Republic of Tunisia
The Ministry of Higher Education
and Scientific Research

The University of Carthage

The Higher Institute of Information and
Communication Technologies

END-OF-STUDY PROJECT REPORT

Submitted in Partial Fulfillment of the Requirements for the
Bachelor Degree in Computer Science

Field of Study : Software Engineering and Information Systems

Modelling and Realization of an Accessible Mobile Banking Solution

By
Ben Moulehem NOUR EL-HOUDA
AND
Ben Yahia WIEM

Conducted within Attijari Bank

Publicly defended on May 25, 2024 in front of the jury members:

Chairman:	Mrs. Syrine KHEMIRI, Sierra Bravo Intelligence
Reporter:	Mrs. Amira BELHEDI, Teacher, ISTIC
Examiner:	Mrs. Mariem MASMOUDI, Teacher, ISTIC
Professional Supervisor:	Mr. Adel CHEBCHOUB, Director, Attijari Bank
Academic Supervisor:	Dr. Ibtissem BEN OTHMAN, Teacher, ISTIC

Academic Year: 2023-2024

Republic of Tunisia
The Ministry of Higher Education and Scientific Research

The University of Carthage

The Higher Institute of Information and Communication Technologies

END-OF-STUDY PROJECT REPORT

Submitted in Partial Fulfillment of the Requirements for the
BACHELOR DEGREE IN COMPUTER SCIENCE

Field of Study : Software Engineering and Information Systems

Modelling and Realization of an Accessible Mobile Banking Solution

By
BEN MOULEHOM NOUR AL HOUDA
AND
BEN YAHIA WIEM

Conducted within Attijari Bank

Authorization of graduation project report submission:

Professional Supervisor:
Mr. Chebchoub ADEL

Issued on :

Signature:

Academic Supervisor: Dr. Ben Othman IBTISSEM

Issued on :

Signature:

Dedications

It is my genuine gratefulness and warmest regard that I dedicate this work to everyone that helped me throughout this journey.

Above all, I would like to thank my parents for their patience and unwavering support, their understanding, guidance, and aid have been invaluable and have played a significant role in my journey.

I want to also express my gratitude to my partner **Nour BEN MOULEHEM**, who has made the completion of this project possible, her resolve has been a source of inspiration, encouraging me to stay on track and complete this journey.

Moreover, I would like to convey special gratefulness to my friends and the family members whom extended their helping hands in dire times.

Lastly, but certainly not least, I would like to pay a special tribute to our supervisor, **Mr. Adel CHEBCHOUB**. His faith in our abilities, continuous guidance, and words of encouragement have been a significant driving force behind our work.

To everyone, your generosity and encouragement have been invaluable to me, thank you.

Wiem Ben Yahia

Dedications

First and foremost, I express my gratitude to God Almighty for giving me the strength to pull through.

I thank my parents for their unwavering belief in me. Their words and encouragement were my fuel and I dedicate this work for them. Words aren't enough to express my gratitude for what they've done for me. I owe them everything and aspire to make them proud.

I'm deeply grateful to my partner **Wiem BEN YAHIA**, who made the journey not only professional but also special and enjoyable. I have learned a lot from you and you have undeniably changed me for the better. I wish you every success and all the best on your journey. Thank you for being my partner and my closest friend. This achievement is as much yours as it is mine.

I thank the setting of Attijari Bank, especially our supervisor Mr. Adel Chabchoub, for his professionalism and for supporting and encouraging us.

Nour Ben Moulehem

Acknowledgements

We are reserving this page to extend our most valuable acknowledgments and appreciations to the people that aided us throughout our end-of-studies project.

First and foremost, we would like to express our gratitude to **Dr. Ibtissem BEN OTHMAN**, for constantly looking out for us, giving us constructive assessment, moreover, her kindness granted us the strength to overcome hardships. We appreciate her commitment and dedication to the project.

We would like to extend our appreciation to **Mr. Adel CHEBCHOUB** for supporting us and believing in our abilities to achieve the project's goal. We would like to thank him for the quality and the seriousness of his guidance. His unprecedented communication and leadership abilities assist massively in shaping us professionally.

Our profound thanks go out to the team and interns at **Attijari Bank** for keeping a collaborative environment through their expertise and kinship. Their professional help assisted in the execution of various tasks, as well as their guidance clarified the setting of the office setting and work culture.

Ultimately, our warmest thanks are extended to the esteemed jury members for their expertise in giving us constructive evaluation, in hopes that our project exceeds their expectations.

Contents

General Introduction **1**

1 Project Overview **3**
1.1 Introduction . . . 3
1.2 Company overview . . . 3
1.3 Project Presentation . . . 3
1.3.1 Problematic . . . 4
1.3.2 Study of The Existing . . . 4
1.3.3 Proposed Solution . . . 5
1.4 Work Methodology and Planning . . . 6
1.4.1 Agile methodology . . . 6
1.4.2 Scrum . . . 6
1.5 Conclusion . . . 7

2 State Of The Art **8**
2.1 Introduction . . . 8
2.2 Virtual Assistant . . . 8
2.2.1 Natural Language Processing . . . 8
2.2.2 Intent recognition . . . 9
2.3 Neural Network Architectures . . . 9
2.3.1 Recurrent Neural network (RNN) . . . 9
2.3.2 Transformer . . . 11
2.4 Conclusion . . . 13

3 Requirements Specifications **14**
2.1 Introduction . . . 14
2.2 Functional requirements . . . 14
2.3 Non-functional requirements . . . 15
2.4 Identification of the actors . . . 15
2.4.1 Global use case diagram . . . 16
2.4.2 Product's Backlog . . . 17
2.5 Technical requirements . . . 18
2.6 Conclusion . . . 21

4 Sprint 1 : Authentication flow **22**
3.1 Introduction . . . 22
3.2 Sprint 1 Backlog Identification . . . 22
3.3 Sprint 1 refinement . . . 22
3.3.1 Refinement of the « Register » use case . . . 22

3.3.2 Refinement of the « Authenticate » use case . . . 23
3.3.3 Refinement of the « Recover Password » use case . . . 25
3.4 Sprint 1 Modelling . . . 26
3.4.1 « Register » Use Case Modelling . . . 26
3.4.2 « Authenticate » Use Case Modelling . . . 28
3.4.3 « Recover Password » Use Case Modelling . . . 30
3.5 Sprint 1 Class Diagram . . . 31
3.6 Deployment Diagram . . . 32
3.7 Realization . . . 33
3.7.1 Realisation of « Register » Use Case . . . 33
3.7.2 Realisation of « Recover Password » Use Case . . . 34
3.7.3 Realisation of « Authenticate » Use Case . . . 35
3.8 Conclusion . . . 35

5 Sprint 2 : Virtual Assistant 36
5.1 Introduction . . . 36
5.2 Libraries used . . . 36
5.3 Machine learning life cycle . . . 37
5.3.1 Data preparation . . . 37
5.3.2 Model development . . . 39
5.3.3 Model Deployment . . . 40
5.4 Conclusion . . . 41

6 Sprint 3: Operations and Transfers 42
6.1 Introduction . . . 42
6.2 Sprint 3 Backlog Identification . . . 42
6.3 Sprint 3 Refinement . . . 42
6.3.1 Refinement of the « Check account's Balance » Use Case . . . 43
6.3.2 Refinement of the « Check history of Operations » Use Case . . . 43
6.3.3 Refinement of the « Manage Card » Use Case . . . 44
6.3.4 Refinement of the « Check history of transfers » Use Case . . . 45
6.3.5 Refinement of the « Manage Beneficiaries » Use Case . . . 46
6.3.6 Refinement of the « Transfer Money » Use Case . . . 47
6.3.7 Refinement of the « Change Password » Use Case . . . 49
6.4 Sprint 3 Modelling . . . 49
6.4.1 « Check account's Balance » Use Case Modelling . . . 49
6.4.2 « Check history of Operations » Use Case Modelling . . . 50
6.4.3 « Manage Card » Use Case Modelling: . . . 52
6.4.4 « Check history of transfers » Use Case Modelling . . . 53
6.4.5 « Manage Beneficiaries » Use Case Modelling . . . 54
6.4.6 « Transfer Money » use case Modelling . . . 56
6.4.7 « Change Password » Use Case Modelling . . . 57
6.5 Sprint 3 Class Diagram . . . 58
6.6 Realization . . . 59
6.6.1 «Check Balance» Use Case Realization . . . 59
6.6.2 «Consult Operations» Use Case Realization . . . 60
6.6.3 «Check Card» Use Case Realization . . . 61
6.6.4 « Check history of transfers » Use Case Realization . . . 61
6.6.5 « Manage Beneficiaries » Use Case Realization . . . 62

6.6.6 « Transfer Money » use case Realization 63
6.6.7 « Change Password » Use Case Realization 64
6.6.8 «Change Settings» Backlog Item Realization: 64
6.7 Conclusion 65

7 Sprint 4 : Administration and Client Relations 66
7.1 Introduction 66
7.2 Sprint 4 Backlog Identification 66
7.3 Refinement of Sprint 4 66
7.3.1 Refinement of the « Contact Assistants » use case 67
7.3.2 Refinement of the « Check notifications » use case 67
7.3.3 Refinement of the « Manage Assistants » use case 68
7.3.4 Refinement of the « Manage Clients requests » use case 69
7.4 Modelling of Sprint 4 70
7.4.1 « Contact Assistants » Use Case Modelling 70
7.4.2 « Check notifications » Use Case Modelling 71
7.4.3 « Manage Assistants » Use Case Modelling 72
7.4.4 « Manage Clients Request » Use Case Modelling 73
7.5 Global Class Diagram 74
7.6 Realisation 75
7.6.1 Realization of « Contact Assistants » Use Case 75
7.6.2 Realization of « Check Notifications » Use Case 76
7.6.3 Realization of « View dashboard » Use Case 76
7.6.4 Realization of « Manage Assistants » Use Case 77
7.6.5 Realization of « Manage Client's requests » Use Case 77
7.7 Conclusion 78

General Conclusion 79

Netography 81

List of Abbreviations

URL :*Uniform Resource Locator*

REST API :*Representational State Transfer Application Programming Interface*

API :*Application Programming Interface*

ML :*Machine Learning*

NLP :*Natural Language Processing*

NN :*Neural Network*

RNN :*Recurrent Neural Network*

BERT :*Bidirectional Encoder Representations from Transformers*

GPU :*Graphics processing unit*

WCAG :*Web Content Accessibility Guidelines*

W3C :*World Wide Web Consortium*

List of Figures

1.1 Company's logo [1] . . . 3
1.2 WeBank [2] . . . 5
1.3 Scrum framework.[3] . . . 7
2.4 Intent detection with similar prompts [4] . . . 9
2.5 Processing a prompt . . . 9
2.6 Simplified Neural Network [5] . . . 10
2.7 simplified architecture of Recurrent Neural Network [6] . . . 10
2.8 LSTM (Long Short Term Memory) [6] . . . 11
2.9 Transformer architecture [7] . . . 12
2.10 Simplified Architecture of BERT . . . 13
2.1 Global Use Case Diagram. . . . 17
3.1 Refinement of the « Register » use case . . . 23
3.2 Refinement of the « Authenticate » use case . . . 24
3.3 Refinement of the « Recover Password » use case . . . 25
3.4 Class Diagram for the « Register » use case . . . 26
3.5 Class Diagram for the « Validate Email » use case . . . 26
3.6 Sequence Diagram for the « Register » Use Case. . . . 27
3.7 Sequence Diagram for the « Validate Email » Use Case. . . . 28
3.8 Class Diagram for the « Authenticate » use case . . . 28
3.9 Sequence Diagram for the « Authenticate » Use Case . . . 29
3.10 Class Diagram for the « Recover Password » use case . . . 30
3.11 Sequence Diagram for the « Recover Password » Use Case . . . 31
3.12 Sprint 1 Class Diagram . . . 32
3.13 Deployment Diagram . . . 32
3.14 Multi-Step Registration . . . 33
3.15 Account Verification . . . 34
3.16 Forgot Password Process . . . 34
3.17 Authenticate Interface . . . 35
5.18 Machine learning life cycle . . . 37
5.19 Template of 'virement' intent . . . 38
5.20 Template of prepend request entity . . . 38
5.21 Distribution of Different Intents . . . 38
5.22 Model Architecture Layers . . . 39
5.23 Flask's logo . . . 40
5.24 Virtual Assistant Integration . . . 41
6.1 Refinement of « Check account's Balance » use case . . . 43
6.2 Refinement of « Check history of Operations » use case . . . 44
6.3 Refinement of « Manage Card » use case . . . 45
6.4 Refinement of « Check history of transfers » use case . . . 46

