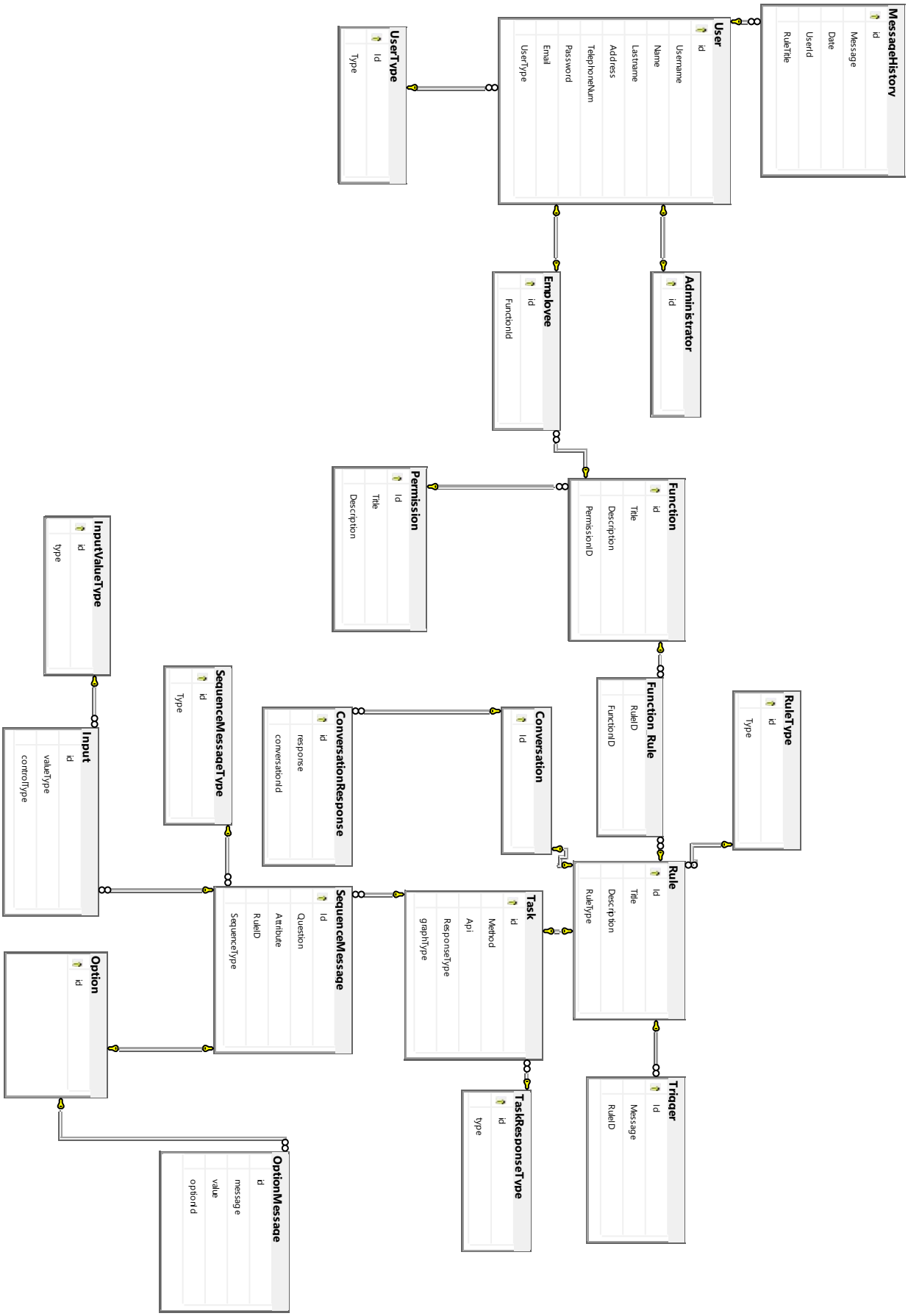

Figure 25: The relational model

3. The Data dictionary

Table name	Column name	Data type	Size	Obligatory	Key	
					Primary	Foreign
User	Id	Int	2^31	Yes	Yes	No
User	Username	Varchar	50	Yes	No	No
User	Name	Varchar	50	No	No	No
User	Lastname	Varchar	50	No	No	no
User	Address	Varchar	50	No	No	no
User	TelephoneNum	Int	2^31	No	No	no
User	Email	Varchar	50	No	No	no
User	Password	Varchar	50	Yes	No	no
User	UserType	Varchar	50	Yes	No	Yes
Administrator	Id	Int	2^31	Yes	Yes	Yes
Employee	Id	Int	2^31	Yes	Yes	Yes
Employee	FunctionId	Int	2^31	Yes	No	Yes
UserType	Id	Int	2^31	Yes	Yes	No
UserType	Type	Varchar	50	Yes	No	No
MessageHistory	Id	Int	2^31	Yes	Yes	No
MessageHistory	Username	Varchar	50	Yes	No	Yes
MessageHistory	Message	Varchar	MAX	Yes	No	No
MessageHistory	Date	Varchar	50	No	No	No
Function	Id	Int	2^31	Yes	Yes	No
Function	Title	Varchar	MAX	Yes	No	No
Function	Description	Varchar	MAX	No	No	No
Function	PermissionId	Int	2^31	Yes	No	Yes
Permission	Id	Int	2^31	Yes	Yes	No
Permission	Title	Varchar	MAX	Yes	No	No
Permission	Description	Varchar	MAX	No	No	No
Rule	Id	Int	2^31	Yes	Yes	No
Rule	Title	Varchar	50	Yes	No	No
Rule	RuleTypeId	Int	2^31	Yes	No	Yes
FunctionRule	RuleId	Int	2^31	Yes	Yes	Yes
FunctionRule	FunctionID	Int	2^31	Yes	Yes	Yes
RuleType	Id	Int	2^31	Yes	Yes	No
RuleType	Type	Varchar	50	Yes	No	No
Trigger	Id	Int	2^31	Yes	Yes	No
Trigger	Message	Varchar	MAX	Yes	No	No
Trigger	RuleId	Int	2^31	Yes	No	Yes
SequenceMessage	Id	Int	2^31	Yes	Yes	No
SequenceMessage	Question	Varchar	MAX	Yes	No	No
SequenceMessage	Attribute	Varchar	MAX	Yes	No	No
SequenceMessage	RuleId	Int	2^31	Yes	No	Yes
SequenceMessage	SequenceType	Varchar	50	Yes	No	Yes
Input	Id	Int	2^31	Yes	Yes	Yes
Input	ValueType	Varchar	50	Yes	No	Yes
Input	ControlType	Varchar	50	Yes	No	No

Option	Id	Int	2^31	Yes	Yes	Yes
OptionMessage	Id	Int	2^31	Yes	Yes	No
OptionMessage	Message	Varchar	50	Yes	No	No
OptionMessage	Value	Varchar	50	Yes	No	No
OptionMessage	OptionId	Int	2^31	Yes	No	Yes
Task	Id	Int	2^31	Yes	Yes	Yes
Task	Method	Varchar	10	Yes	No	No
Task	Api	Varchar	MAX	Yes	No	No
Task	ResponseTypeId	Int	2^31	Yes	No	Yes
TaskResponseType	Id	Int	2^31	Yes	Yes	No
TaskResponseType	Type	Varchar	50	Yes	No	No
Conversation	Id	Int	2^31	Yes	Yes	Yes
ConversationResponse	Id	Int	2^31	Yes	Yes	No
ConversationResponse	Response	Varchar	50	Yes	No	No
ConversationResponse	ConversationId	Int	2^31	Yes	No	Yes

Table 8: Data dictionary

4. The hardware architecture

The hardware architecture provides a detailed view of the way components will be deployed across the system infrastructure. It details network capabilities, server specifications, hardware requirements and other information related to deploying the proposed system.

5. The deployment diagram:

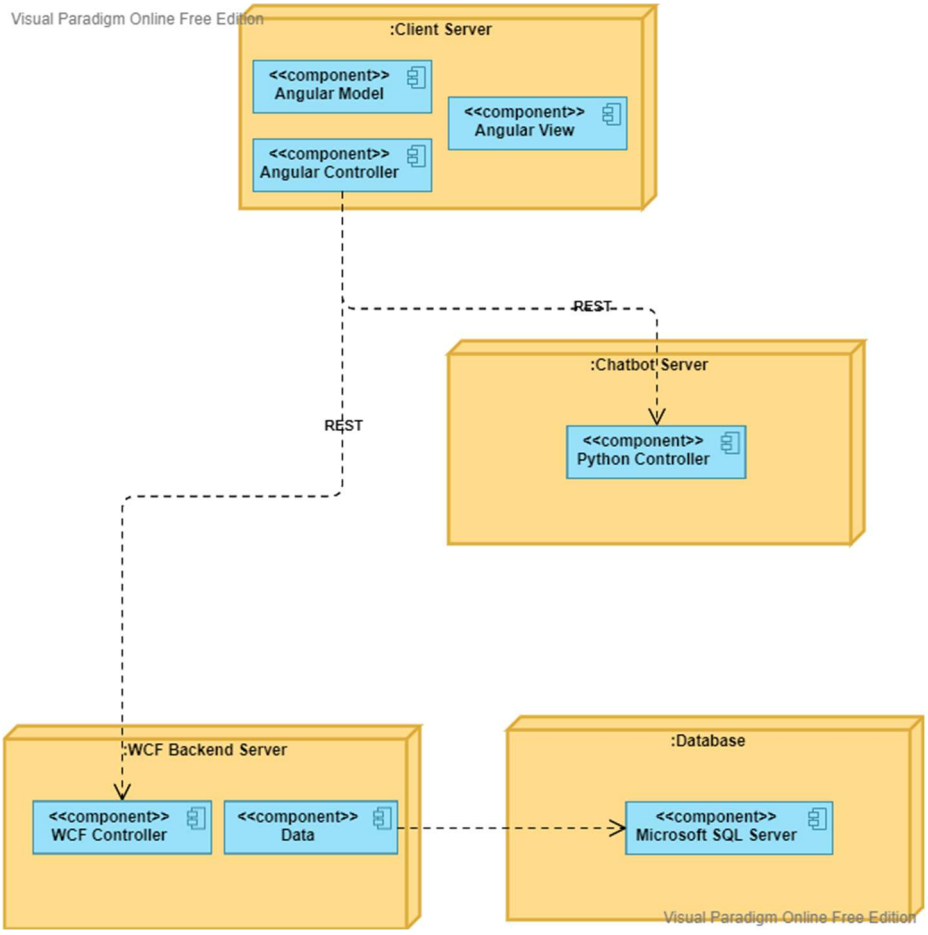

Figure 26: The deployment diagram

Conclusion

During this chapter we presented our conceptual solution for the system using the dynamic and the static modeling. In the next chapter we will describe the development process of the project.

Chapter 4
Realization

Introduction

After finishing the conceptual part of the project, this final chapter consists of defining the various hardware and software tools that we used during this project, then we will showcase the main graphical interfaces and finally we will finish the chapter with the testing phase.

I. Development environment

The development environment is the set of processes and programming tools used to create the program. Here is a description of our project's hardware and software environments.

1. The hardware environment

- **Laptop Asus x556u**
  - Processor: Intel(R) Core (TM) i5-6198DU CPU @2.30GHz 2.40GHz
  - Live memory (RAM): 8Go.
  - Primary hard drive: 1To
  - Graphic card: NVIDIA GeForce 920MX.
  - Monitor: 16.6 HD Display
- **Laptop HP**
  - Processor: Intel(R) Core (TM) i5-7500DU CPU @3.40GHz
  - Live memory (RAM): 8Go.
  - Primary hard drive: 1To
  - Graphic card: NVIDIA GeForce 920M.
  - Monitor: 16.6 HD Display

2. The software environment

- **Windows 10**

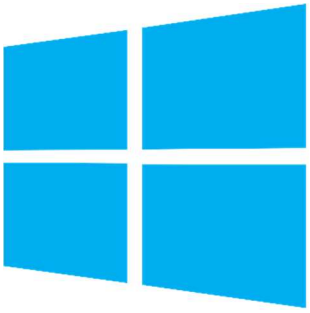

Figure 27: Windows 10 Logo

Windows 10 is a major version of the Microsoft Windows operating system that was released on July 29, 2015. It is built on the Windows NT kernel and follows Windows 8 [N11].

- **Angular 11**

Figure 28: Angular 11 Logo

Angular is a platform and framework for building single-page client applications using HTML and TypeScript led by the Angular Team at Google [N12].

Choice justification: This framework allows us to create a single page application. Its architecture is based on modules and components and facilitates application maintenance.

- **Python**

Figure 29: Python Logo

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects [N13].

- **TensorFlow**

Figure 30: TensorFlow Logo

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks [N14].

- **WCF (Windows Communication Foundation)**

Figure 31: WCF Logo

Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another [N15].

- **SQL Server Management Studio 18**

Figure 32: SQL Server Management 18 Logo

SQL Server Management Studio (SSMS) is a software application first launched with Microsoft SQL Server 2005 that is used for configuring, managing, and administering all components within Microsoft SQL Server [N16]

- **Anaconda**

Figure 33: Anaconda Logo

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment [N17].

- **Microsoft visual studio**

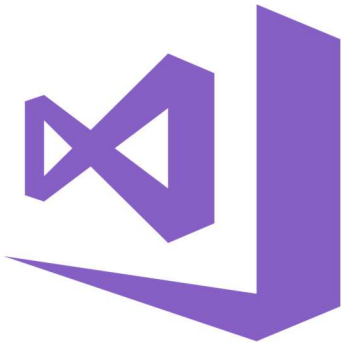

Figure 34: Microsoft Visual Studio Logo

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps [N18].

- **Visual studio code**

Figure 35: Visual Code Logo

Visual Studio Code is a freeware source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git [N19].

- **Visual paradigm online**

Figure 36: Visual Paradigm Online Logo

Visual Paradigm is a UML CASE Tool, it provides report generation and code engineering capabilities including code generation. It can reverse engineer diagrams from code, and provide round-trip engineering for various programming languages [N20].