6.5 Refinement of « Manage Beneficiaries » use case 47
6.6 Refinement of « Transfer Money » use case 48
6.7 Refinement of « Change Password » use case 49
6.8 « Check account's Balance » Class Diagram 50
6.9 « Check account's Balance » Sequence Diagram 50
6.10 « Check history of Operations » Class Diagram 51
6.11 « Check history of Operations » Sequence Diagram 51
6.12 « Manage Card » Class Diagram 52
6.13 « Manage Card » Sequence Diagram 52
6.14 « Check history of transfers » class diagram 53
6.15 «Check history of transfer» Sequence Diagram 53
6.16 « Manage Beneficiaries » class diagram 54
6.17 « Manage Beneficiaries » Sequence Diagram 54
6.18 « Add Beneficiary » Sequence Diagram 55
6.19 « Update Beneficiary » Sequence Diagram 55
6.20 « Delete Beneficiary » Sequence Diagram 56
6.21 « Transfer Money » class diagram 56
6.22 « Transfer Money » Sequence Diagram 57
6.23 « Change Password » class diagram 58
6.24 « Change Password » Sequence Diagram 58
6.25 Sprint 3 Global Class Diagram 59
6.26 «Check account's balance» Use Case Realization 60
6.27 «Consult operations» Use Case Realization 60
6.28 «Manage card» Use Case Realization 61
6.29 «Check history of transfers» Use Case Realization 61
6.30 « Manage Beneficiaries » Use Case Realization 62
6.31 Transfer Money Process 63
6.32 « Change password» Use Case Realization 64
6.33 Change Settings 64
6.34 Color Blind Modes 65
7.1 Refinement of the « Contact Assistants » use case 67
7.2 Refinement of the « Check notifications » use case 68
7.3 Refinement of the « Manage Assistants » use case 68
7.4 Refinement of the « Manage Clients requests » use case 69
7.5 Class diagram for the « Contact Assistants » Use Case. 70
7.6 « Contact Assistants » Sequence Diagram 71
7.7 Class diagram for the « Check notifications » use case. 71
7.8 « Check notifications » Sequence Diagram 72
7.9 Class diagram for the « Manage Assistants » use case. 72
7.10 « Manage Assistants » Sequence Diagram 73
7.11 Class diagram for the « Manage Clients Request » use case. 73
7.12 « Manage Clients Request » Sequence Diagram 74
7.13 Global Class Diagram 75
7.14 « Contact Assistants » use case realization 76
7.15 « Check Notifications » use case realization 76
7.16 « View dashboard » use case realization 77
7.17 « Manage Assistants » use case realization 77
7.18 « Manage Client's requests » use case realization 78

List of Tables

2.1 Detailed functionalities of the actors 16
2.2 Product backlog 18
3.1 Sprint backlog 22
3.2 « Register » use case refinement. 23
3.3 « Authenticate » use case refinement. 24
3.4 « Recover Password » use case refinement. 25
5.6 Evaluation Results 40
6.1 Sprint backlog 42
6.2 « Check account's Balance » use case refinement. 43
6.3 « Check history of Operations » use case refinement. 44
6.4 « Manage Card » use case refinement. 45
6.5 « Check history of transfers » use case refinement. 46
6.6 « Manage Beneficiaries » use case refinement. 47
6.7 « Transfer Money » use case refinement. 48
6.8 « Change Password » use case refinement. 49
7.1 Sprint 4's Backlog 66
7.2 « Contact Assistants » use case refinement. 67
7.3 « Check notifications » use case refinement. 68
7.4 « Manage Assistants » use case refinement. 69
7.5 « Manage Clients requests » use case refinement. 70

General Introduction

The financial sector contributes massively to a country's economy. A well-performing financial sector can boost the overall economic health, and the reverse is also true. In this context, this critical role drives banks to focus on enhancing their performance metrics or Key Performance Indicators (KPIs).

One effective strategy to augment their KPIs is expanding their customer base by promoting various services, such as banking and finance applications.

In today's digital age, particularly in Tunisia, such applications are gaining immense relevance. Given the rise in mobile device usage, it's almost a prerequisite for every bank to offer a platform for remote account management, and to meet this demand, they are increasingly investing in the development of mobile applications catering to their customer's needs.

However, one aspect often overlooked, which can significantly impact customer reach and consequently, the return on investment (ROI), is the accessibility of these applications. Accessibility ensures that applications can be fully utilized by everyone, including individuals with disabilities.

As access to information and communication technologies has become a fundamental human right, Attijari Bank recognized this, and believes in the benefits that it will contribute to their innovations. However, it has been identified that WeBank, its mobile application for conducting daily financial transactions entirely online, is not accessible to people with disabilities and vision impairments. As a result, the bank is committed to redesigning its application, emphasizing its dedication to inclusivity and ease of use.

To present our end-of-studies project more effectively, the various functionalities of the application will be described in this report, which consists of seven chapters.

- Chapter 1, "Project Overview", will give a general presentation of the project, starting by introducing the host enterprise, a study of the existence and the solution proposed, finishing by a description of the project management methodology adopted.
- Chapter 2, "State Of The Art", will give a detailed description of the building of a virtual assistant starting by defining Natural Language Processing, moving onto intent recognition then Neural Network architectures and finishing off by detailing Transformers.
- Chapter 3, "Requirements Specifications", will give an outline of the functional and non-functional requirements of the system, including a description of the system's actors, the global use case diagram, the product backlog, and finally detailing the the technical requirements.
- Chapter 4 entitled "Sprint 1 : Authentication flow", will detail the first sprint, discussing the backlog items for this sprint and providing class and sequence diagrams to illustrate the implementation details, concluding with the realization.

- Chapter 5 entitled “Sprint 2 : Virtual Assistant”, will start by introducing the libraries used for the virtual assistant, and delineate the machine learning life-cycle.
- Chapter 6 entitled “Sprint 3 : Balance and transactions”, will include a detailed explanation for the third sprint, introducing the backlog items and providing the UML models to illustrate the implementation details.
- Chapter 7 entitled “Sprint 4 : Administration and Client Relations ”, will offer a thorough account of the last sprint, it will go into the backlog items addressed during this sprint and includes class and sequence diagrams. The chapter concludes by detailing the outcomes achieved during the sprint.

To conclude, a general conclusion that summarizes our work and offers potential for this project will be included.

Chapter 1

Project Overview

1.1 Introduction

This following inaugural chapter, will include a description of the host company, the project's presentation, The problematic and its proposed solution and lastly a definition of the adopted methodology.

1.2 Company overview

Attijari Bank, is one of the largest banks in North Africa, with its presence extending to over 20 countries.
Founded in Tunisia in 1971, it has since developed its activities with the aim of becoming a leader in the banking sector. As such, it offers a complete range of financial products and services for individuals, businesses, and public institutions.
The bank also stands out for its commitment to innovation and digitization, thus offering modern and practical banking solutions. Its drive for transition to digital is not limited to the products it offers to its customers but also targets the internal processes of the bank.

Figure 1.1: Company's logo [1]

1.3 Project Presentation

Throughout this section, the project context will be detailed by outlining the problematic, a thorough study and critique of the existing, consequently unveiling the solution proposed.

1.3.1 Problematic

Building accessible solutions means creating applications that can be used by everyone. However, some of the best digital solutions lack the necessary accessibility features, leaving people with disabilities behind and strip them of their right to use technology. People with disabilities often rely on screen readers, a feature that converts screen content into speech, allowing them to interact through audio feedback. To make mobile apps compatible with screen readers, important code needs to be included, which developers unfortunately seem to skip.

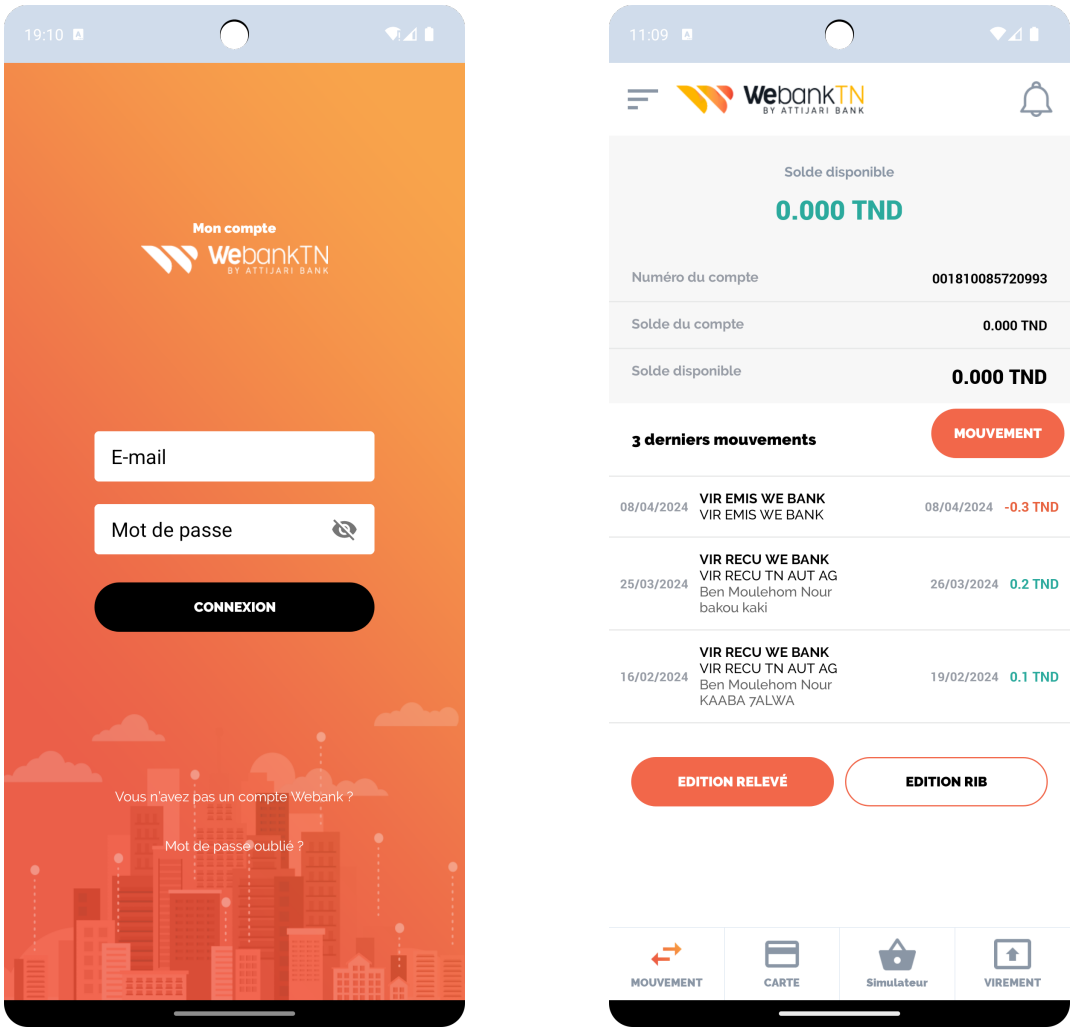
1.3.2 Study of The Existing

Study of the existing is a critical step when progressing towards the creation of a new system that meets the client's specifications. It supports the building of the product in a sense that it provides learning points and insights over the market. It helps with decision making and the creation of unique product that can significantly stand out.

Description of the existing

WeBank, a digital solution developed by Attijari Bank, designed to offer a set of banking services in a user-friendly interfaces. The aim was simplifying bank account management. However, in figures 1.2, it has been observed that it lacks certain accessibility features, which are significant for ensuring that all clients can fully utilize the services. People with disabilities are encountering issues with various aspects of the application, including:

- **Lack of Accessibility Features:** The application is missing necessary accessibility features. For example, form labels, which provide context and clarity for users, are notably absent. This make it difficult for users to interact with the forms on the platform.
- **Incompatibility with Assistive Technologies:** During testing with screen readers, it has been found that these tools do not recognize elements such as buttons in the application due to a lack of accessibility in the code. Namely, when trying to test VoiceOver on the authentication button, it will only announce 'button'. This doesn't deliver any insight into the purpose of that button or what will happen when it's pressed. Screen readers, such as VoiceOver for iOS and TalkBack for Android, are crucial assets that guide people with disabilities into technologies.
- **Contrast and Color Choices:** The application's color scheme and contrast levels are not optimal for users with visual impairments. High contrast and color differentiation are crucial for readability and ease of use, especially for users with conditions like color blindness or low vision.



(a) WeBank Authentication [2] (b) WeBank Interface [2]

Figure 1.2: WeBank [2]

– **Complex Navigation:** People with disabilities often analyze the content of the screen before deciding on the action they are going to take. WeBank's interface lacks semantic clarity from the perspective of these users, making it a challenge for them to make decisions and interact with the platform adequately.

– **Lack of Customization:** WeBank's interface is not versatile enough to accommodate all users, and the white color scheme may not be suitable for everyone.

1.3.3 Proposed Solution

In order to ensure that the online services offered by WeBank holds all their clientele, including those with disabilities, particularly with visual impairments, a decision has been made to revamp WeBank. A thorough study and analysis of the current system have highlighted the essential elements needed to enhance the application's accessibility:

– **Commitment to the WCAG/W3C accessibility principles:** The Web Content Accessibility Guidelines (WCAG) established by the World Wide Web Consortium (W3C) provide a set of guidelines for making software products more acces-

sible. These guidelines are based on four fundamental principles: content must be *perceivable*, *operable*, *understandable*, and *robust*.