- **Apache cordova**

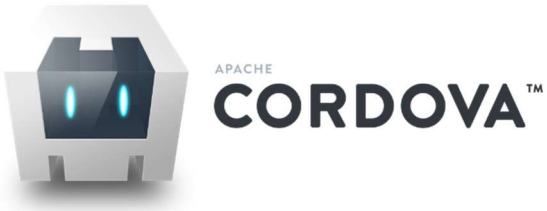

Figure 37: Apache Cordova Logo

Apache Cordova is an open-source framework that enables web developers to use their HTML, CSS, and JavaScript content to create a native application for a variety of mobile platforms [N22].

II. Main graphical interfaces

- Login

Allows the user to connect to the application using his username and password

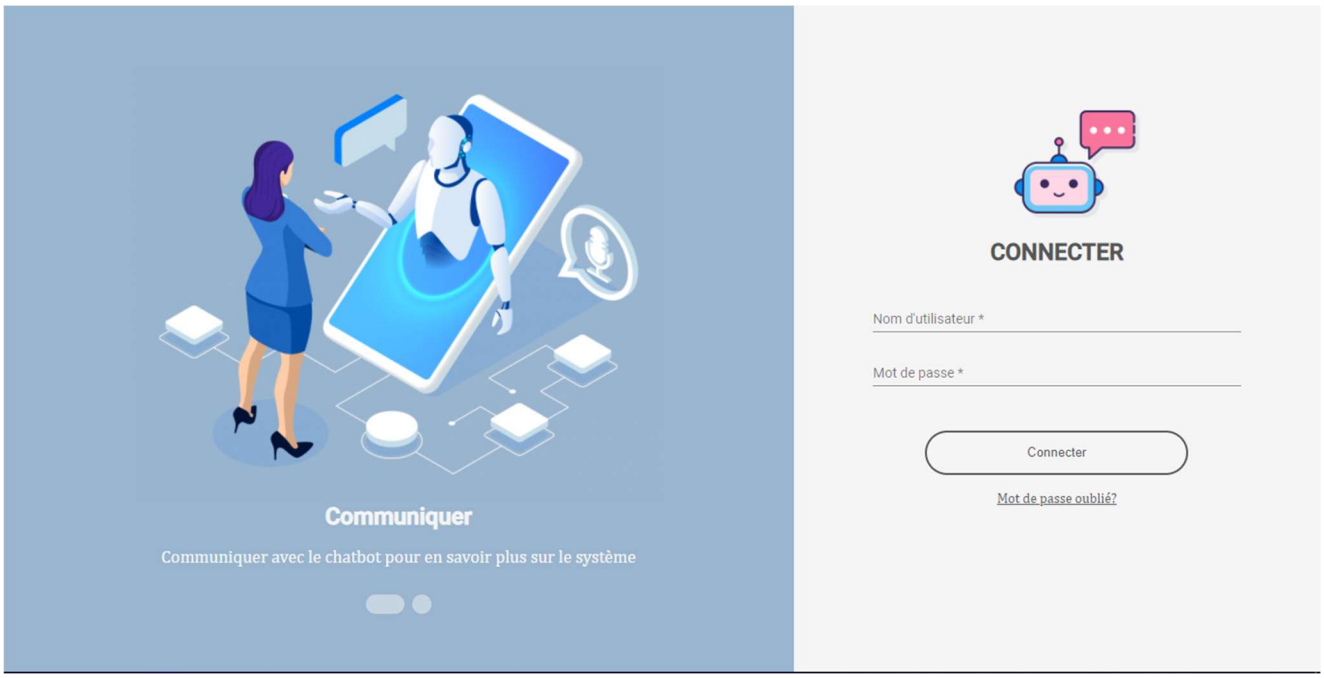

Figure 38: Login interface desktop version

Figure 39: Login interface mobile version

- Chatbot

Allows the user to interact with the chatbot in text mode and in French and also consult all his pinned rules.

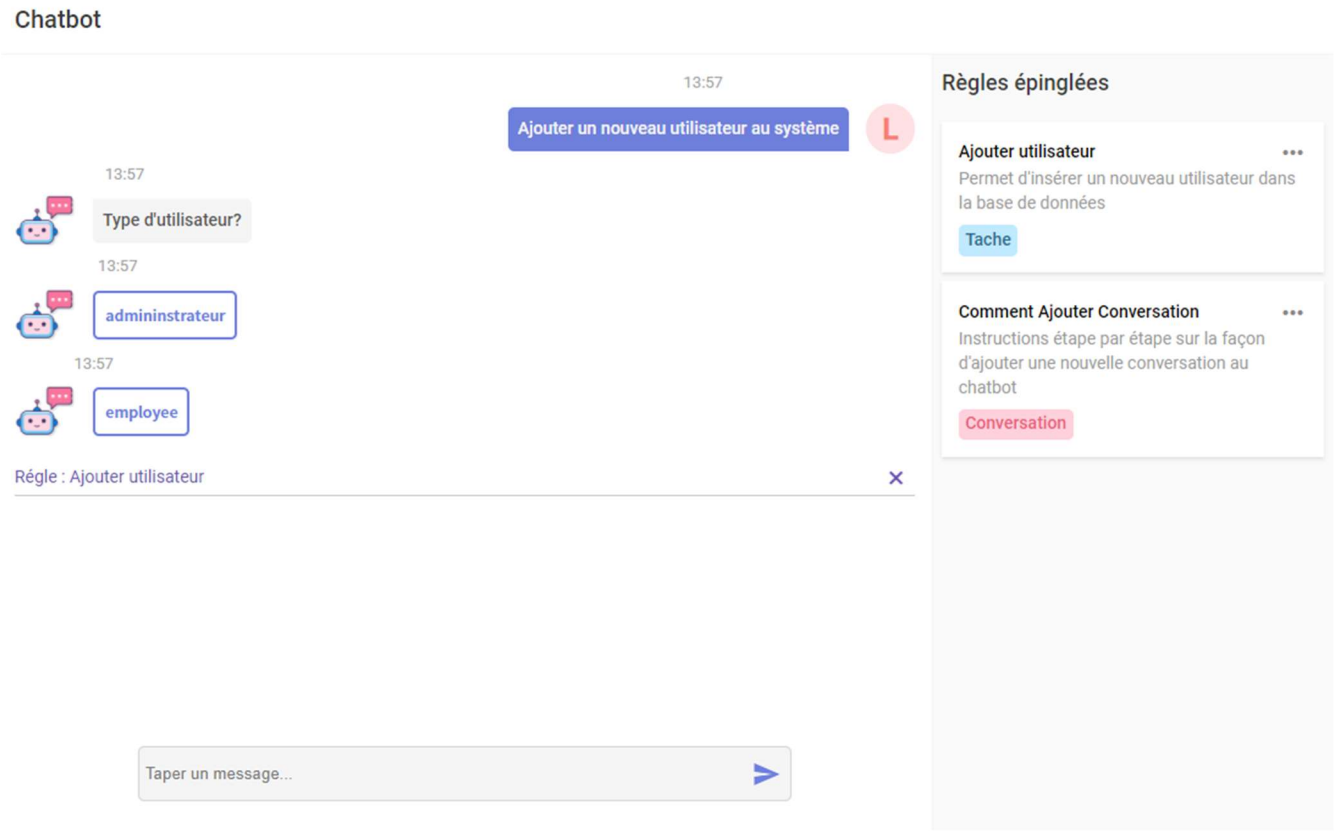

Figure 40: Chatbot interface desktop version

Figure 41: Chatbot interface mobile version

- Rules of use

Allows the user to consult all rules of use.

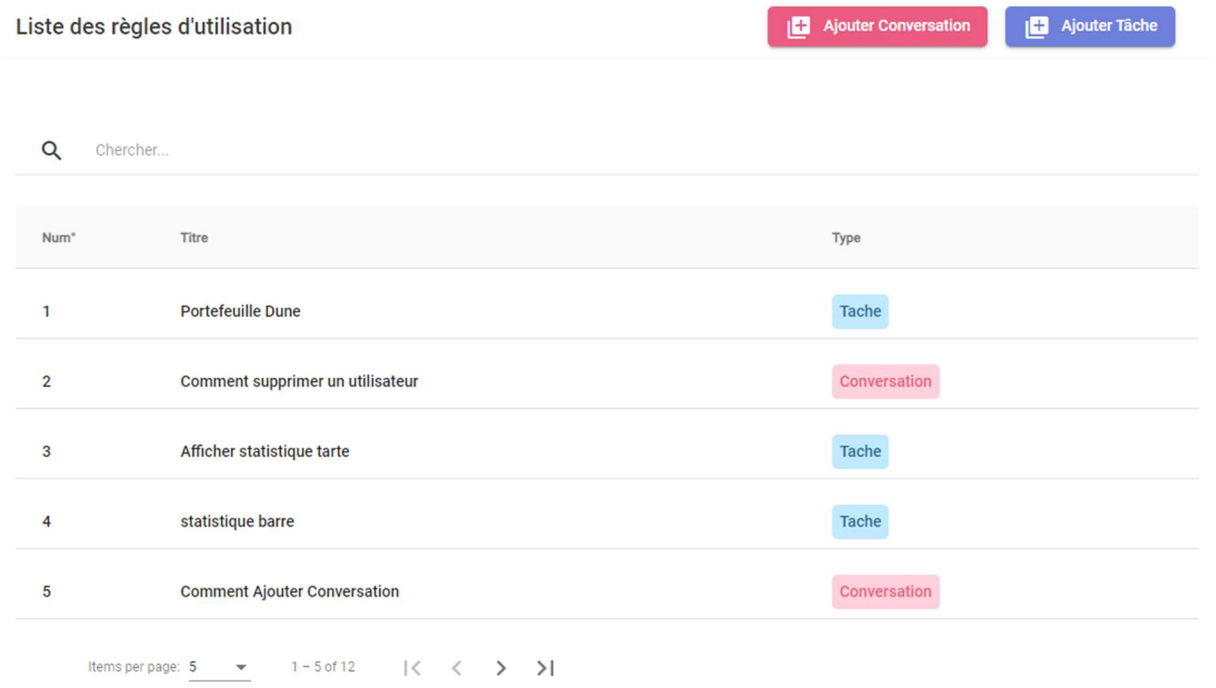

Figure 42: Rules Interface desktop version

Figure 43: Rules Interface mobile version

- Add task

Allows the user to add a task to the chatbot.