- **Readability with screen reader:** The screen reader must identify content within the interface and announce the label and purpose of each component.
- **Interface customization:** Customization like changing the color schemes (dark, light, protanopia, deuteraneupia...) can significantly enhance user experience. This flexibility augment the user experience and ensure that the product reaches a wider audience.
- **Intuitive Interfaces:** A straightforward and simple user interface designed to help people with disabilities make decisions more quickly, incorporating best practices in layout design. This includes making buttons clear and prominent, ensuring that the text is large enough to be easily be read, and arranging the interface in a way that is intuitive and user-friendly.
- **Virtual Assistant:** Incorporate a virtual assistant to convert banking services into voice commands.

1.4 Work Methodology and Planning

This segment will include a definition of the implemented work methodology, and an overview of its implementation mechanism.

1.4.1 Agile methodology

Agile embodies the capacity to innovate and adjust in the face of evolving circumstances. It serves as a strategy for navigating and ultimately thriving in an environment marked by unpredictability and instability [8].

1.4.2 Scrum

Scrum is a nimble framework crafted to support individuals, teams, and organizations to create value through dynamic solutions for complex challenges [3]. In this case, Scrum was used because it had the following example characteristics that benefit the project:

- Scrum's incremental and iterative nature allows for simultaneous planning and execution.
- Rooted in Lean thinking and empiricism, Scrum promotes efficiency, quality, and flexibility.
- The pillars of Transparency, Inspection, and Adaptation foster client engagement and project alignment.
- Scrum's value-driven approach ensures potentially shippable increments after each sprint, empowering client decision-making.

Therefore, due to the complexity and incertitude of this project, Scrum was almost the natural choice to implement.

To further understand the framework, down below figure 1.3, shows the Scrum loop. Based off of Scrum.org's 2020 'The Scrum Guide', Scrum is composed of a Scrum team, artifacts and timeboxed events.

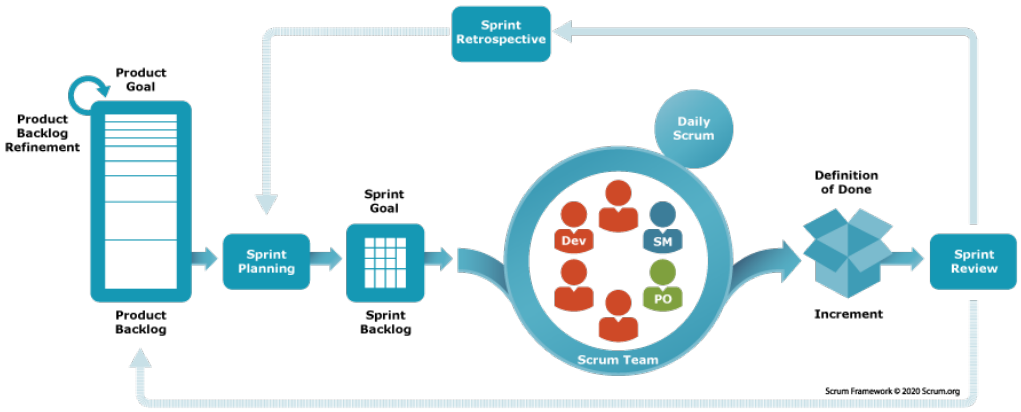


Figure 1.3: Scrum framework.[3]

1.5 Conclusion

In this chapter, we have presented the project, including a thorough study of the existing system. we have also outlined our objectives to create a secure and user-friendly application. Furthermore, we have described the methodology employed to develop the final product.

The next chapter will be dedicated to the functional and technical specification of the project, which presents the various functional and non-functional requirements, the global use case diagram, as well as the product backlog and the working environment.

Chapter 2

State Of The Art

2.1 Introduction

Virtual assistants such as Siri and Google Assistant have been around for quite some time, and since their existence they have revolutionized the way we interact with our devices. These invention have enabled all people, especially those with disabilities, to interact with technology more fluidly. In this chapter, explanations will be provided on the base of building a virtual assistant starting from defining Natural Language Processing, intent classification, Recurrent Neural Networks, transformers, and finally BERT.

2.2 Virtual Assistant

The main goal of a conversational artificial intelligence or a virtual assistant is to understand the user goal or *intention*. if a virtual assistant succeeds in extracting the meaning behind such a prompt and classify it, it can offer best possible service.

Such intelligent interactive system implements an intent classification model which is a common NLP task.

2.2.1 Natural Language Processing

Natural Language Processing (NLP) is a machine learning field that aims to understand, interpret, and extract the meaning out of the human language. There are a lot of use cases and goals tasks that falls under NLP, some of them:

- **Text Classification**: a higher-level task that classifies a document into a relevant group.
- **Machine Translation**: a task that translates a piece of text from one language to another language.
- **Content Moderation**: filtering out anything that is not relevant for a certain group. A common use case for is filtering social media posts for a group of people or country.
- **Sentiment Analysis**: classifying a given text if it has a positive or negative sentiment.

2.2.2 Intent recognition

Intent detection is a NLP task that classifies human's intention. The main objective is to categorize a prompt or command using a predefined label called *intent*. Natural language is complex, the same prompt can be expressed in various phrases. This is where the strength of an intent recognition model comes into play. Each intent can be represented by an enormous amount of different templates, which help in detecting the intent of a command, regardless of the way it's said [4].

Figure 2.4 illustrates how inputs about the weather are presented in different ways while still recognizing them as the same intent 'weather'.

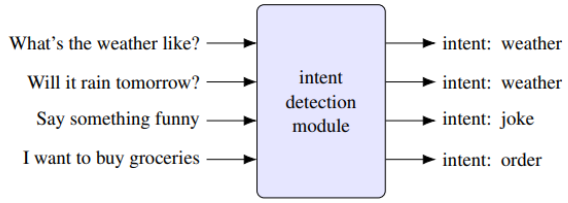


Figure 2.4: Intent detection with similar prompts [4]

some prompts that belong to the same intent can contain *entities*. These are keywords that belong to a given intent and can vary with each prompt. For example emails, names, locations, food and so on. As an example, if the prompt was 'remind me to buy groceries at 5 PM', then the detected entity would be '5 PM' representing the time (figure 2.5).

Figure 2.5: Processing a prompt

2.3 Neural Network Architectures

In the next section, a detailed explanation of Recurrent Neural network and Transformers will be outlined, and why transformers has been the chosen model.

2.3.1 Recurrent Neural network (RNN)

Neural network are deep learning models that mimic the human brain, in the figure below we can see three layers, first the information come to input layer, and then passed to the hidden layer, which represent the activity of the neurons and typically not observable from the system's input or output, and lastly the output is produced, all the inputs are given at the same time and processed sequentially (figure 2.6).

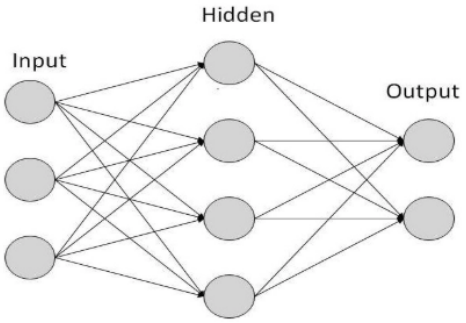


Figure 2.6: Simplified Neural Network [5]

Persisting memory is what makes Neural Network models stronger. Recurrent Neural Networks were developed to address this lack in standard NNs. Through their sequential architecture, RNNs are able to maintain memory. Compared to a traditional Neural Network, where all features are given at the same time, RNNs process features one after another or *sequentially*. Inside the architecture of RNNs, each input (x_t, where t represents the current time) is processed by the same neurons or network, then the output (h_t) is passed to the next cell. This processing occurs in a time step manner, as illustrated in figure 2.7.

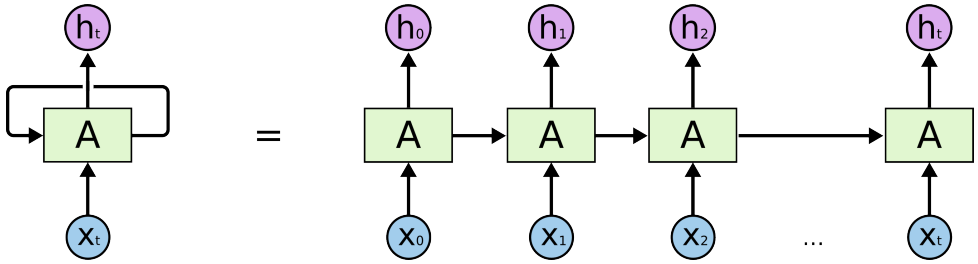


Figure 2.7: simplified architecture of Recurrent Neural Network [6]

RNN, for a long time have revolutionized the world of Machine Learning, with their simplicity. However, with time it was found that there were some challenges with RNNs:

- **Vanishing Gradient:** In some cases, some inputs require context from previous and earlier states. As the gap between dependencies get wider, the information is lost through propagation. This is what makes RNN slower and inefficient.
- **Parallelism:** As mentioned earlier, RNN process inputs sequentially. By feeding one input at a time into the model, it becomes slower and uses significant amount of computational resources.

LSTM (Long Short Term Memory)

LSTM was introduced to solve some issues with standard RNN. It is capable of remembering important information since the start of the process. It contains two hidden layers: $h(t-1)$, the hidden state from the previous step, and $c(t-1)$, the long term memory hidden state. LSTM contains specific structures called gates:

- Forget gate: decides which part of the long term memory should be removed.
- Input gate: decides which part of the hidden state should be added to the hidden state (passed on to C).

- Output gate: decides which part of the long term state should be added to the output at this time step.

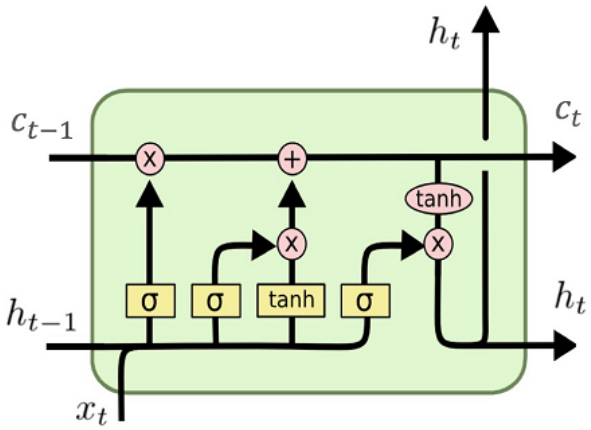


Figure 2.8: LSTM (Long Short Term Memory) [6]

With the introduction of gate structures, the model can capture long-range dependencies minimize the vanishing gradient challenge. However, the same sequential structure is still present which require computational resources making it less efficient than models that process data in parallel.

2.3.2 Transformer

The Transformer model was introduced to address some of the issues with RNNs, through its *self-attention* mechanism and parallel processing. Attention is the ability of the model to focus on the importance of a sentence. This helps the model persist information, captures long-range dependencies, operates faster due to its parallelism, making it more effective than RNNs. Internally, Transformers consist of stack of six identical layers of each of encoders and decoders. Figure 2.9 summarizes the layers that constitute each of them.
Encoder, architecture on the left, plays a pivotal role in understanding the context of a given sequence. It takes the entire input data and generates an encoded representation that holds the attention scores and relationship from the entire input sequences. Each encoder layer is consisted of one self-attention layer (Multi-Head Attention) and one feed-forward layer.
Decoder, architecture on the right, is responsible in generating sequences, one word at a time. The decoder takes two main input data: the representation generated by the encoder layer that helps guide the decoder on focusing on important tokens (or words) and the previous generated word. This mechanism helps in keeping coherency and prevent grammatical errors. Each Decoder layer consist of two attention layers and one feed forward layer [7].
Both of these structures consist of identical layers with a slight difference:

- **Input/Output embedding:** Neural network models can't deal with raw textual data. During this layer each word is translated into a dense vector [9].

- **Positional encoding:** Unlike RNN, transformers have no recurrence, making it a challenge to know the order of the words in a sequence. Positional Encoding fix this issue by integrating relevant data about the order [9].

- **Multi-Head Self-Attention layer:** There's a minor difference between the encoder and decoder here. The encoder takes all input words simultaneously and compares each word with every other word in the sequence. This aids in understanding the context and the relationship between tokens, generating an attention score matrix. On the other hand, the decoder, through its Multi-Head Attention layer (also known as the Encoder-Decoder Attention), takes the representation generated by the encoder to learn information about the input sequence and helps in predicting output words. This layer also capture context between generated words to maintain the correlation between them. The Masked Multi-head Attention prevents future leakage. As the decoder generates one token at a time, this layer masks future words to only focus or *attend* previous generated tokens [9].

- **Feed forward:** This is a neural network layer responsible for more complex calculation and goes beyond then just the connections of between words. It helps capture complex structure in a sentence [9].

- **Add & Norm:** during after every Multi-Head Attention layer the encoded input is 'added' to the output of this layer and then 'normalized', the same for feed forward layer [9].

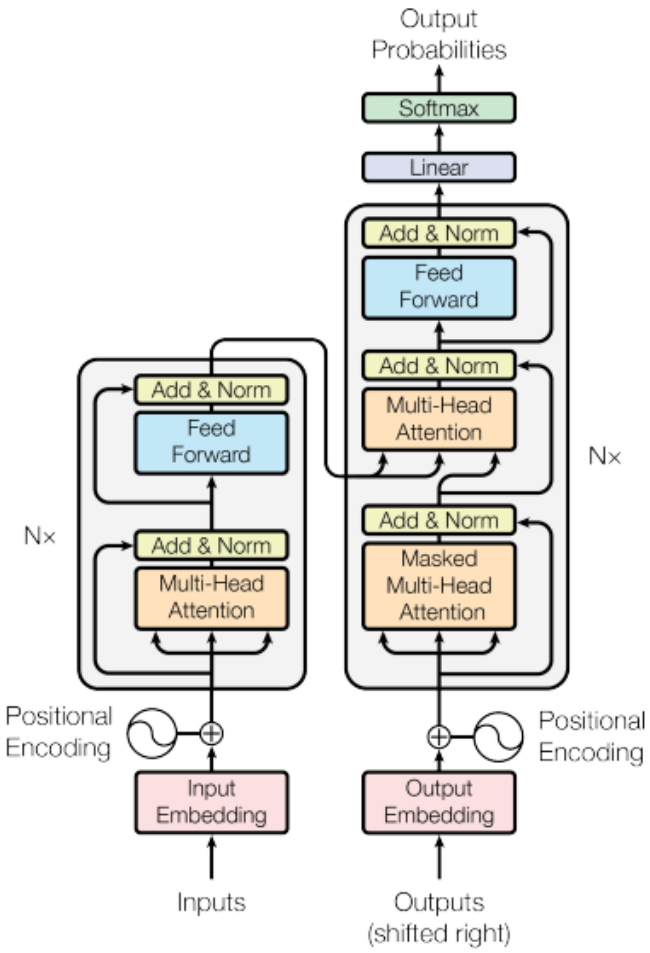


Figure 2.9: Transformer architecture [7]

Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers, or BERT, is a model based on the transformer architecture that has revolutionized the field of NLP. BERT can learn specific language tasks due to its training method, which allows to understand how words relate to each other. Fine-tuning BERT for various language tasks has shown to present impressive results. BERT only consist of encoders stack, making it a model that possesses a strong understanding of languages. it also includes a layer that embeds the input, and a final layer that processes the output to make sense of it. Thus, additional layers after the output could exist, like the Softmax function [10].
Figure 2.15 represents a simplified architecture of BERT.

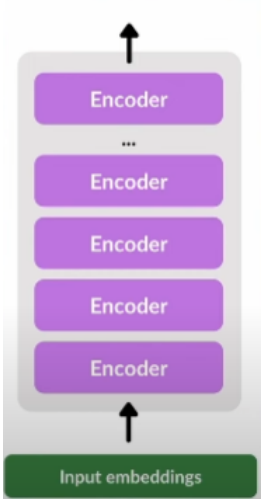


Figure 2.10: Simplified Architecture of BERT

How did BERT got trained and was able to understand the core of a language? Actually, BERT has been trained on two different tasks:

1. **Masked language modelling**: 15% of the words are masked or blanked out. The model's goal is to predict the words that should fill in these blanks [10].

2. **Next sentence prediction**: The model is given a group of sentences, and it must predict whether or not they belong together [10].

Training BERT on this two tasks has made it a highly performing language model. To customize BERT for other tasks, like for intent recognition, it needs to be fine-tuned. Two components are required for BERT's fine-tuning, a new output layer and data for the specific task. BERT's pre-training on large textual data, with its ability to fine-tune for specific tasks, which makes it highly customizable and suited for various natural language processing tasks, including intent recognition for virtual assistants [10].

2.4 Conclusion

In this chapter, Natural Language Processing and different Neural Network architectures have been presented. Next, Moving onto the "Requirements Specifications" chapter which will detail the application's requirements.

Chapter 3

Requirements Specifications

2.1 Introduction

This chapter will include a comprehensive overview of the application's requirements. Starting with a representation of the specific features the software must have including functional and non-functional requirements, a definition of the system's actors, the global use case diagram, detail the product backlog, and conclude with the technical specifications. This will present a fundamental step in ensuring that the final product aligns with the organization's vision.

2.2 Functional requirements

Functional requirements define how the system should behave, they provide a clear and precise description of what it must accomplish, they can be in the form of tasks, services or operations that must be carried out.
Down below are the description for each functional requirements:

- **Register:** The system allows the client to register in order to create a new account.
- **Verify account:** Upon registering, the client should verify their account through their email.
- **Authenticate:** The system allows the client to access their account by authenticating, providing their credentials that were set in the registration phase.
- **Check bank account's balance:** The systems enables the client to consult their bank account's balance.
- **Check history of operations:** The system lets the client check the history of their account's operations.
- **Check history of transfers:** The system enables the client to check the history of their transfers.
- **Check Card:** The system allows the client to check and activate/deactivate their card.
- **Transfer Money:** The system lets the client transfer money to a selected beneficiary.

- **Manage beneficiaries:** The system allows the client to manage the list of beneficiaries.
- **Change Settings:** The system lets the client customize the application's appearance.
- **Change password:** The system enables the client to change their password.
- **Check notifications:** The system enables the client to check their notifications.
- **Contact assistant:** The system enables the client to contact the bank's assistant.
- **Manage assistants:** The system permits Administrators to give access to assistants by managing their accounts.
- **Consult Dashboards:** The system grants the Assistants and administrators the consultation of the dashboard that gives insights about the application's evolution.
- **Manage Client Requests:** The system allows Assistants and administrators to manage support requests sent by clients.

2.3 Non-functional requirements

Non-functional requirements guide to identify the constraints that the system must meet to guarantee its seamless operations. The following represents the criteria that it needs to meet:

- **Accessibility:** The system must commit to the accessibility standards established by Web Content Accessibility Guidelines (WCAG) to ensure that the application can be fully utilized by everyone, particularly people with disabilities.
- **Performance:** The system has to respond gracefully to user commands, even under higher demands.
- **Maintainability and Scalability:** The application's code must be readable and well-organized, following the best conventions to facilitate scalability and maintainability.
- **Security:** User data safety and the confidentiality of personal data must be the primary focus of the system.
- **Error Handling:** The system must handle ambiguities with finesse through clear error messages to guide and assist users.

2.4 Identification of the actors

Actors are entities that interact with a system. Below, table 2.1, are the main actors of our application.

- **Client:** The actor who benefits from the various banking services offered by the application.
- **Administrator:** this actor has elevated privileges. They can control and manage access to assistants by granting them permissions.

- **Assistant:** The actor who checks the dashboard to help with decision making and orient the future of the application, and manage client requests.
- **Email System:** A system that sends emails for multiple purposes such as account verification.

Table 2.1: Detailed functionalities of the actors

Actor	Role
Client	Register Verify account Authenticate Check account's Balance Check history of operations Check history of transfers Transfer Money Manage beneficiaries Interact with virtual assistant Check Card Change Settings Change password Check notifications Contact Support
Administrator	Authenticate Manage Assistants Consult Dashboards Manage Client Requests
Assistant	Authenticate Consult Dashboards Manage Client Requests
Email System	send emails

2.4.1 Global use case diagram

The use case diagram is an essential step in building any software applications. it provides insights about the relationships between actors, use cases, and the outside system. it's a blueprint of a user's interactions with the system.

Figure 2.1 is an illustration of the global use case diagram that showcase different scenarios for different actors.

Figure 2.1: Global Use Case Diagram.

2.4.2 Product's Backlog

In this segment, the product backlog (Table 2.2), deduced from the specified requirements and the global use case diagram, will be presented. It's a list of prioritized tasks that will lead the development team in building the software within the estimated time frame.

Table 2.2: Product backlog

Backlog Item	Priority	Estimation	Planning
As a Client, I can register.	1	High	Sprint 1
As a (Client + Assistant + Admin), I can authenticate.	1	High	Sprint 1
As a Client, I can interact with the virtual Assistant.	2	High	Sprint 2
As a Client, I can check my bank account's balance.	3	Average	Sprint 3
As a Client, I can change settings.	3	Average	Sprint 3
As a Client, I can check the history of my operations.	3	Low	Sprint 3
As a Client, I can manage my card.	3	Average	Sprint 3
As a client, I manage beneficiaries.	3	Average	Sprint 3
As a client, I can transfer money to a selected beneficiary.	3	High	Sprint 3
As a client, I can check the history of my transfers.	3	Low	Sprint 3
As a client, I can change my password.	3	Low	Sprint 3
As a client, I can check my notifications.	4	Average	Sprint 4
As a client, I can contact bank's assistant.	4	Average	Sprint 4
As an Admin, I can manage assistants.	4	Average	Sprint 4
As a (Assistant + Admin), I can view dashboard.	4	Average	Sprint 4
As a (Assistant + Admin), I can manage clients request.	4	Average	Sprint 4

2.5 Technical requirements

This section will provide a detailed overview of the technical requirements, defining the tools and technologies that helped in the realization of the project.

In this section, software and technologies that were used to develop the applications will be presented.

Software Environment

Table 2.3, represents the sets of software that we have used.

Visual Studio Code: A popular integrated development environment developed by Microsoft. There are multiple distinctive features making it the go-to code editor: it handles numerous programming languages, provides powerful debugging tools, the experience can be customized and enhanced with extensions, and it's packed with Git for a better version control [11].

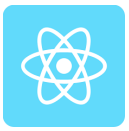
IntelliJ IDEA: A Java based code editor. It facilitates the development of Java projects, making it easy for beginners to start their journey. It offers a set of features like debugging tools that enhance productivity [12].

Postman: With its user-friendly interfaces, testing, documenting, sharing, and following the life-cycle of Application Programming Interfaces (APIs) becomes an easy and productive task for developers. Postman has become a crucial software for building full-stack applications over the years [13].

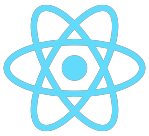
Draw.io: A free and easy-to-use software, accessible via web browsers, for building diagrams ranging from Unified Modeling Language (UML) diagrams, to flowcharts, and more. With its user-friendly interface, it's easy to share and collaborate with teams [14].

Front-end Development Tools

Table 2.4, represents the sets of front-end technologies have been used.



React Native: A mobile framework created by Facebook, popular for its principle "Learn once, write everywhere". This means the same code is compatible with various platform like iOS, Android and the web. This makes it productive and time-efficient. Under the hood, React Native uses React and JavaScript, making it easy to transition to mobile development. In addition, its large community also makes the learning curve more manageable [15].



ReactJS: A JavaScript library developed by Facebook for creating single-page applications. What makes React distinguishable is its component-based architecture, which facilitates re-usability, making it productive for creating complex front-end applications. React components are based on JSX, which allows the inclusion of HTML within JavaScript file [16].

Redux: A library for JavaScript applications used to manage application state. A state is a global object shared across the application, and its alteration can change the application's behavior. With the features offered by Redux, accessing and modifying the state is simple and efficient [17].

Material UI: Material UI is an open-source styling library for React that obeys Google's Material Design. With a collection of ready-to-use components. Material UI simplifies the building of user interfaces. Its commitment to Material Design standards reassures a high quality user experience [18].

React Native Paper: a Material Design library created with React Native. It offers a collection of ready-to-use components that provides a native look across Android and iOS platforms. With features like platform adaptation, and accessibility support like screen reader adaptability, React Native Paper help developers in creating mobile interfaces with ease [19].

Back-end Development Tools

Back-end development tools, table 2.5, are a set of software tools and frameworks issued to develop the serve-side of a software application by focusing on the logic that drives it.

Axios: a simple library for handling and managing API requests within JavaScript applications. It's efficient for communicating with the back-end through sending asynchronous requests and handling responses gracefully, whether it's an error or a success making easy to guide the user and notify them about errors [20].

Spring Boot: a Java framework that facilitates building Spring application with handling and simplifying the configuration process allowing for writing code directly without having to deal with XML configuration [21].

JSON Web token: or JWT, a library used to transfer information in a JSON object between two parties in a secured manner. JWT can be used for authentication and authorization [22].

PostgreSQL: a powerful, open source object-relational database system that has earned a strong reputation for reliability, feature robustness, and performance [23].

Docker: a tool for creating, managing, and running containers, acting as a transferable environment for applications and their dependencies. Docker supplies benefits like flexibility, lightweightiness, and portability, making it ideal for containerizing applications [24].

2.6 Conclusion

During this chapter, the requirement specifications including functional and non-functional, actors, the global use case diagram, the product backlog, and the technical requirement have been presented.
Next, the first sprint titled 'Authentication flow' will be presented, discussing its design and implementations process.

Chapter 4

Sprint 1 : Authentication flow

3.1 Introduction

This chapter introduces the first sprint titled 'Authentication flow' which describes the client's authentication process. It begins by outlining the backlog, refinement, modelling, and implementation of each use case. The aim is to create a seamless authentication process to ensure that the application is built on a secure base.