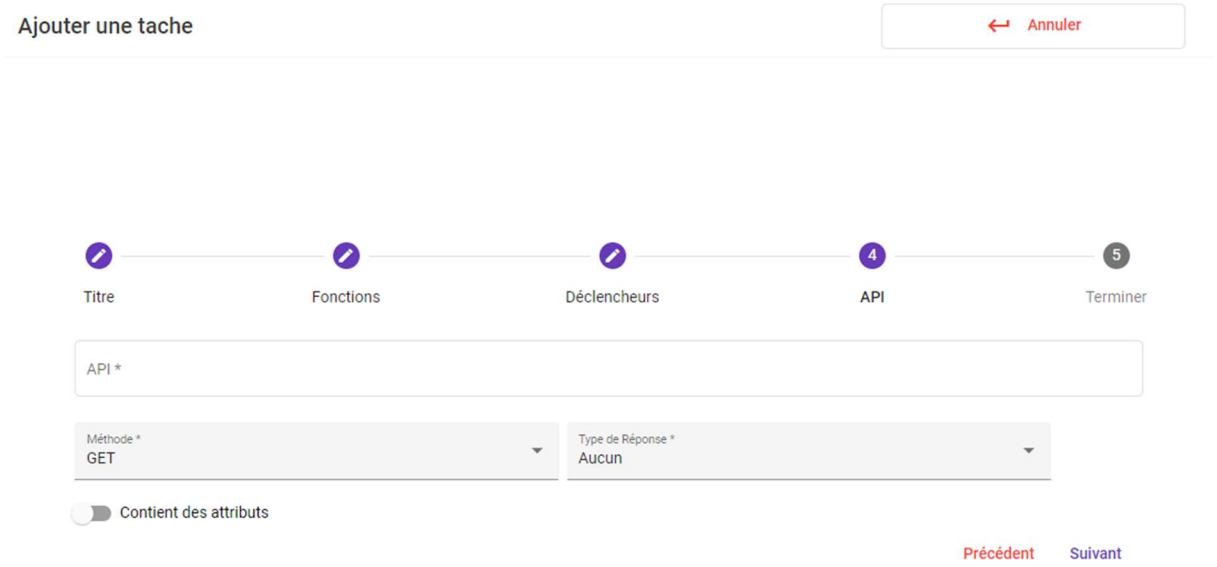

Figure 44: Add task interface desktop version

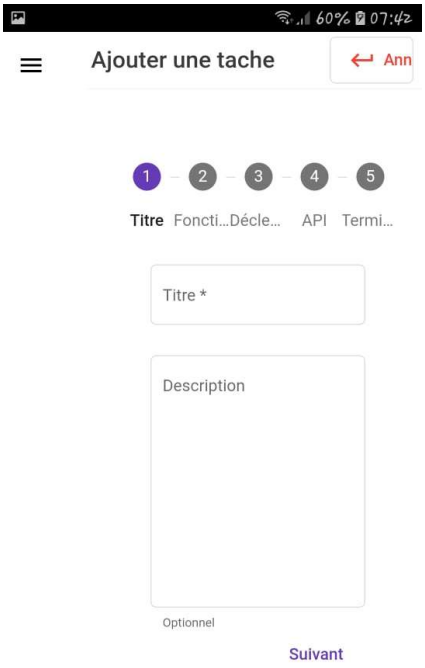

Figure 45: Add task mobile version

- Update conversation

Allows the user to update a conversation

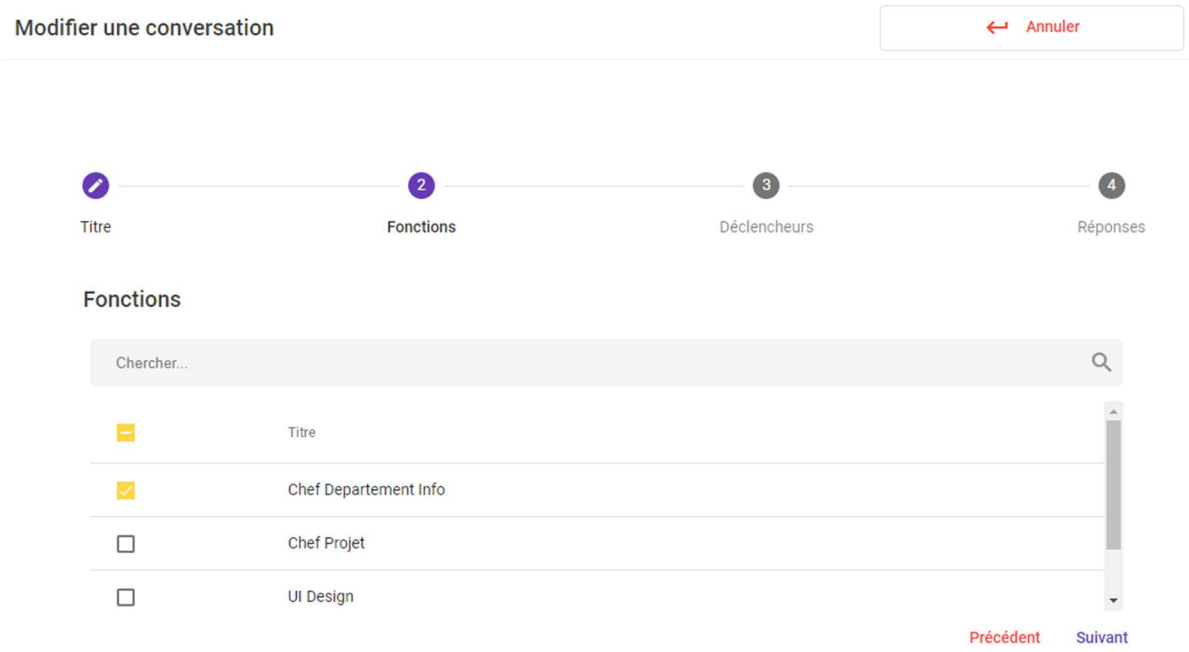

Figure 46: Update conversation interface desktop version

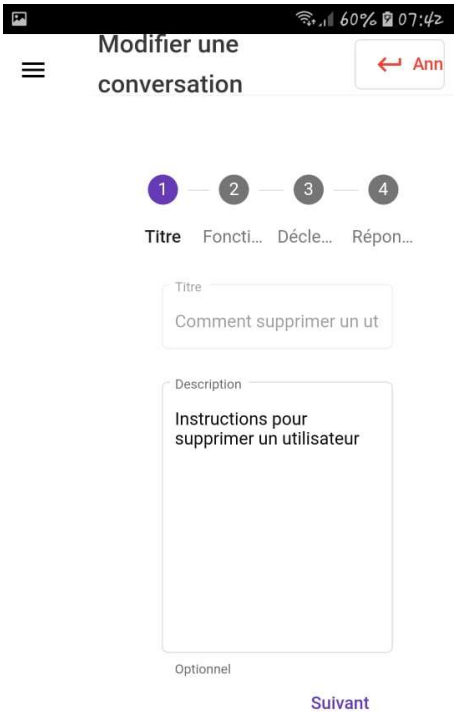

Figure 47: Update conversation interface mobile version

- Consult rule of use

Allows the user to consult a rule of use

Figure 48: Consult rule interface desktop version

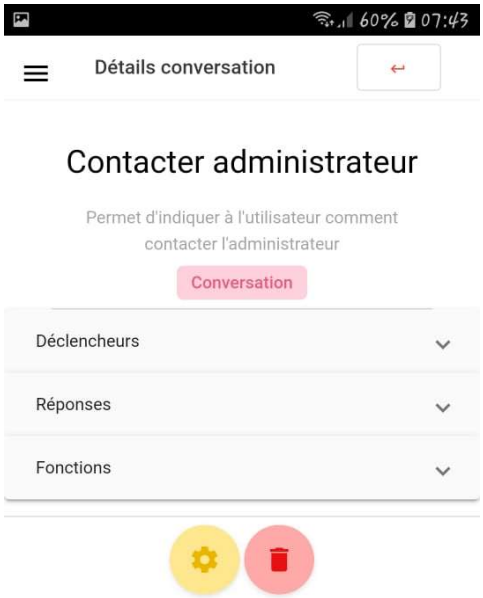

Figure 49: Consult rule interface mobile version

- users

Allows the administrator to consult all users.

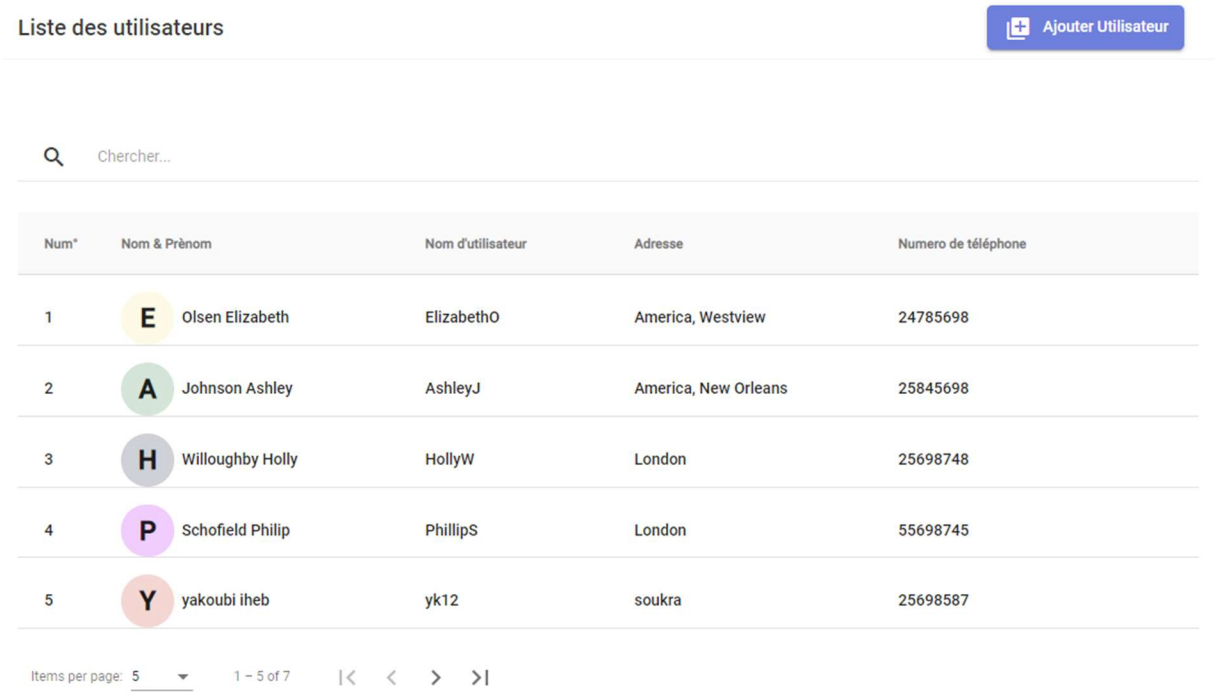

Figure 50: users interface desktop version

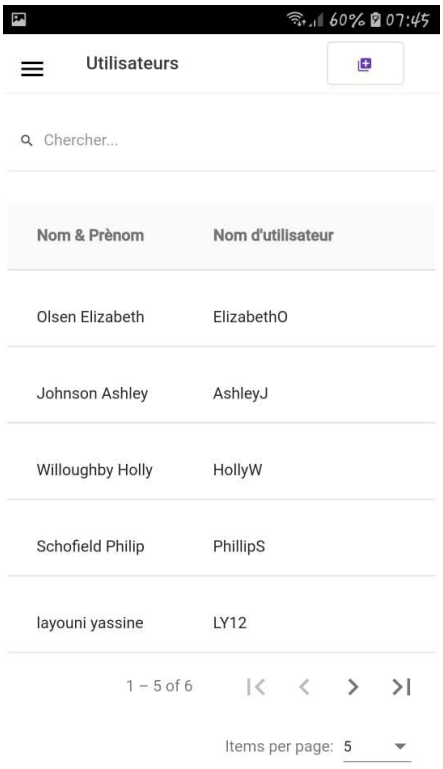

Figure 51: users interface mobile version

- Add user

Allows the administrator to add a new user.

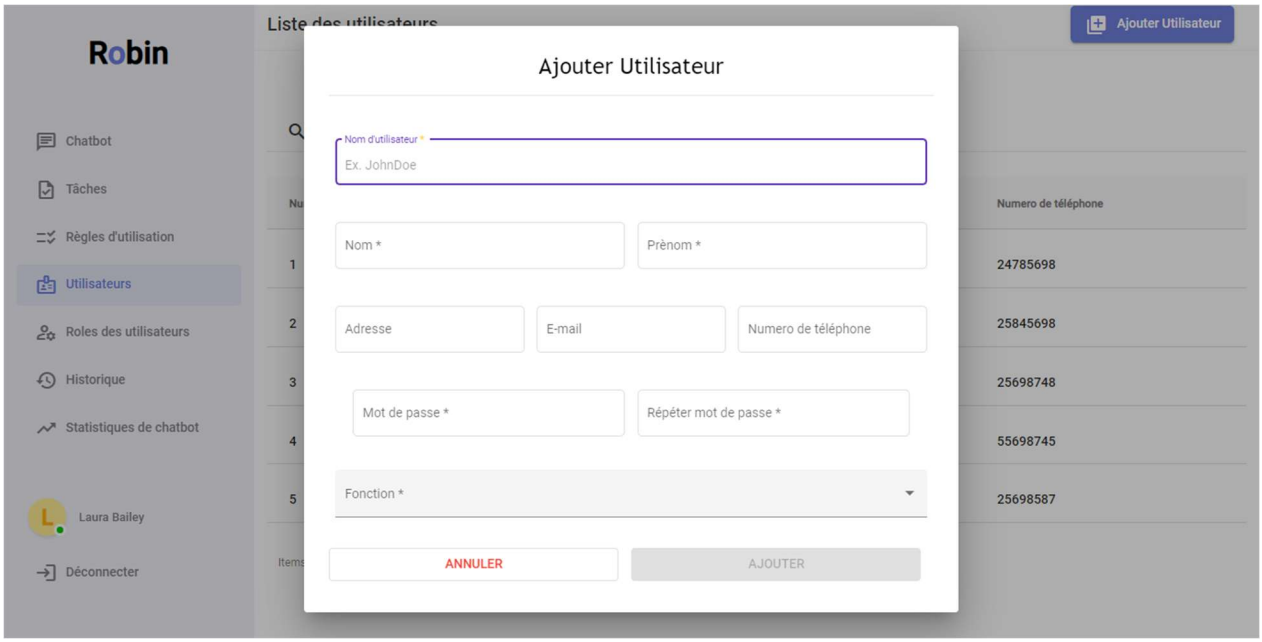

Figure 52: Add user interface desktop version

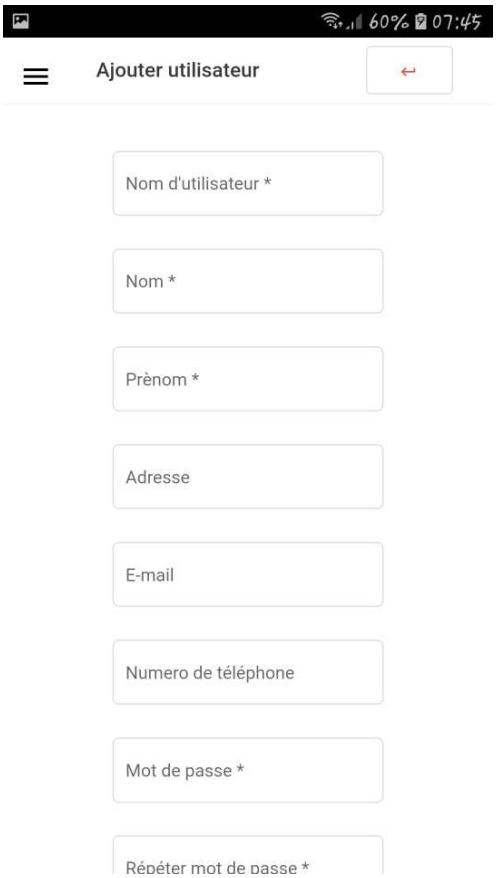

Figure 53: Add user interface mobile version

- Consult user

Allows the administrator to consult a user.