3.2 Sprint 1 Backlog Identification

Table 3.1 describes Sprint 1's backlog, highlighting the client stories that make up the first sprint.

Table 3.1: Sprint backlog

Backlog Item	Priority	Estimation	Planning
As a Client, I can Register.	1	High	Sprint 1
As a Client, I can verify my account.	1	Average	Sprint 1
As a (Client + Assistant + Admin), I can Authenticate.	1	High	Sprint 1
As a Client, I can reset password.	1	Average	Sprint 1

3.3 Sprint 1 refinement

In this sprint, the following use cases are the focal point:

- Register;
- Verify account;
- Authenticate;
- Recover password.

3.3.1 Refinement of the « Register » use case

Registering is the first step clients must do in order to gain access to the application's features. Afterwards, the client must verify their account by clicking on the link sent to

their email address. Clicking on the link will enable their account.

Figure 3.1 illustrates the refinement of the « Register » use case.

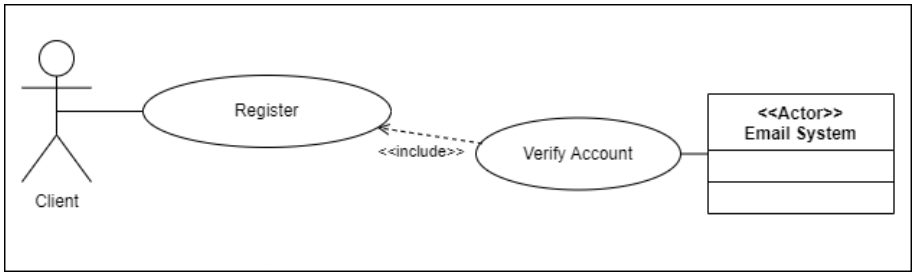


Figure 3.1: Refinement of the « Register » use case

Table 3.2 details the refinement of the « Register » use case:

Table 3.2: « Register » use case refinement.

Use case	Register
Actor(s)	Client, Email System
Pre-condition	System running
Post-condition	Registration established
Main scenario	- The system displays the registration screen. - The actor enters their personal information. - The actor clicks on the "Register" button. - The system verifies the data and sends an email verification. - The system displays the message of the success of sending the email.
Inclusion	- Email Validation
Exception	- The system displays an error message if the email format is incorrect. - The system displays an error message if the other data fields are missing or incorrect. - The system displays an error message if the email already exists. - The system displays an error message if the password isn't strong enough.

3.3.2 Refinement of the « Authenticate » use case

In order to control access, the client must authenticate providing their email and password that were set up during the registration process. Successful authentication

grants them access to the application's functionalities.
Assistants and Administrators can authenticate with their corporate email and password that were provided to them.

Figure 3.2 illustrates the refinement of the « Authenticate » use case.

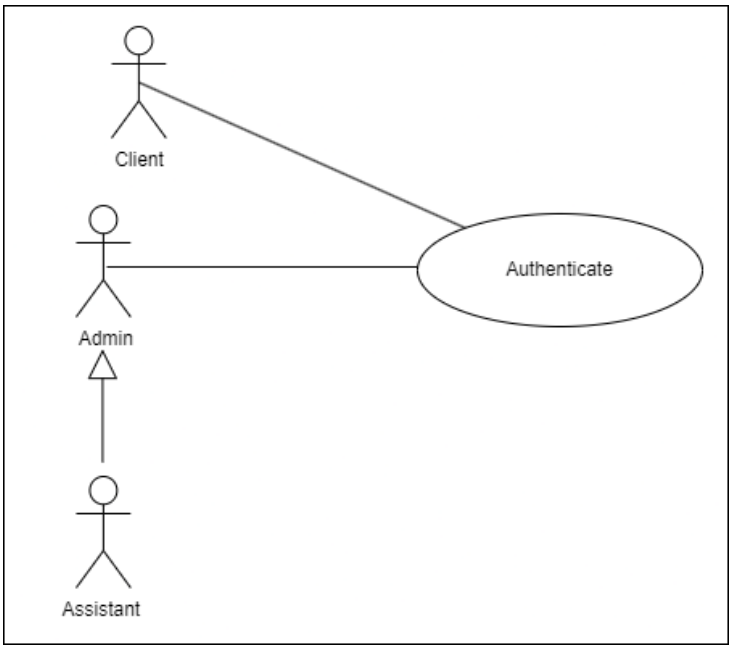


Figure 3.2: Refinement of the « Authenticate » use case

Table 3.3 presents the refinement of the « Authenticate » use case:

Table 3.3: « Authenticate » use case refinement.

Use case	Authenticate
Actor(s)	Clients, Assistants and Administrators
Pre-condition	System running
Post-condition	The client is authenticated
Main scenario	- The system displays the authentication screen. - The client enters their email and password. - The client clicks on the "Authenticate" button. - The system verifies the email and password, and create a token for the client's session if the data is correct. - The system displays the home page.
Exception	- The system displays an error message if the email doesn't exist or not validated. - The system displays an error message if the combination of email and password are incorrect.

3.3.3 Refinement of the « Recover Password » use case

In case of loss of credentials, clients can recover their password in order to regain access to their account.

Figure 3.3 illustrates the refinement of the « Recover Password » use case.

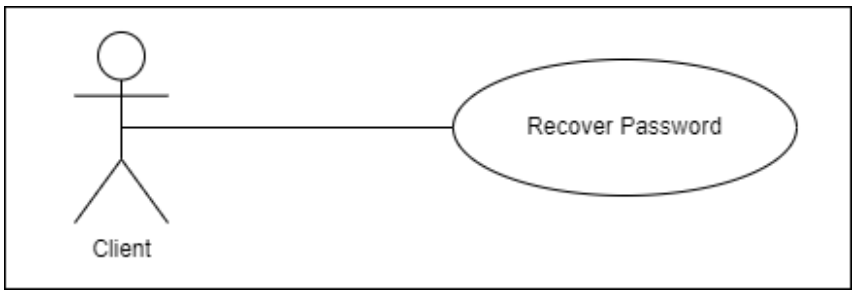


Figure 3.3: Refinement of the « Recover Password » use case

The following Table 3.4, presents the refinement of the « Recover Password » use case:

Table 3.4: « Recover Password » use case refinement.

Use case	Recover Password
Actor(s)	Client
Pre-condition	System running
Post-condition	The password updated successfully.
Main scenario	- The system displays the recover password interface. - The client enters their data. - The actor clicks on the "Send" button. - The system verifies the provided information. - The system sends a password reset link to the provided email. - The client is directed to a page where they can reset their password. - The system securely updates the client's new password. - The system redirects the client to Authenticate page.
Exception	- The system displays an error message if the information are wrong. - The system displays an error message if the email doesn't exist.

3.4 Sprint 1 Modelling

Modelling serves as a blueprint when it comes to building a software application. It shed's light on the system's structure and functionalities offering a guidance and clear road-map to the development team. Additionally, guarantee that the final product aligns with the requirement specified. In this section, we focus on the design of each use case in Sprint 1.

3.4.1 « Register » Use Case Modelling

Class Diagram

The class diagram serves to clarify the static structure of the application. It traces the interfaces, controllers, and entities to give overview of the system. Figure 3.4 illustrates the class diagram for the « Register » use case.

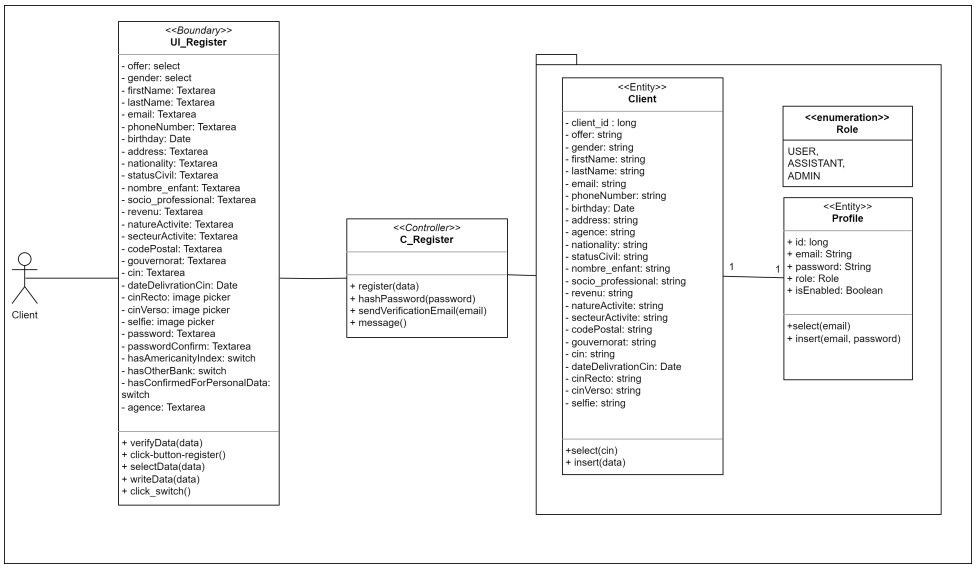


Figure 3.4: Class Diagram for the « Register » use case

Upon registering, the client must verify their account through clicking on a link sent to their email. Clicking on this link, the 'isEnabled' attribute is set to 'true' if the link is still valid. Figure 3.5 represents the class diagram for the « Validate Email » use case

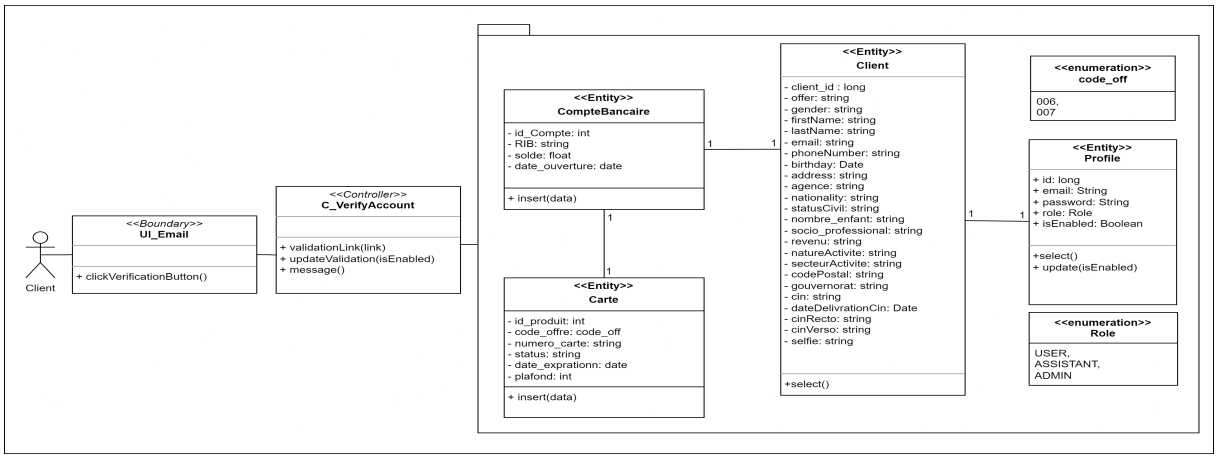


Figure 3.5: Class Diagram for the « Validate Email » use case

Sequence Diagrams

In this section, the sequence diagrams for the « Register » and « Validate Email » are presented.

Sequence Diagram for the « Register » Use Case

Upon launching the application, an authentication screen will be displayed. Clients without an account can select the 'I don't have an account' option, which will then bring up the registration interface. They should fill in all the required fields and then select the 'Register' button. If the information are correct and the email is not already in use, a verification email will be sent and a success message will pop up saying 'Account has been created, Check your Email!' otherwise it will say 'This Email is already used!' (Figure 3.6).