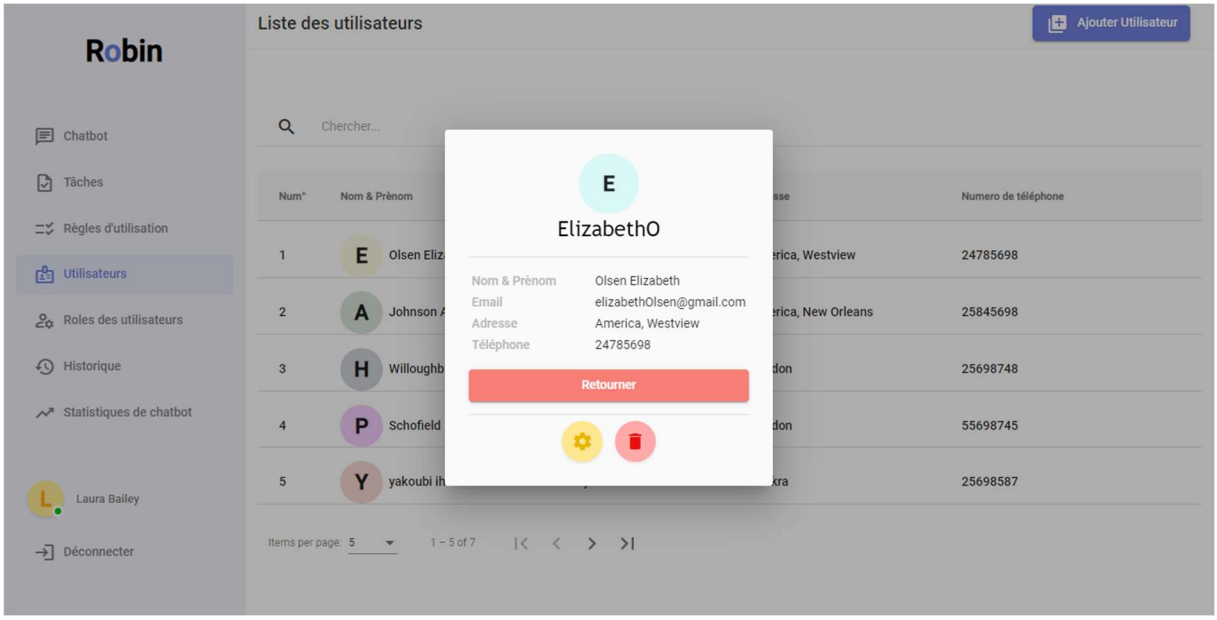

Figure 54: consult user interface desktop version

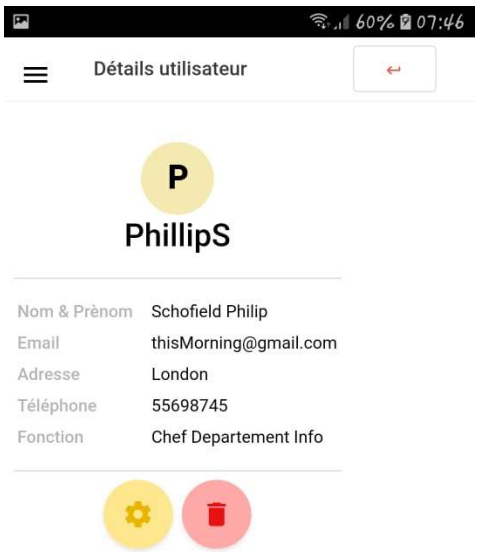

Figure 55: Consult user interface mobile version

- Functions

Allows the administrator to consult all user functions.

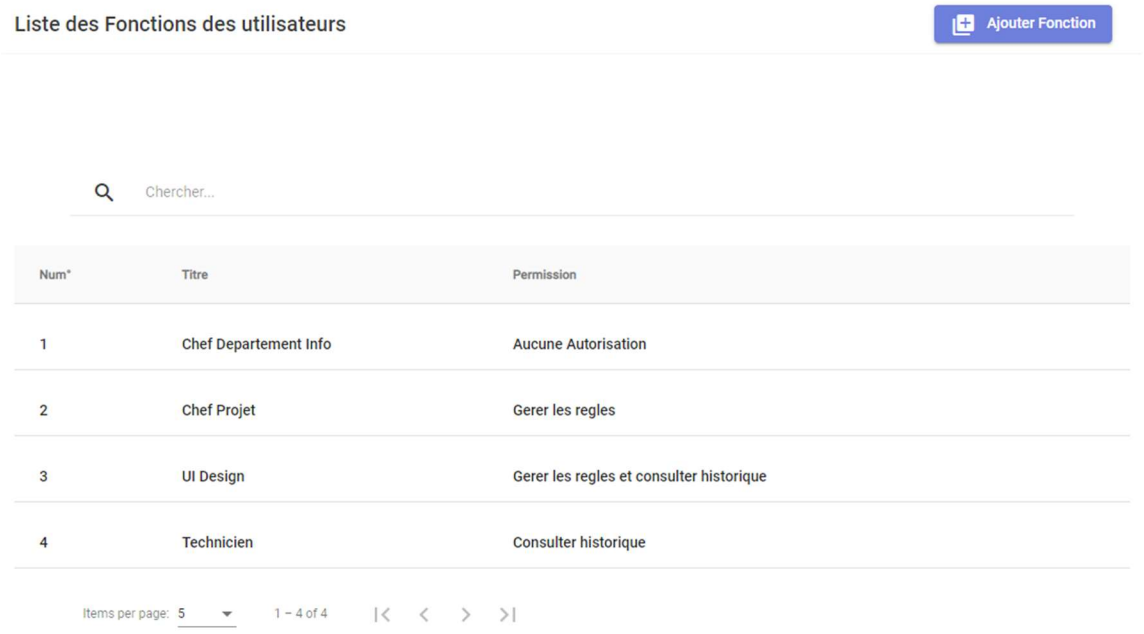

Figure 56: functions interface desktop version

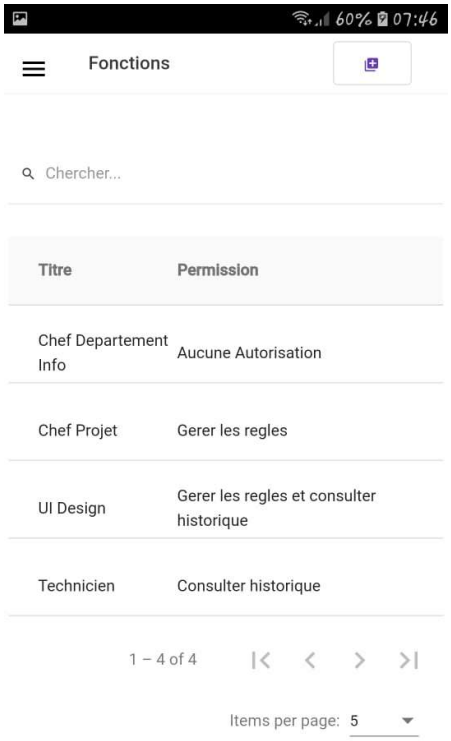

Figure 57: Functions interface mobile version

- Update function

Allows the administrator to update a function.

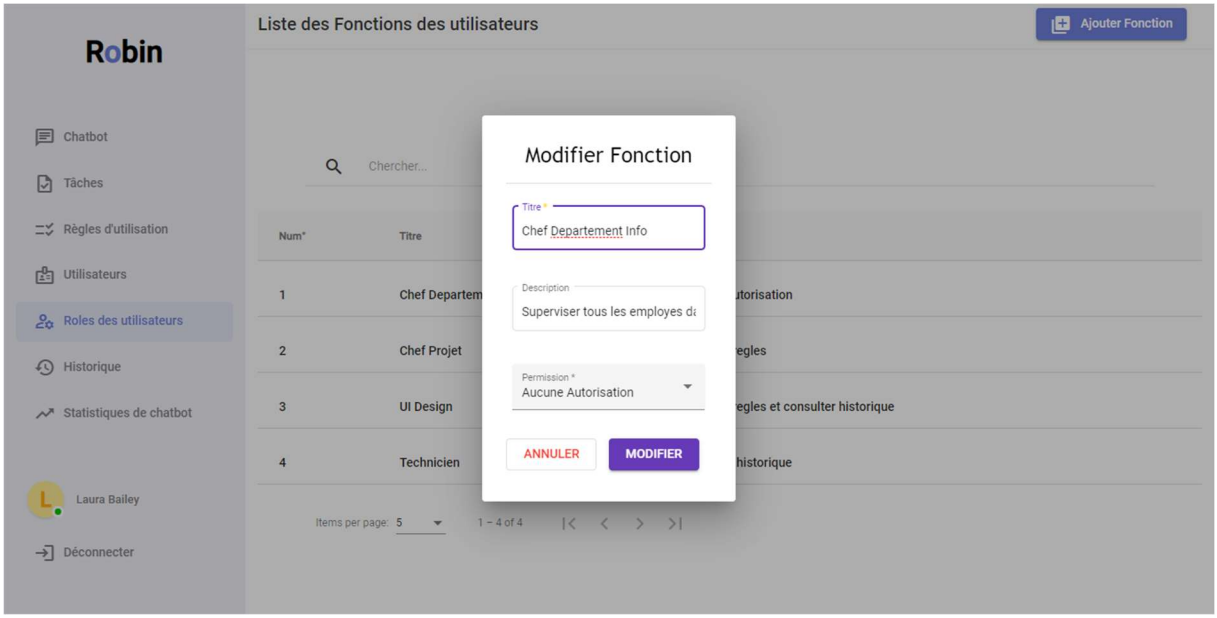

Figure 58: Update function interface desktop version

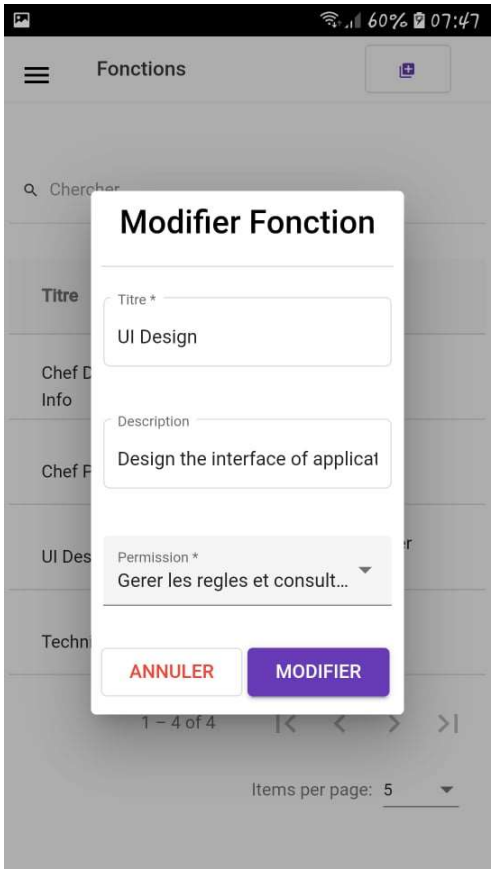

Figure 59: Update function mobile version

- Consult all users' history

Allows the user to consult all users' message history.

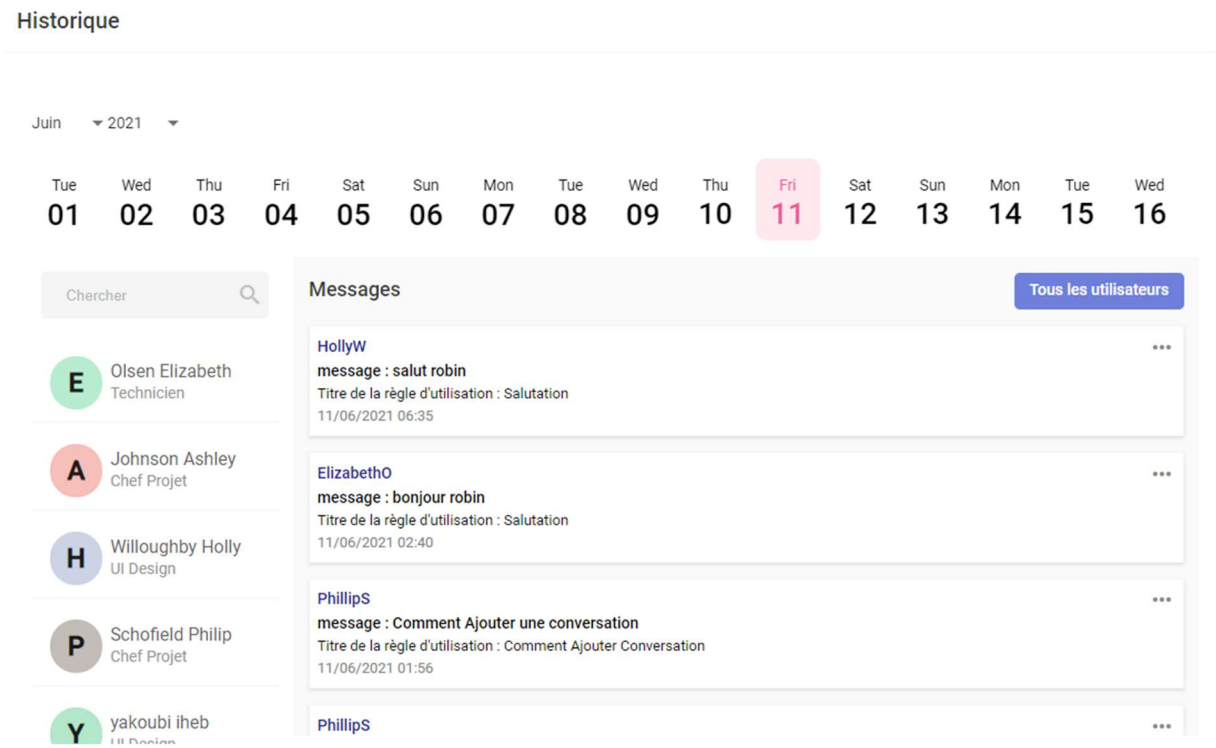

Figure 60: All users' history interface desktop version

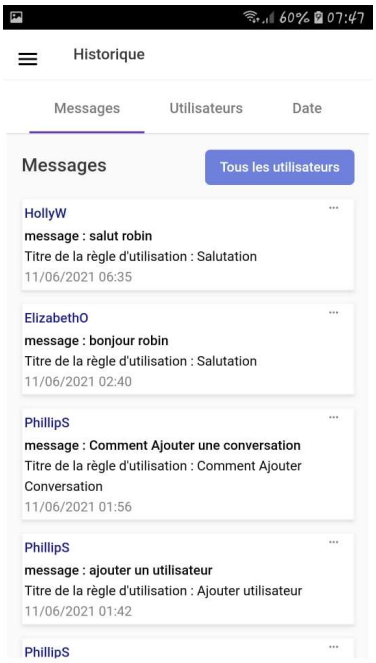

Figure 61: All user's history mobile version

- Consult matrix

Allows the user to consult a matrix returned by the chatbot.

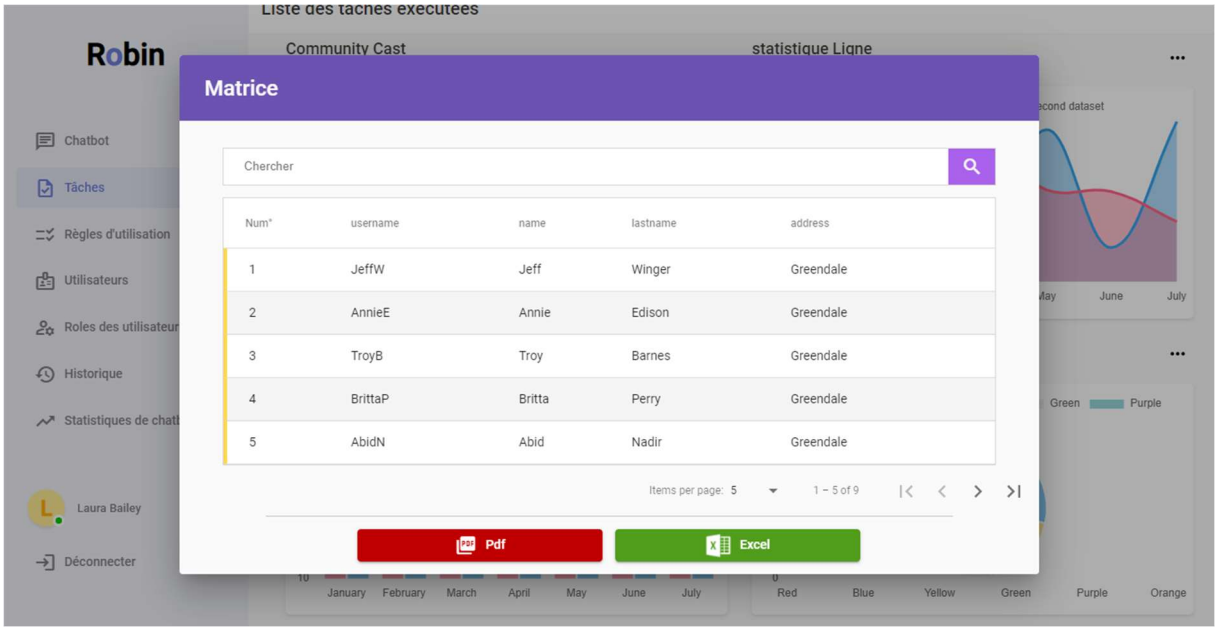

Figure 62: Consult matrix interface desktop version

- Consult chatbot statistics

Allows the user to consult the statistics of the chatbot.

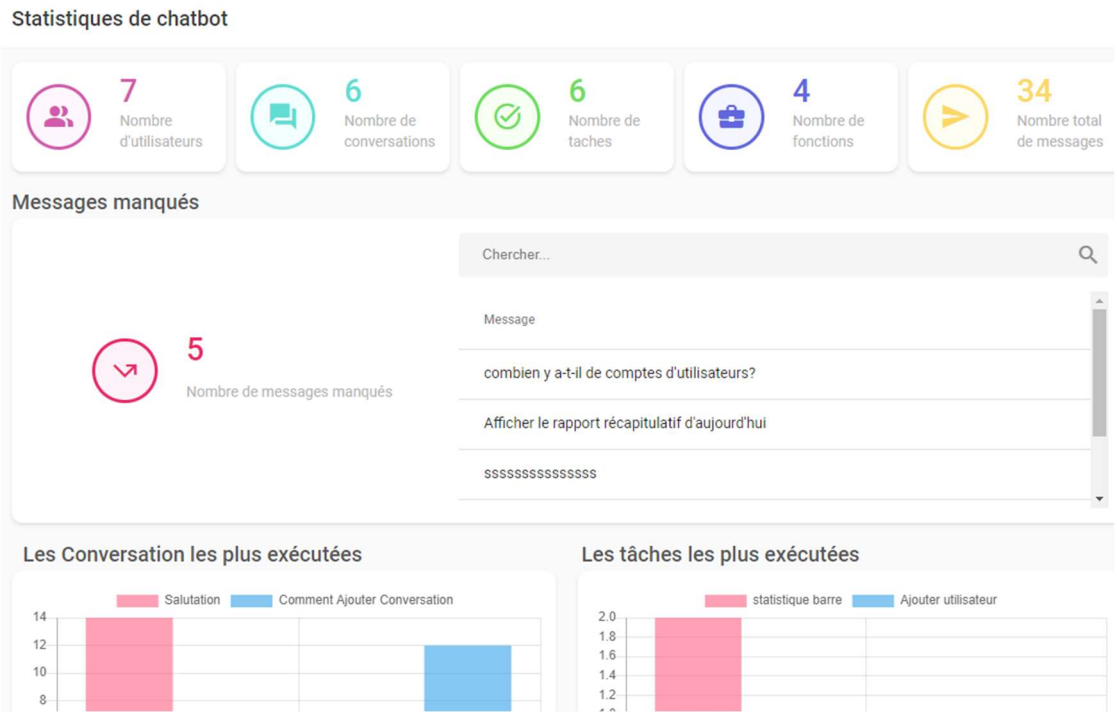

Figure 63: Consult chatbot statistics interface desktop version

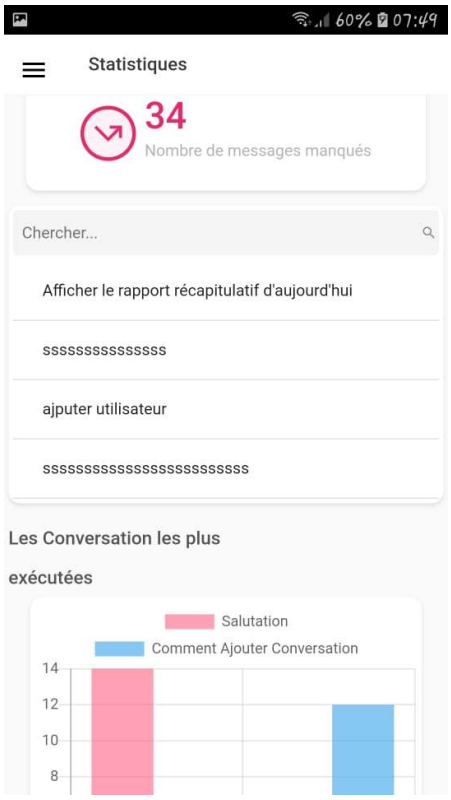

Figure 64: Consult chatbot statistics interface mobile version

III. Testing [N25]

Testing is crucial for creating a reliable program, it ensures that the application doesn't run into any common problems that can result in system failure.

for this reason, we prepared the unit tests and the integration tests of our application:

1. The unit tests

A unit test verifies a small portion of your code independently from other modules of your application, it doesn't test a module as a whole, it tests the smallest units that make up a more significant module.

A unit test could make sure that a method:

- Returns an expected value.
- Throws an exception under the tested condition.
- Changes the state of the system
- Calls another function.

We prepared the unit test manually using POSTMAN.

2. The integration tests

Once we are finished with the unit test, and the small modules of our service are working properly, it is time to test the service as a whole, making sure that the connection between the database and the REST API is functioning perfectly.