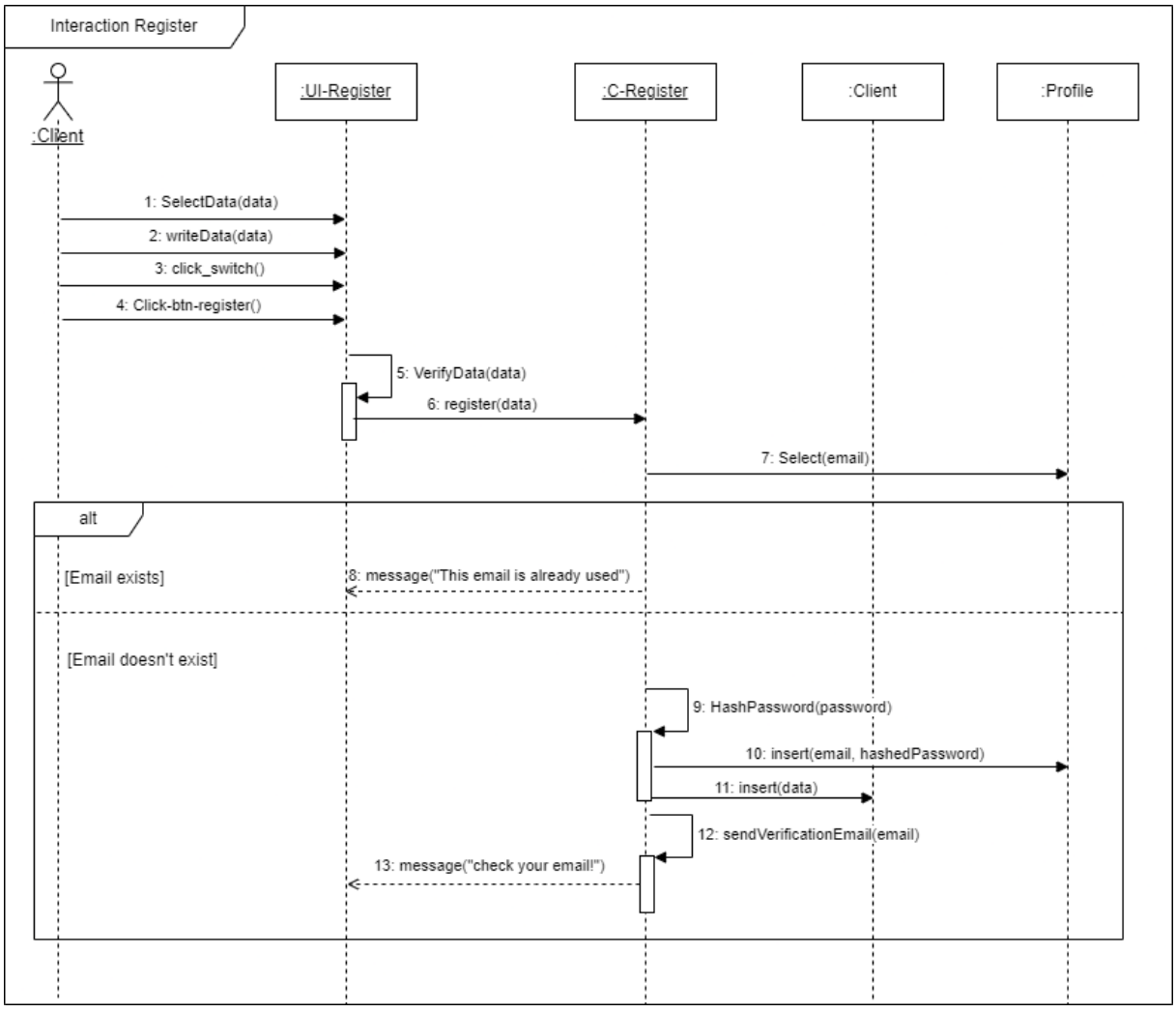


Figure 3.6: Sequence Diagram for the « Register » Use Case.

Sequence Diagram for the « Validate Email » Use Case

Upon registration, a verification email will be sent. The client can then click on the button within this email. The server checks the validity of the link using a token. If the token is valid, the system extracts the email from the token and verifies if the client

exists. If the client exists, a success message stating 'Your account has been verified!' is sent, otherwise, 'the link has expired!' (Figure 3.7).

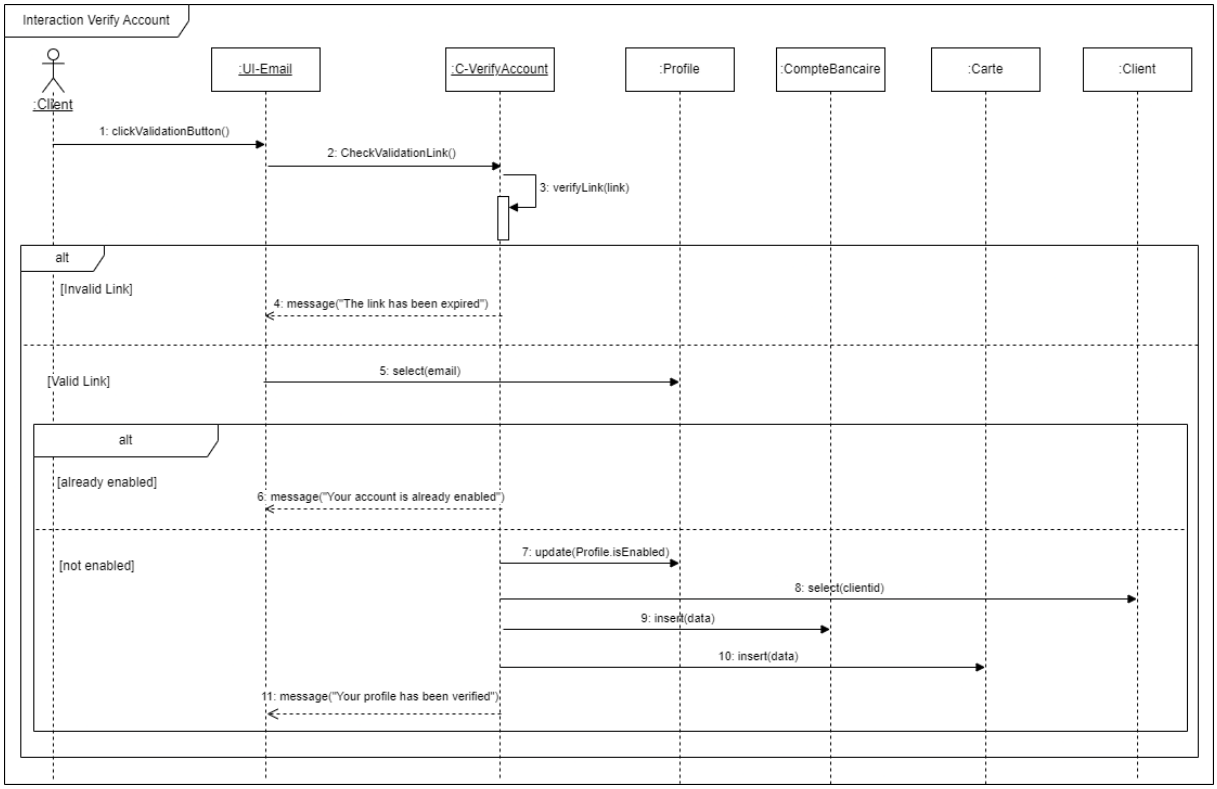


Figure 3.7: Sequence Diagram for the « Validate Email » Use Case.

3.4.2 « Authenticate » Use Case Modelling

During section the class diagram and sequence diagram for the « Authenticate » use case are presented.

Class Diagram

Figure 3.8 represents the class diagram related to the « Authenticate » use case:

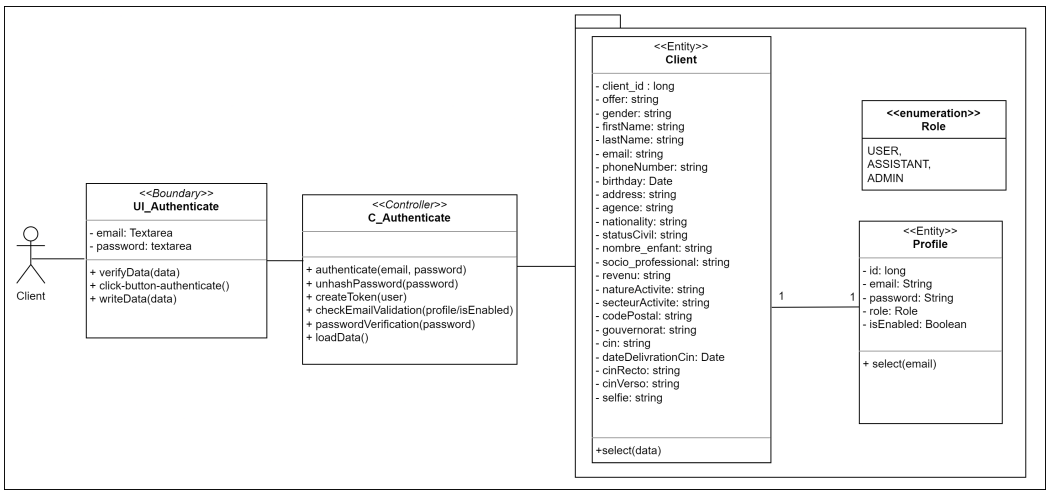


Figure 3.8: Class Diagram for the « Authenticate » use case

Sequence Diagram for the « Authenticate » Use Case

When the application is launched, an authentication screen appears. Clients must enter their email addresses and passwords in order to access to their account. These credentials will be verified on the client side and then upon clicking the 'authenticate' button, they will be sent to the server. The authentication service will validate the data for accuracy, checks if the password matches and whether the email exists and has been verified. If conditions are met, the system will redirect the client to the home screen, otherwise it will display an error message (Figure 3.9).
For administrators and assistants, they go through the same authentication process as the clients. However, in this case, only the profile table is needed, as there is no need to hold extra information. The token holds the email, role, and ID and stored in a Http Only cookie to grant a secure session.

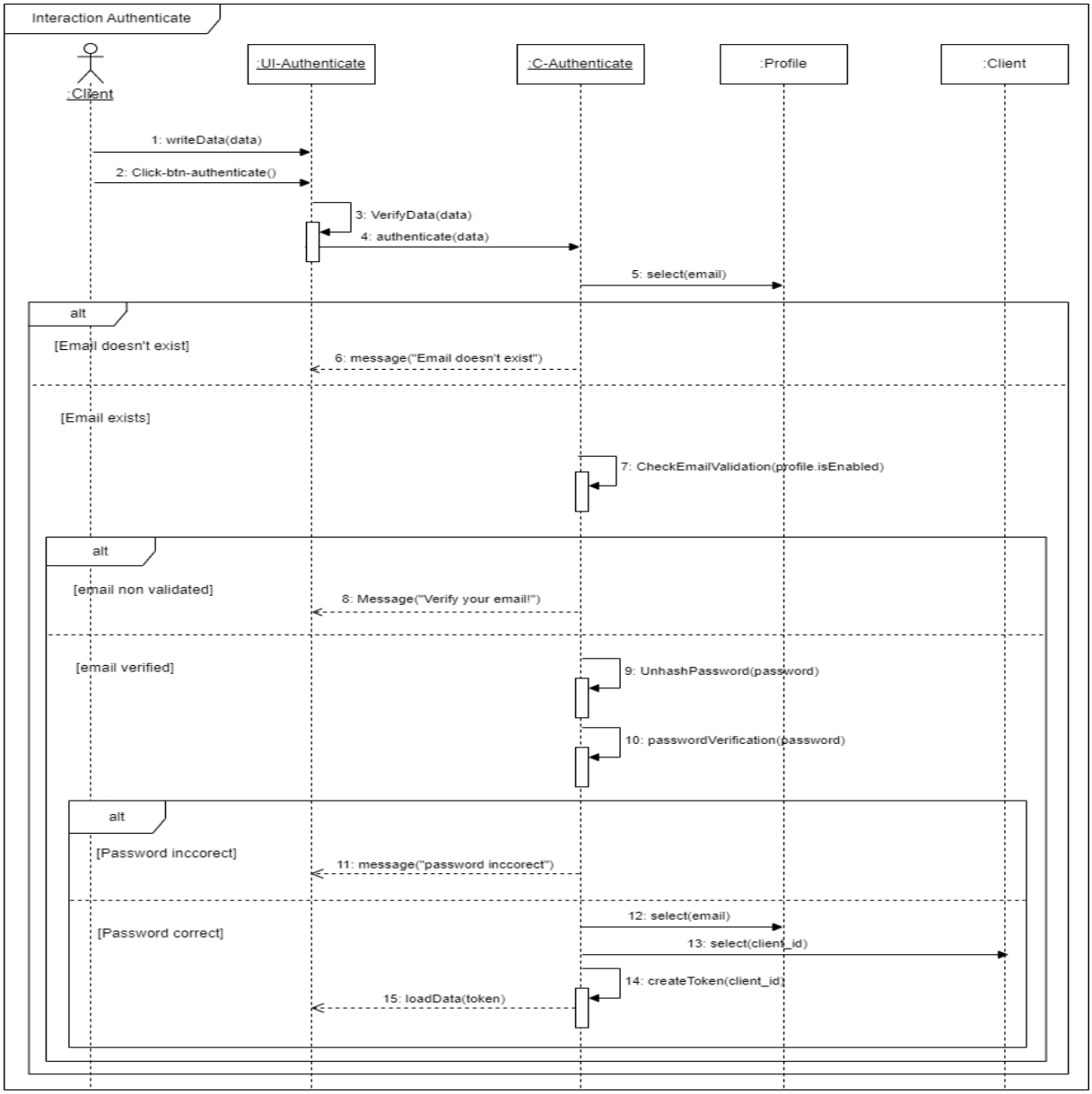


Figure 3.9: Sequence Diagram for the « Authenticate » Use Case

3.4.3 « Recover Password » Use Case Modelling

In this section the class diagram and sequence diagram for the « Recover Password » use case are presented.