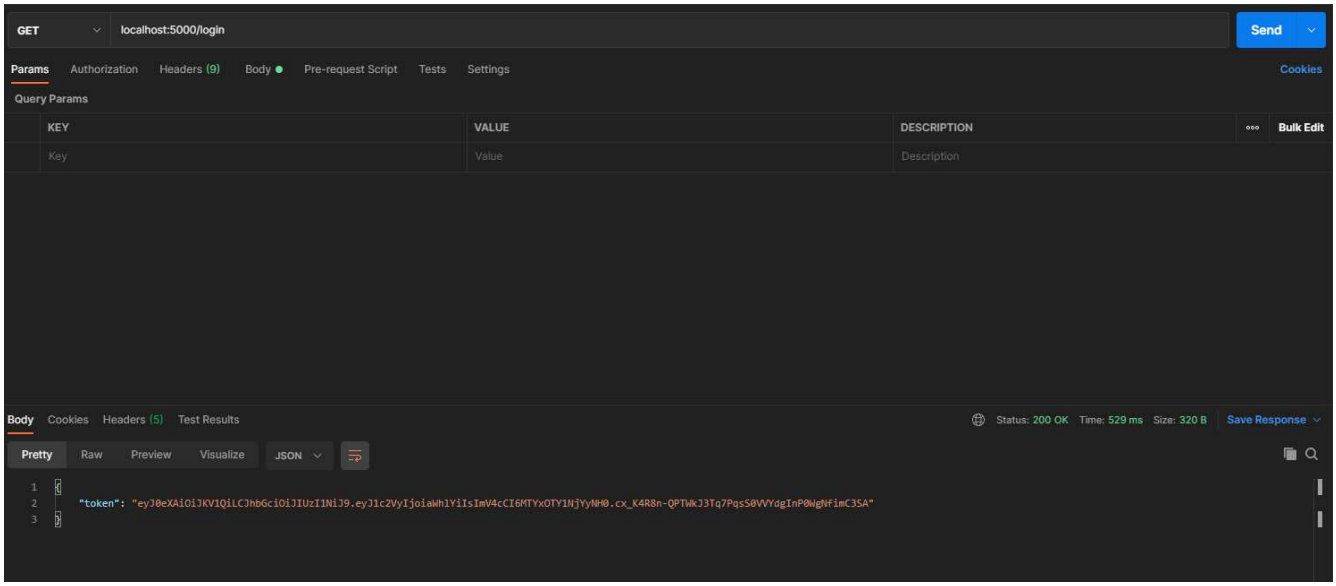

Figure 65: Integration test of the login service

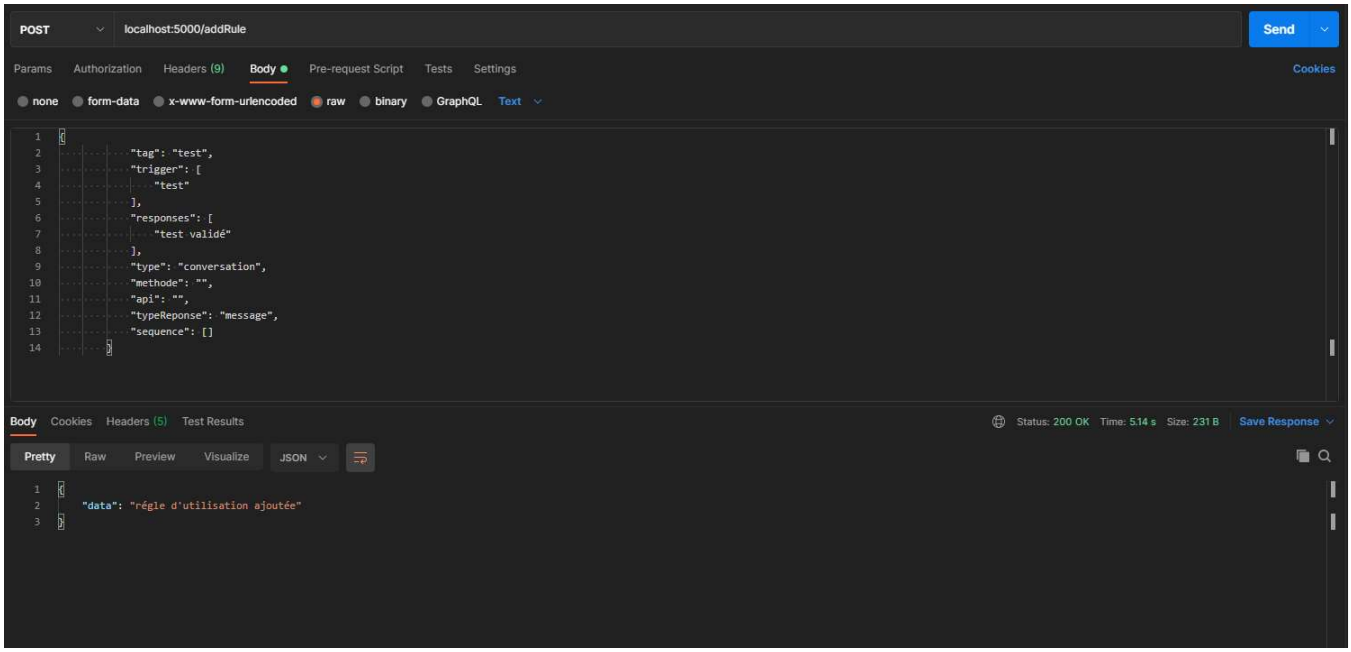

Figure 66: Integration test of add rule service

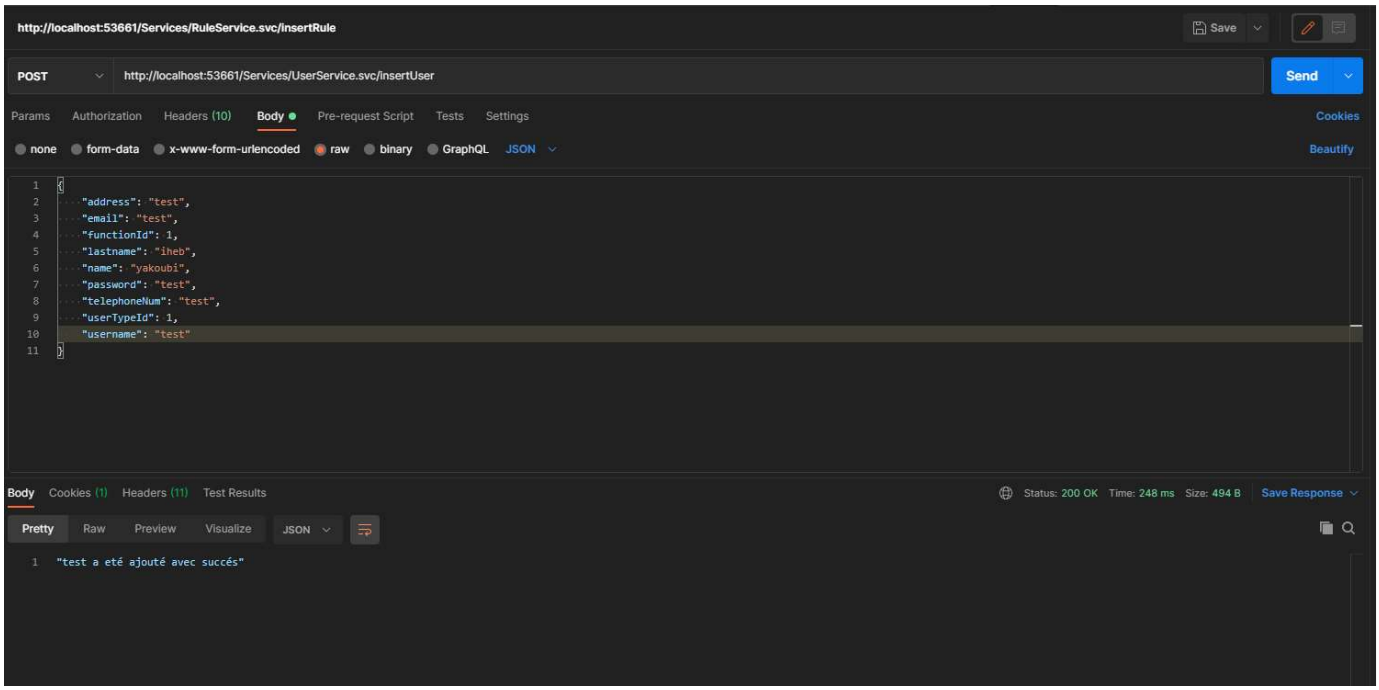

Figure 67: Integration test of add user

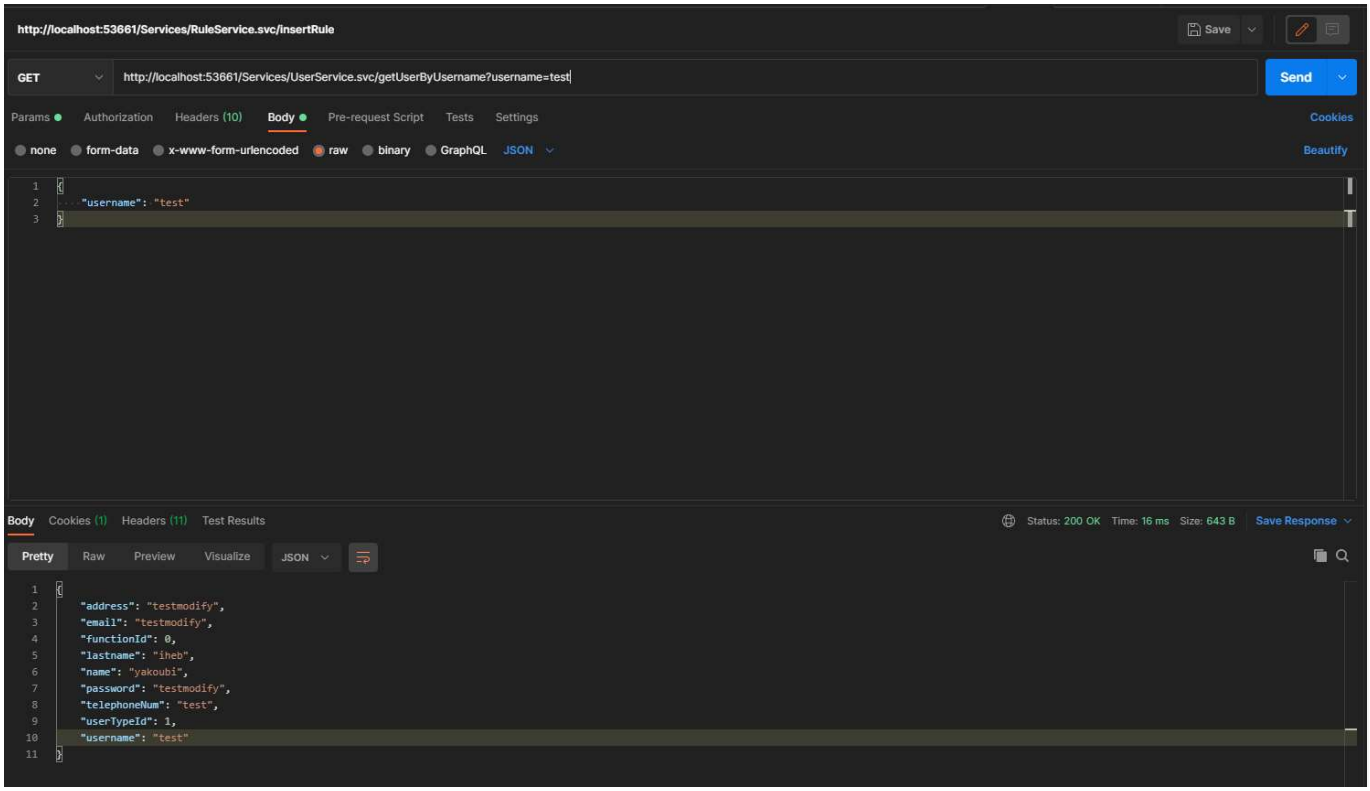

Figure 68: Integration test of get user

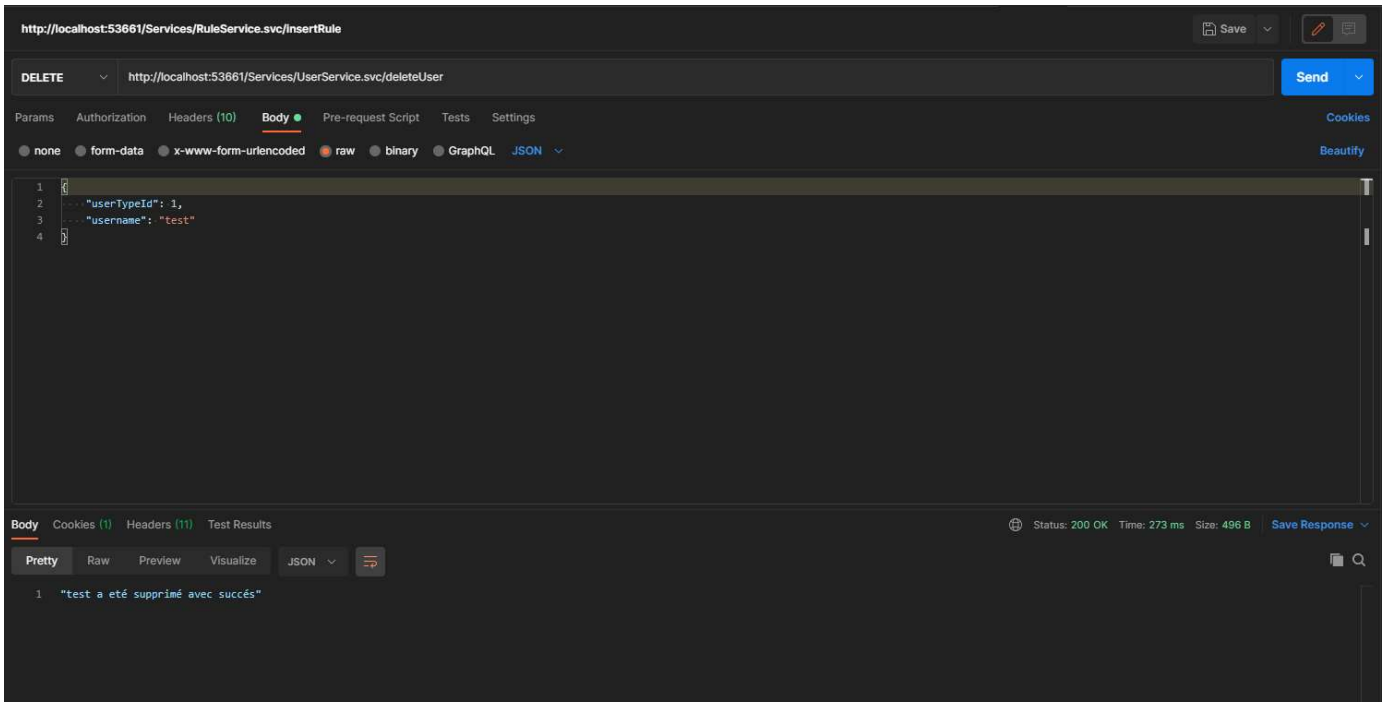

Figure 69: Integration test of delete user

Conclusion

During this final chapter we presented the development environment, defining the various programs that we used during our project. We also illustrated the main graphical interfaces of our application and detailing the tests we made.

General conclusion

To conclude, let us recall that this project was carried out within Prosoft international, it consists of facilitating the use of existing financial applications developed by Prosoft using a chatbot.

This chatbot will reduce customer response time, offers customer support providing information about the application, and facilitates the user/machine interaction by interpreting user inputs to handle real-time actions manipulating the software automatically. To increase the chatbot's efficiency and accessibility we developed the mobile version and the desktop version.

In terms of development, we used Python as a programming language for the chatbot, specifically python's TensorFlow and NLTK (Natural Language Toolkit) libraries. As for the frontend we opted for angular 11 and as for the backend we decided to use WCF (Windows Communication Foundation).

In the end, we were able to deploy our application in order to carry out tests and repair any system failures. This project provided us with multiple challenges and obstacles, but it also taught us how to overcome these difficulties, daring us to take new alternatives for the completion of the project.

We have also followed the waterfall model which ended up being very suitable with our system.

This project was very interesting, it allowed us to explore artificial intelligence, its significance and how it can be used to improve customer's experience, it also allowed us to implement everything we have learned these past three years.

However, our project remains open to several extensions seeing that for a chatbot to be successful it requires intensive monitoring and maintenance in order to keep up and anticipate the customer's needs.

Netography

[N1] http://www.prosoftinternational.com.tn/

[N2] https://www.drift.com/learn/chatbot/why-are-chatbots-important/

[N3] https://flow.ai/blog/kb-different-kinds-of-chatbots

[N4] https://eternalsunshineoftheismind.wordpress.com/2013/02/22/v-model-advantages-disadvantages-usage/

[N5] https://www.simplilearn.com/scrum-project-management-article

[N6] https://www.guru99.com/what-is-sdlc-or-waterfall-model.html

[N7] https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

[N8] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

[N9] https://www.usability.gov/how-to-and-tools/methods/use-cases.html

[N10] https://sparxsystems.com/resources/tutorials/uml/part1.html

[N11] https://techterms.com/definition/windows_10

[N12] https://angular.io/guide/architecture

[N13] https://en.wikipedia.org/wiki/Python_(programming_language)

[N14] https://en.wikipedia.org/wiki/TensorFlow

[N15] https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf

[N16] https://en.wikipedia.org/wiki/SQL_Server_Management_Studio

[N17] https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)

[N18] https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[N19] https://en.wikipedia.org/wiki/Visual_Studio_Code

[N20] https://en.wikipedia.org/wiki/Visual_Paradigm

[N21] https://balsamiq.com/wireframes/desktop/docs/intro/

[N22] https://ionic.io/resources/articles/what-is-apache-cordova

[N23] https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

[N24] https://www.techopedia.com/definition/24649/three-tier-architecture

[N25] https://www.testim.io/blog/unit-test-rest-api/

[N26] https://www.reseaucerta.org/sites/default/files/uc-intro.pdf

Annex

I. The mockups

- History interface mockup

This mockup represents all user's history interface, it allows the user to consult or delete user massages.

Figure 70: all user's history mockup desktop version

- Add user function mockup

This mockup represents the add function interface, it allows the user to add a new employee function

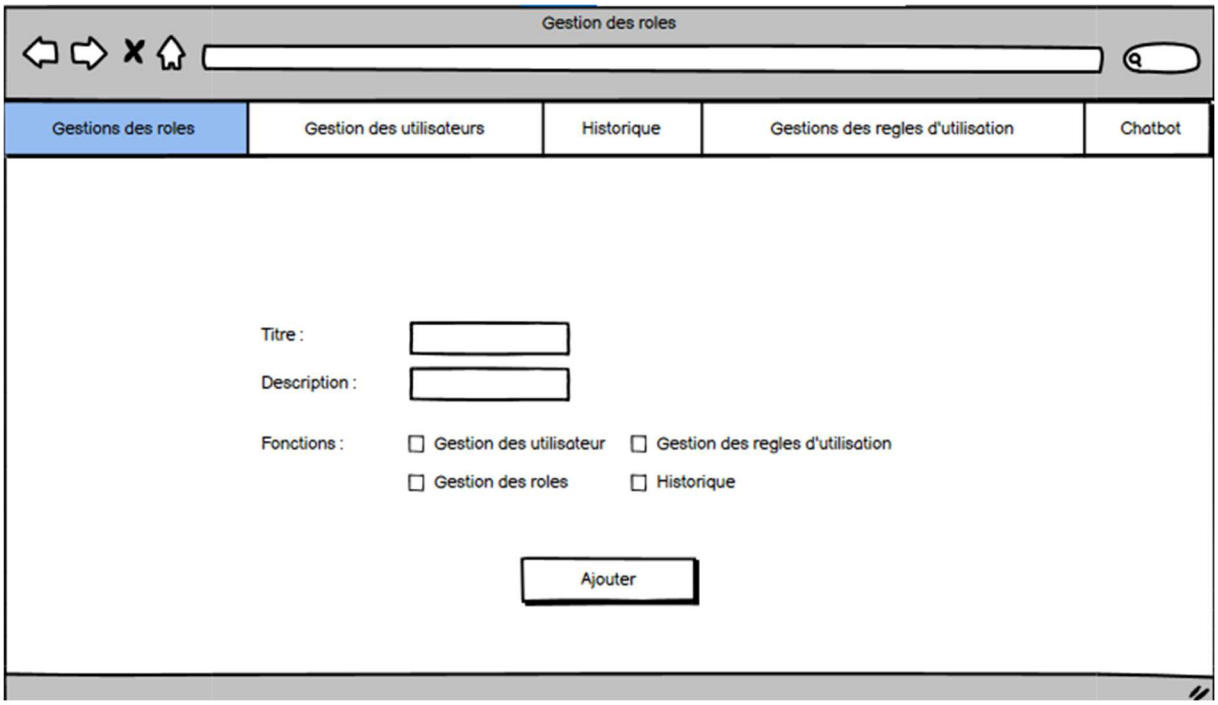

Figure 71: Add function mockup desktop version

Figure 72: Add function mockup mobile version

- Update rule mockup

This mockup represents the update rule interface, it allows the user to update an existing rule of use.

Figure 73: Update rule mockup mobile version

- Add user mockup

This mockup represents the add user interface, it allows the administrator to add a new employee.

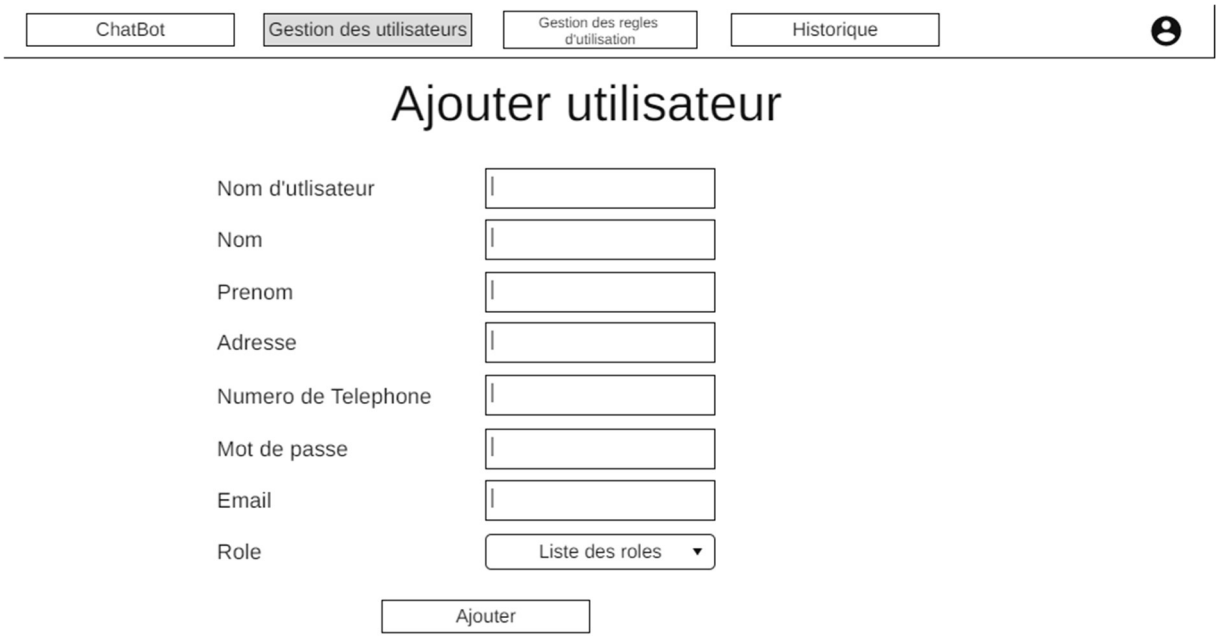

Figure 74: Add user mockup desktop version

Figure 75: Add user mockup mobile version

II. The graphical interfaces

- Menu

Allows the user to navigate through the different components of the application

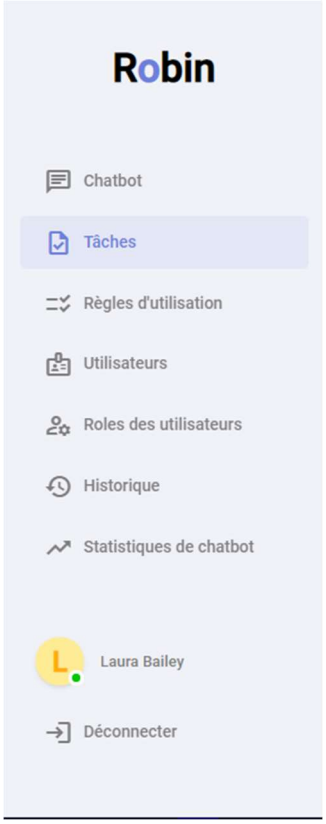

Figure 76: Menu interface desktop version

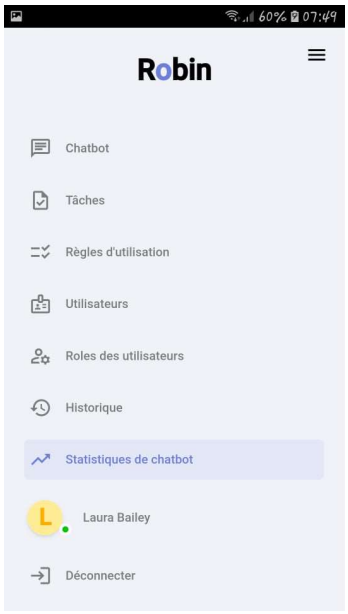

Figure 77: Menu interface mobile version

- Personal history

Allows the user to consult his own personal message history

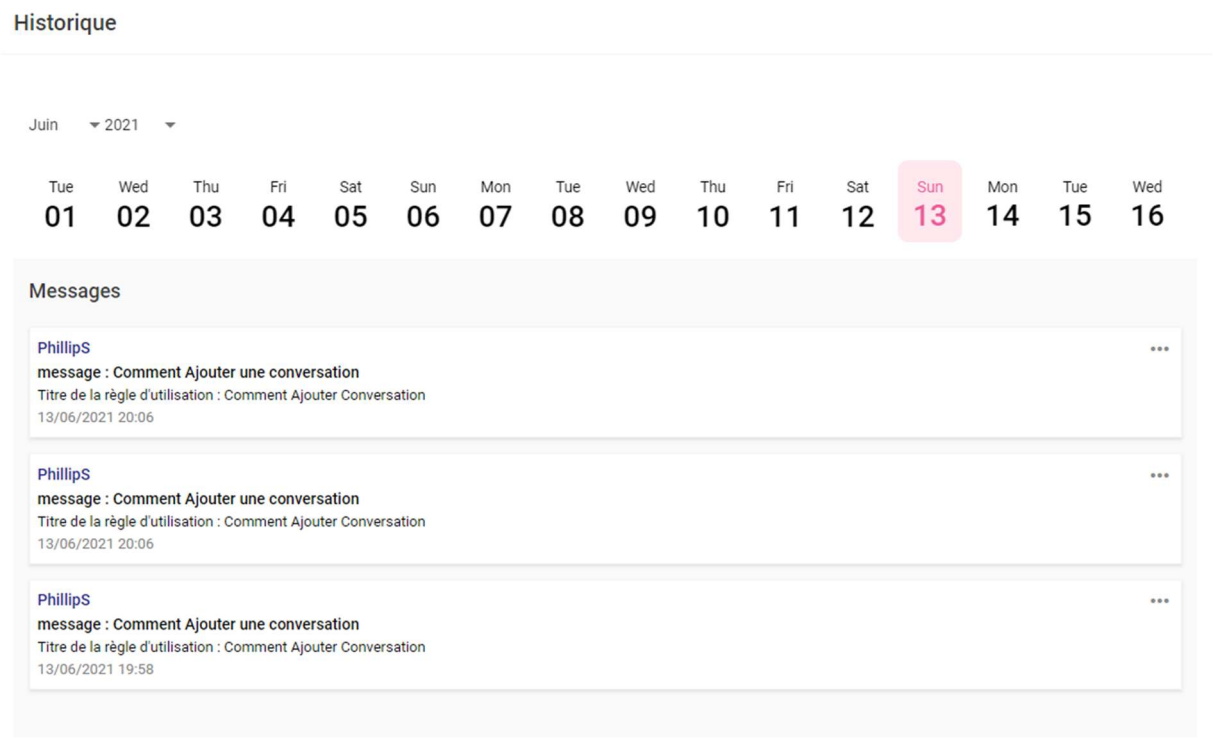

Figure 78: Personal history interface desktop version

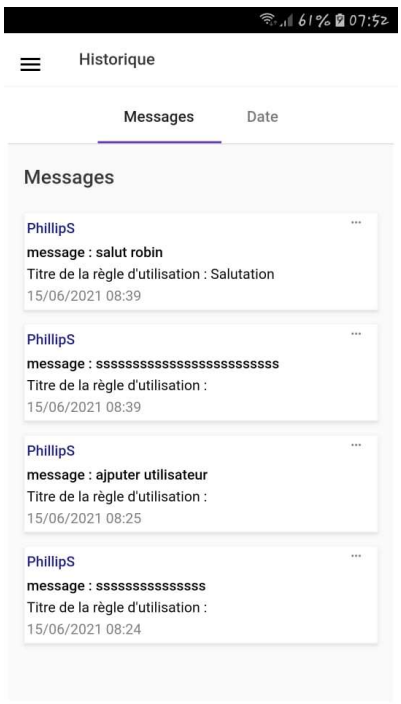

Figure 79: Personal history interface mobile version

- Add conversation

Allows the user to add a new conversation rule to the chatbot

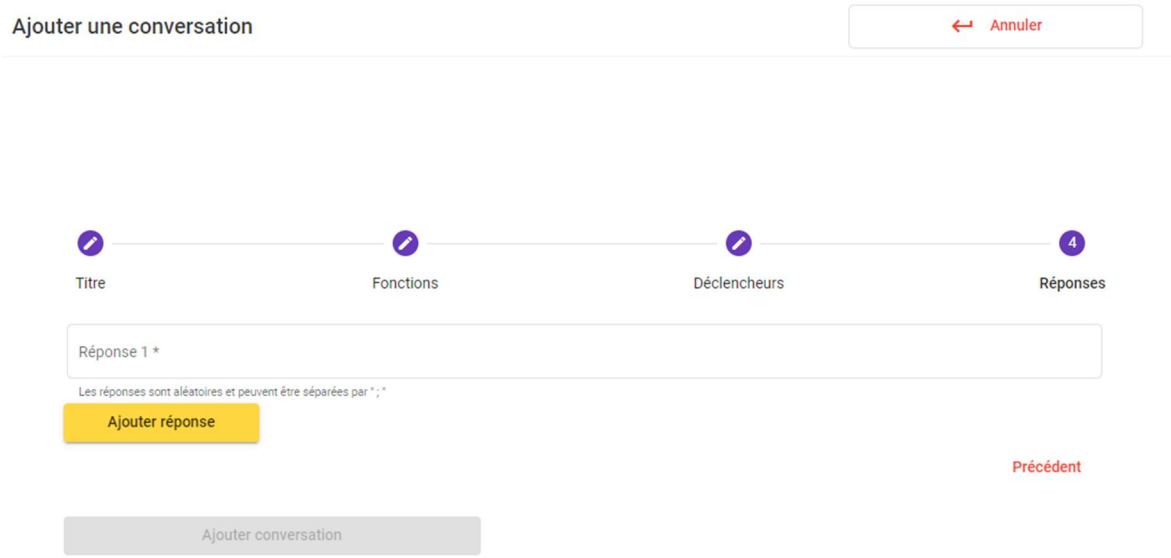

Figure 80: Add conversation interface desktop version

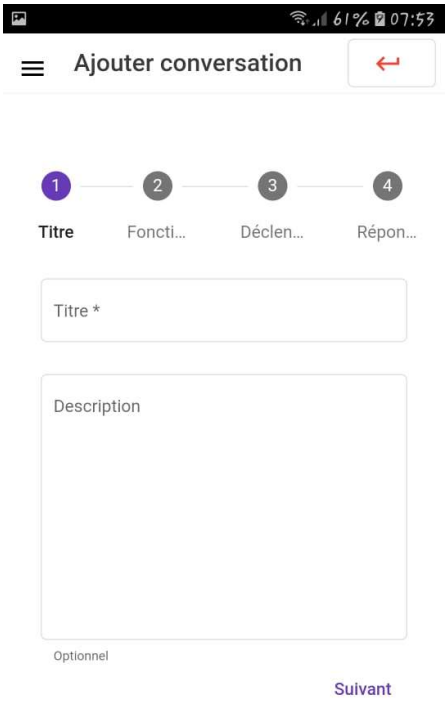

Figure 81: Add conversation interface mobile version

- Update task

Allows the user to update an existing task

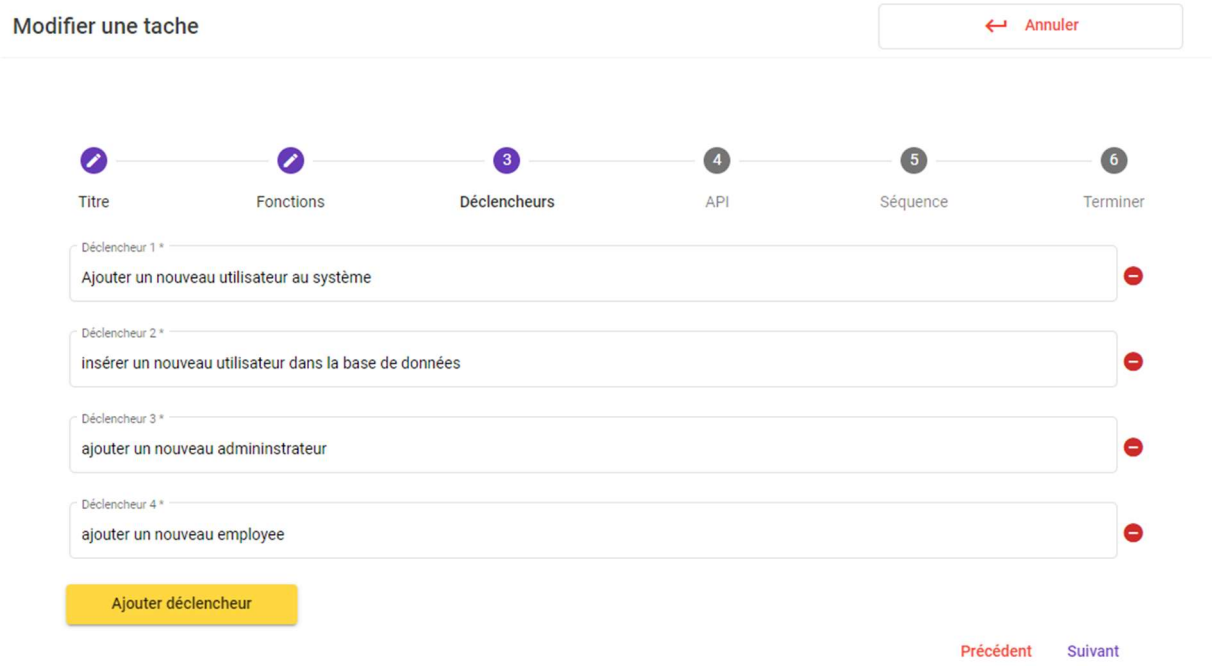

Figure 82: Update task interface desktop version

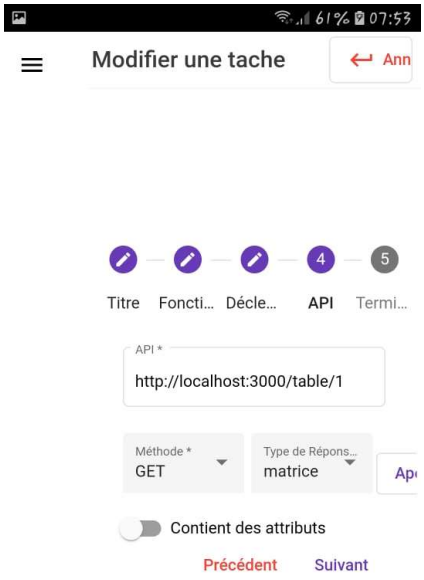

Figure 83: Update task interface mobile version

- Update user

Allows the administrator to update an existing employee

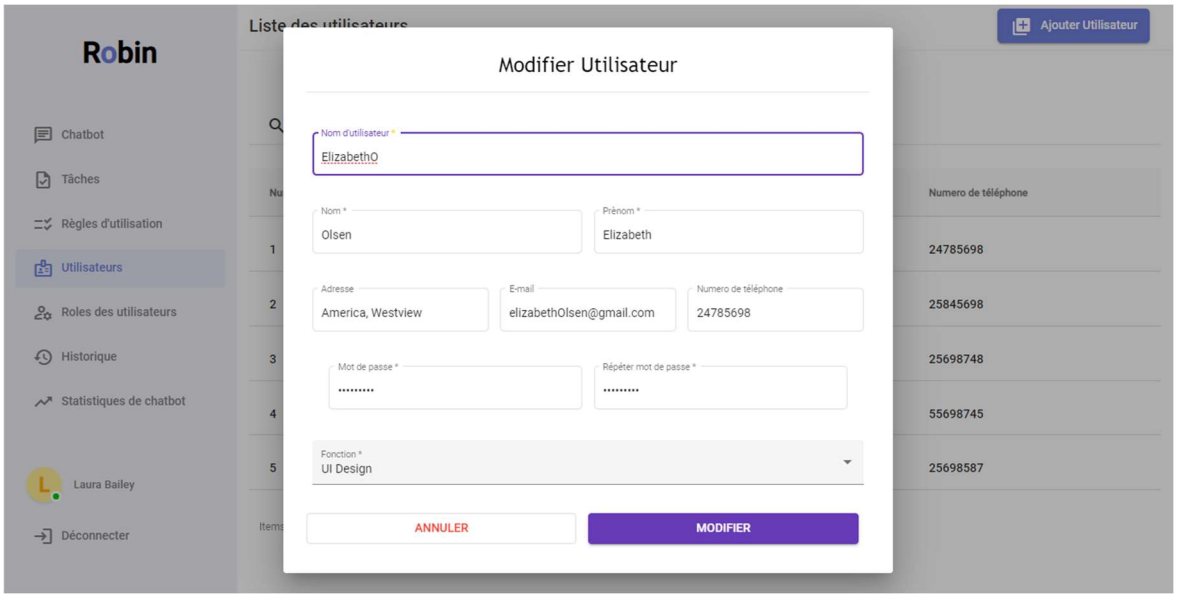

Figure 84: Update User Interface desktop version

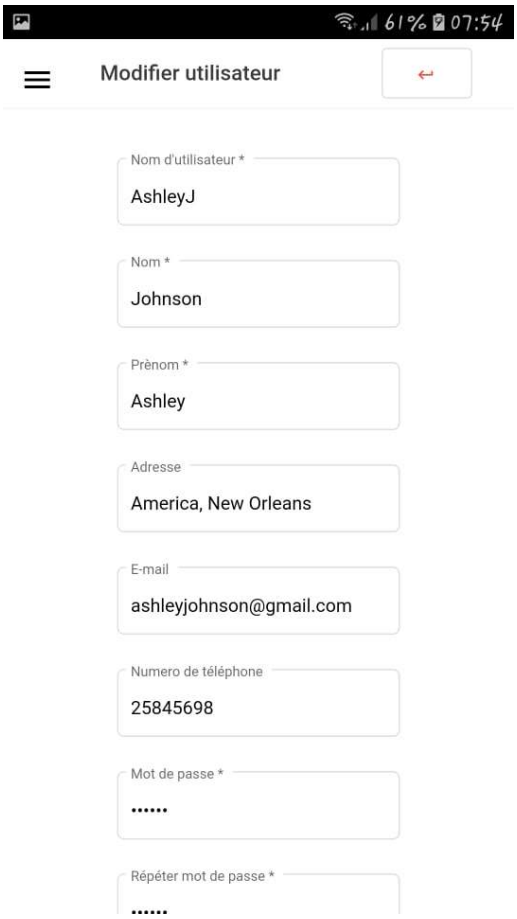

Figure 85: Update User Interface mobile version

- Add function

Allows the administrator to add a new employee function

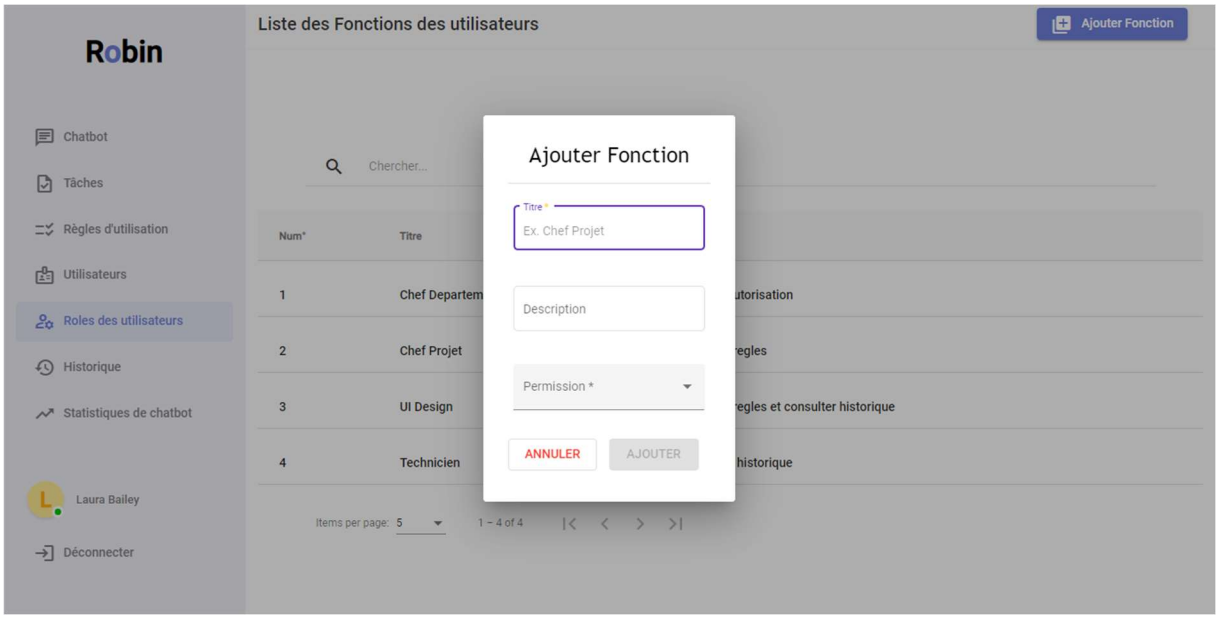

Figure 86: Add function interface desktop version

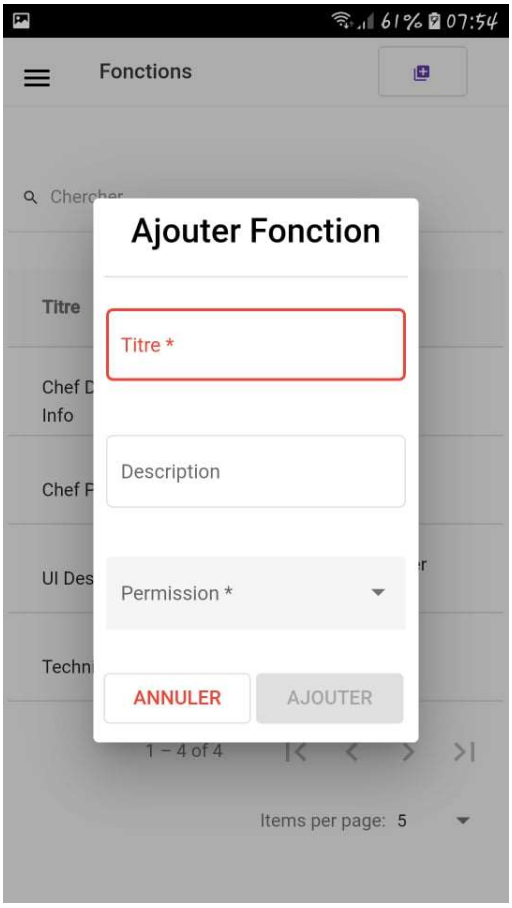

Figure 87: Add function interface mobile version

- Consult statistics

Allows the user to consult statistics' details

Figure 88: Consult statistics interface desktop version

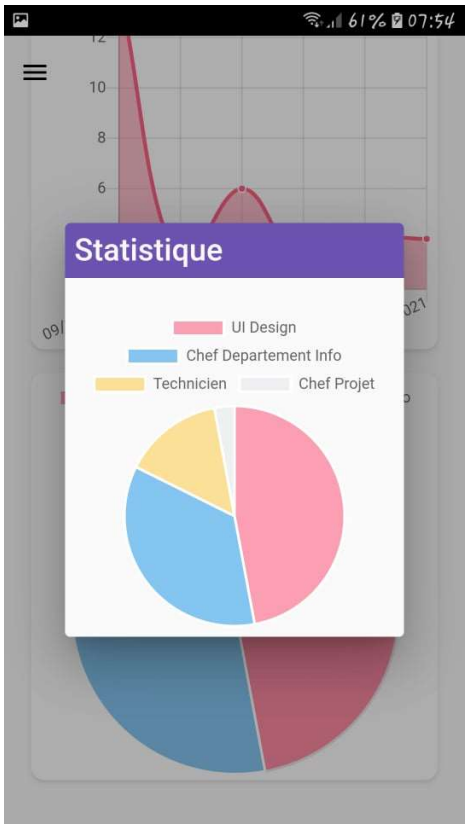

Figure 89: Consult statistics interface mobile version

- Error page