Class Diagram

Figure 3.10 represents the class diagram related to the « Recover Password » use case:

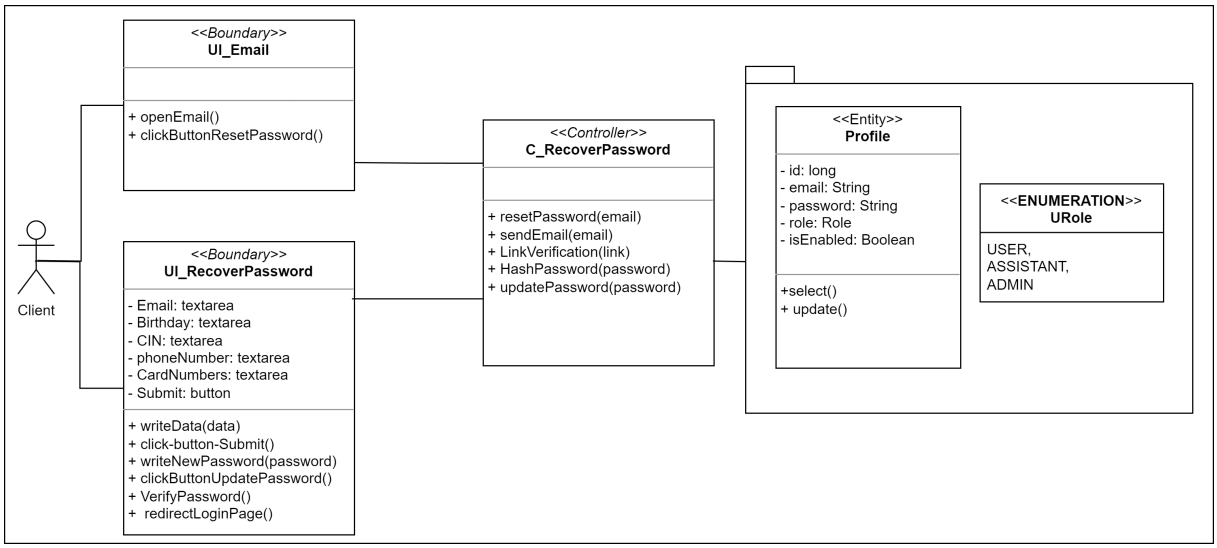


Figure 3.10: Class Diagram for the « Recover Password » use case

Sequence Diagram for the « Recover Password » Use Case

To recover their password, the client can access the authentication interface and click on 'Forgot your password?' to display the Password Recovery interface. The client must enter the required information and click the 'Submit' button for the system to verify the client's existence. Otherwise, it will send an error message, 'Email does not exist!'

If the email is correct and does exist, the system will send an email that expires after a set amount of time. Then, it will display a success message on the Password Recovery interface, 'Check your email!'. Afterwards, the client must click on the 'Reset your password' button that's found in the email, which upon clicking, will display the New Password interface.

Finally, the client must enter their new password and confirm it, then click on the 'Reset' button for the system to change the password in the database after hashing (Figure 3.11).

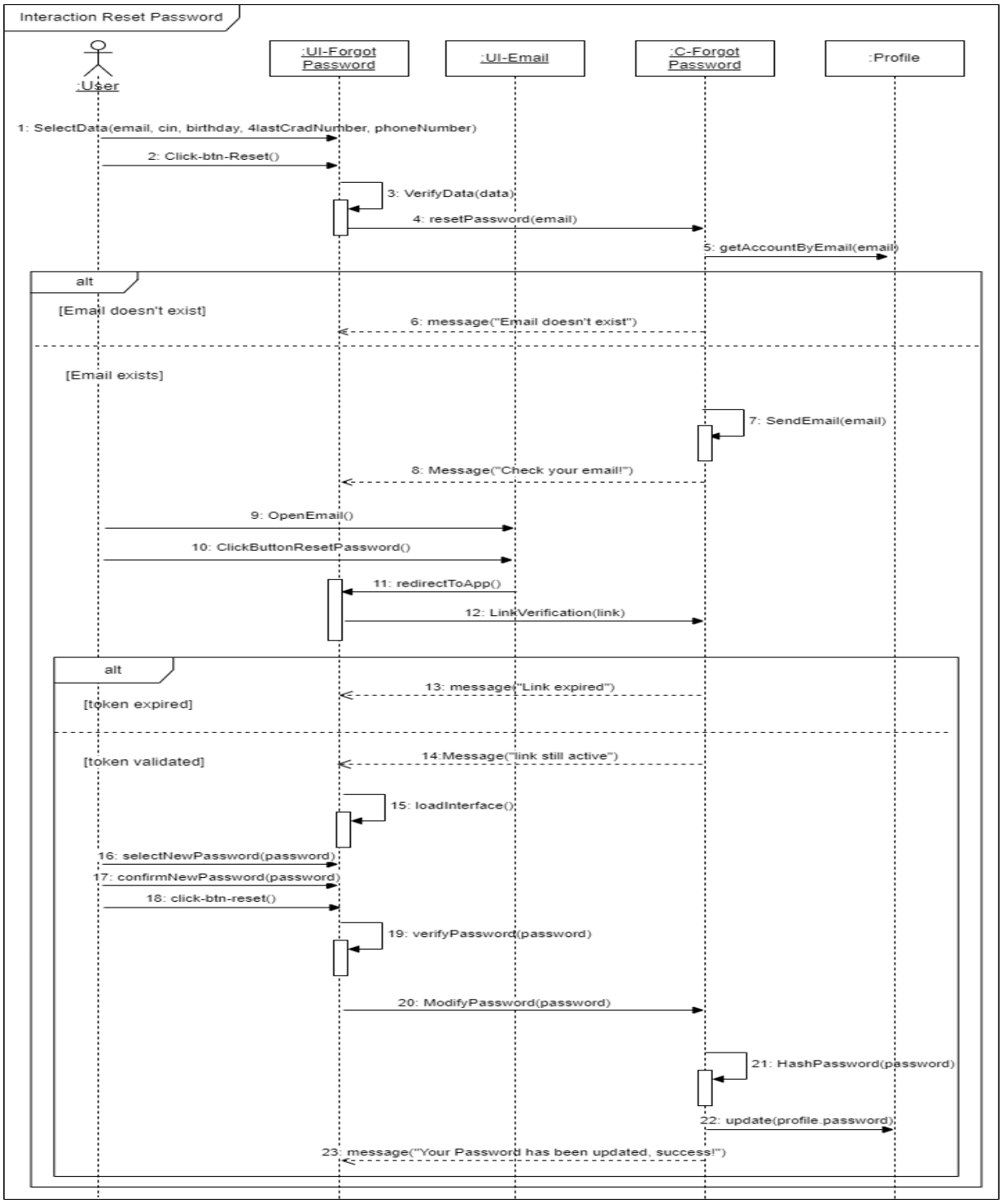


Figure 3.11: Sequence Diagram for the « Recover Password » Use Case

3.5 Sprint 1 Class Diagram

Figure 3.12 is a representation of the Sprint 1 class diagram. It visualizes how the 'Client' class has a One-To-One relationship with other classes. Which means each 'Client' is linked to one instance of another class.

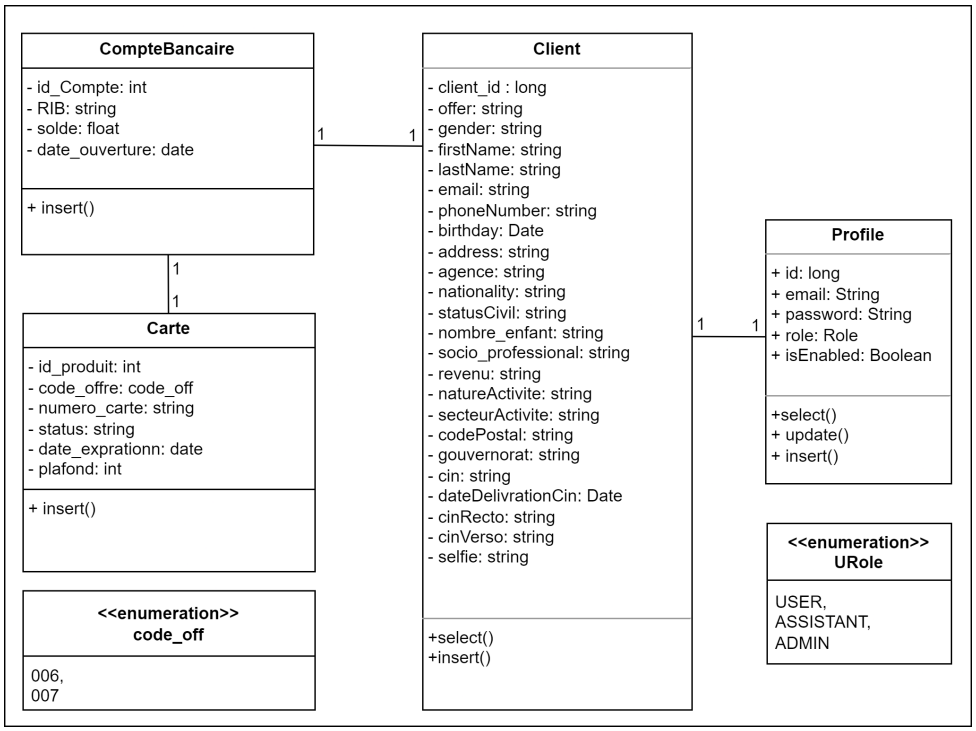


Figure 3.12: Sprint 1 Class Diagram

3.6 Deployment Diagram

Figure 3.13 represents the deployment diagram of the application. It visualizes how different physical devices, represented in nodes, communicate with each other. The communication is made possible through HTTP request. Spring Boot is being the center of our application, handling requests that are coming from different nodes and responsible for the read and write operations to the database.

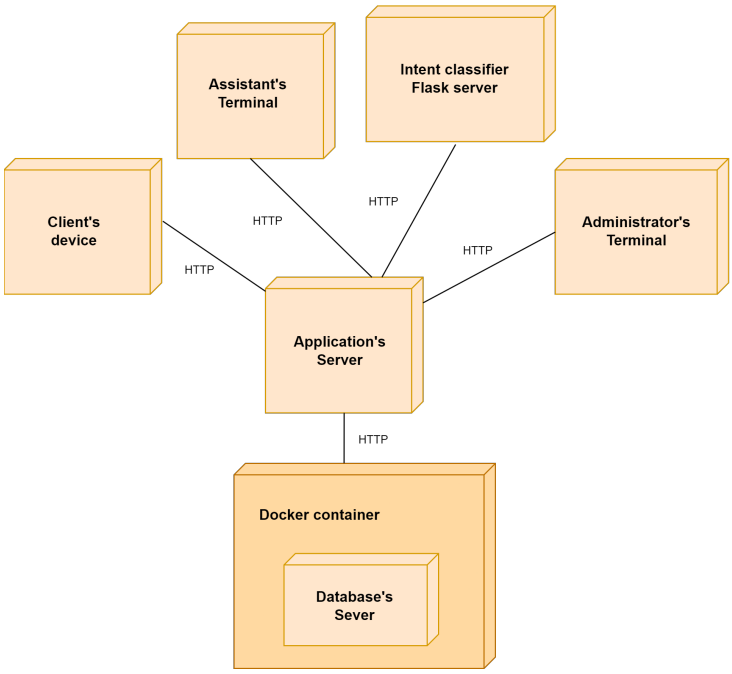


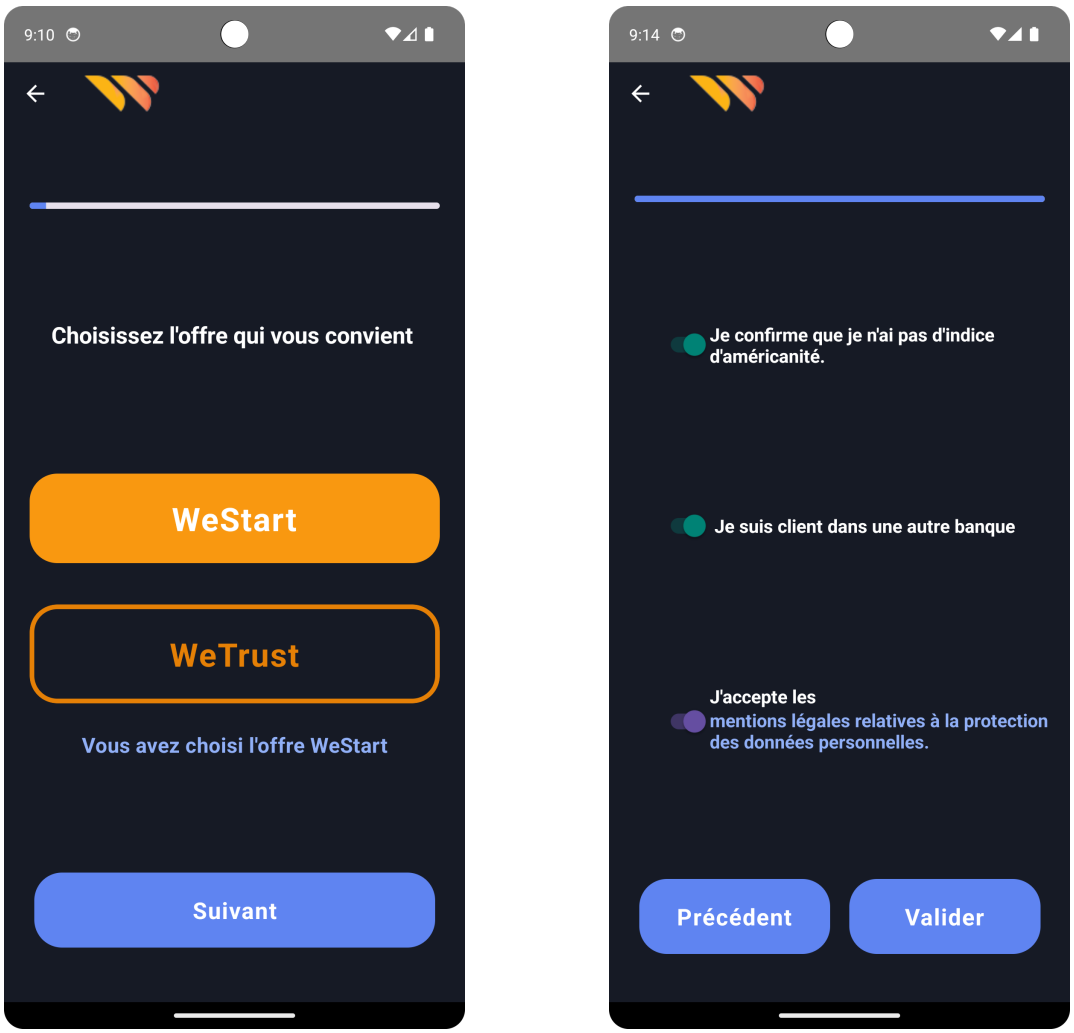
Figure 3.13: Deployment Diagram

3.7 Realization

In this section, we will present the implementation of the sprint 1 features.