This is the interface that appears when the user enters a wrong URL (Uniform Resource Locator) path

Figure 90: Error page desktop version

الخلاصة

تم إعداد هذا التقرير كجزء أخير من مشروع التخرج بهدف الحصول على دبلوم الترخيص التطبيقي في تكنولوجيا الحاسوب.

تم إجراء التدريب داخل منظمة بروسوفت الدولية تهدف إلى تسهيل تطبيقاتها المالية من خلال متحدث آلي. لتنفيذ هذا المشروع ، استخدمنا لغة النمذجة الموحدة كلغة نمذجة و بيثون لبناء المتحدث لآلي و أنغيلار للواجهة الأمامية و مؤسسة اتصالات ويندوس للواجهة الخلفية مع قاعدة بيانات مايكروسوفت كنظام إدارة قواعد البيانات.

كلمات المفاتيح متحدث آلي, ذكاء إصطناعي, النمذجة الموحدة, بيثون, انغيلار11, مؤسسة اتصالات ويندوس, قاعدة بيانات مايكروسوفت.

Résumé

Ce rapport a été préparé en tant que partie finale d'un projet de fin d'études dans le but d'obtenir un diplôme de licence appliquée en technologies de l'informatique. Le stage s'est déroulé au sein de l'organisation Prosoft internationale. Elle vise à faciliter l'utilisation de ses applications financières via un chatbot développé dans une version desktop et une version mobile. Pour mettre en œuvre ce projet, nous avons utilisé UML (Unified Modeling Language) comme langage de modélisation, Python pour créer le chatbot, Angular 11 pour le développement frontend et WCF (Windows Communication Foundation) pour le backend avec Microsoft SQL Server comme système de gestion de base de données.

Mots clés: Chatbot, AI, UML, Python, Angular 11, WCF, Microsoft SQL Server.

Abstract

This report has been prepared as a final part of a graduation project for the purpose of obtaining an Applied Licensing Diploma in Computer Technologies. The internship was conducted within the organization Prosoft international. It aims to facilitate the use of its financial applications through a chatbot developed into a desktop version and a mobile version. To implement this project, we used UML (Unified Modeling Language) as the modeling language, Python to build the chatbot, Angular 11 for frontend development and WCF (Windows Communication Foundation) for the backend along with Microsoft SQL server as the database management system.

Key words: Chatbot, AI, UML, Python, Angular 11, WCF, Microsoft SQL Server.