3.7.1 Realisation of « Register » Use Case

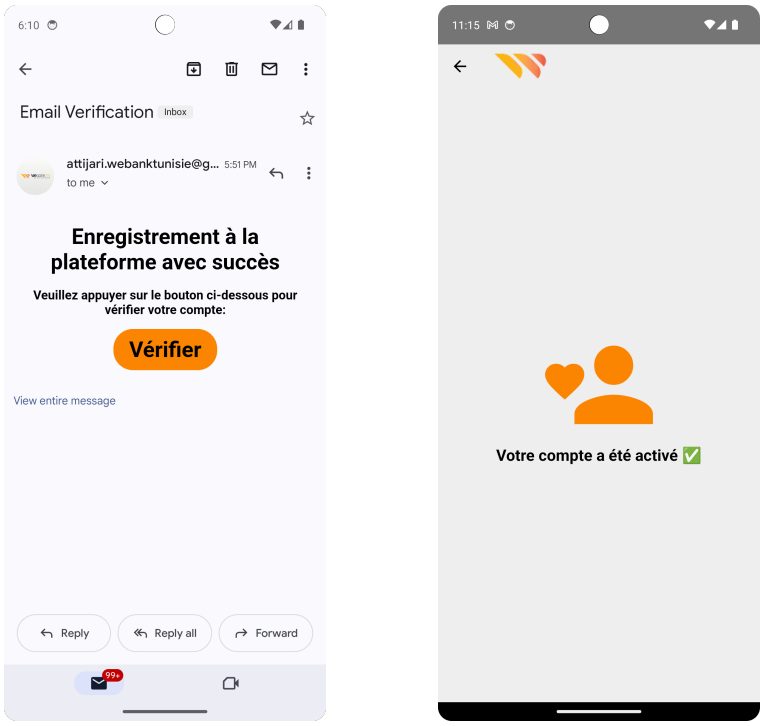
Once a client opens the app, they can access a registration interface.
To simplify the registration process and enhance accessibility, only one input field is displayed at a time. This ensures that all required fields are entered and verified, making the process more fluid, especially for people with disabilities.
Figure 3.14 illustrates the parts of the registration process, which starts by filling out the multi-step form then clicking on register.



(a) First Step (b) Last Step

Figure 3.14: Multi-Step Registration

Figure 3.15 demonstrates how a client would verify their account upon completing the registration phase successfully.

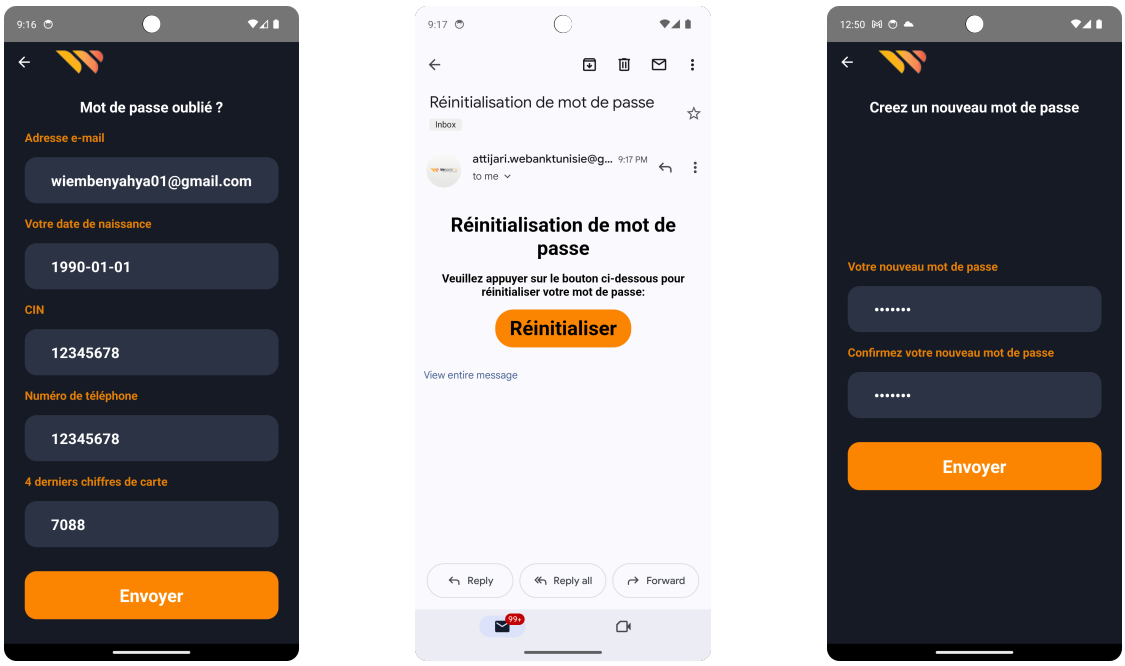


(a) Email Verification (b) Verification Interface

Figure 3.15: Account Verification

3.7.2 Realisation of « Recover Password » Use Case

In case of forgetting the account password, the client can access a the 'Forgot password' interface, fill out the form and they will get an email that will allow them to change their password as described in figure 3.16.



(a) Forgot Password Interface (b) Forgot Password Email (c) New Password Interface

Figure 3.16: Forgot Password Process

3.7.3 Realisation of « Authenticate » Use Case

After validating their email, the client can access an authentication interface that appears as described in figure 3.17.

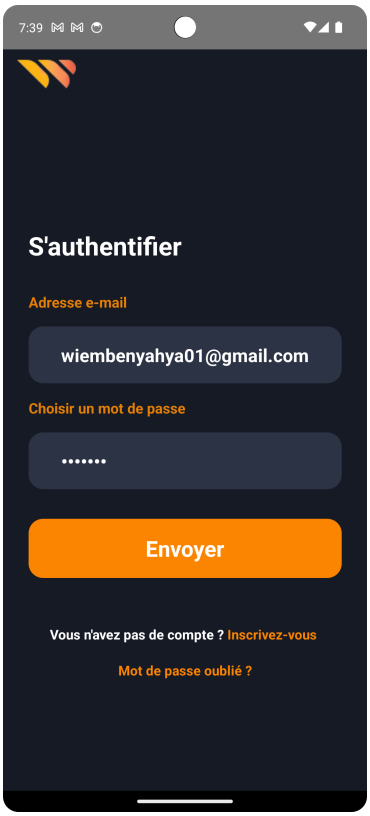


Figure 3.17: Authenticate Interface

3.8 Conclusion

A presentation of the refined first sprint backlog was presented in this chapter, as well as the use case diagrams, their relative class and sequence diagrams and a showcase of the realization of this sprint. Next, Sprint 2, will outline the development of the application's virtual assistant.

Chapter 5

Sprint 2: Virtual Assistant

5.1 Introduction

A virtual assistant is an intelligent conversational software that handles users' speech or text commands, providing answers to questions and execute tasks. What lies under the hood of a virtual assistant is an intent classification system. Sprint 2 dive into the creation of a virtual assistant customized for banking services.

5.2 Libraries used

The following section will focus on the libraries that have been used for building the conversational AI.

NumPy: short for "Numerical Python", an efficient solution for dealing with arrays in Python. It's faster than traditional Python arrays, making it suitable for handling machine learning tasks. What differentiates NumPy is its approach placing the data in continuous places in memory [25].

Pandas: a flexible and easy-to-use Python library for data analysis. It offers a statistical overview over a given dataset. Pandas is crucial in building machine learning models as it reassures the quality of the dataset [2].

Transformers: developed by the Hugging Face, a library that provides quick-to-use APIs for tasks like as tokenization. Transformers are primarily used for Natural Language Processing tasks such as entity recognition, translation, and text classification [26].

TensorFlow: its aim is to simplify the process of building simple to complex network models by providing ready-to-use APIs. TensorFlow is highly flexible and can be used with various data types, including but not limited to developing models with speech recognition, image classification, and text recognition [27].

Keras: offers a user friendly interface for constructing, training, and deploying neural network models. Its APIs conceal the complexity of building such models, making it easy, time efficient and beginner-friendly as it reduces the necessity of writing code and having deep knowledge. Keras is an excellent choice for machine learning beginners. Keras integrates with TensorFlow, Pytorch and JAX making it highly flexible [28].

5.3 Machine learning life cycle

Building a machine learning model consist of three cyclic phases: data preparation, model development, and deployment.

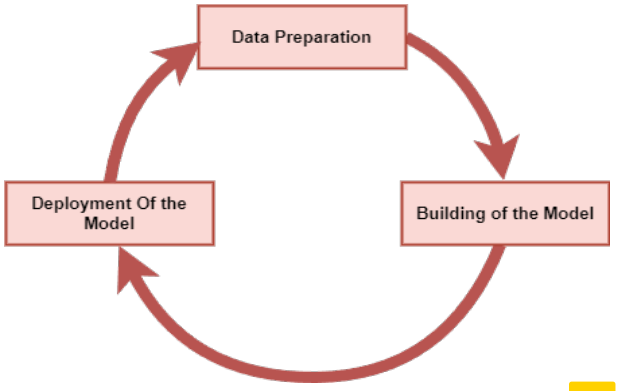


Figure 5.18: Machine learning life cycle

Every enhancement on the dataset will provoke the cycle to repeat itself and lead to the generation of a new and different model, adapted to the upgrades.

5.3.1 Data preparation

In this section, a description of the approach of generating a custom dataset will be provided. Additionally, the process of tokenizing the dataset, which prepares for feeding it into a machine learning model.

a. **Dataset generation**
Looking for a ready-to-use dataset to train the model with was quite a hustle. It was difficult to find a customized dataset to the banking services. So the decision to generate a custom one has been made, this reassures that the dataset is not limited and can be further improved depending on the bank's set of services.
An approach called 'Mad libs' has been opted for generating the dataset. It is a children's game, where players fill in the blanks with the right word. By adopting this method, a dataset was generated from a collection of 'templates'. Figure below clarifies an example of creating template for 'virement', transfer money, intent.

virement.intent M ×

dataset > raw > intents > virement.intent

```
1  {prepend_request} Effectuer un virement de {amount_of_money} au {person}
2  {prepend_request} transférer {amount_of_money} au compte de {person}
3  {prepend_request} faire un virement de {amount_of_money} à {person}
4  {prepend_request} réaliser un virement de {amount_of_money} au compte de {person}
5  {prepend_request} effectuer un virement de {amount_of_money} vers le compte de {person}
6  {prepend_request} transférer {amount_of_money} au nom de {person}
7  {prepend_request} faire un virement de {amount_of_money} à destination de {person}
8  {prepend_request} possible de réaliser un virement de {amount_of_money} vers le compte de {person}
9  {prepend_request} transférer {amount_of_money} à destination de {person}
```

Figure 5.19: Template of 'virement' intent

A typical transfer money request holds the 'amount of money' to transfer and the name of the person, in this case, the beneficiary. 'Prepend request' is some formalities of demanding a service like "would you please' and so on. These are called entities, they can take different values but they fall into the same category. Figure 5.20 illustrates the example of 'prepend-request' entity.

prepend_request.entity M ×

dataset > raw > entities > prepend_request.entity

```
1
2  Pouvez-vous
3  Voudrez-vous s'il vous plaît
4  Ce serait génial si vous pouviez
5  Pourriez-vous s'il vous plaît
6  Ce serait bien si vous pouviez s'il vous plaît
7  Ne pourriez-vous pas simplement
8  Je tiens vraiment à ce que vous
9  Pour l'amour du ciel, juste
10 S'il vous plaît
11 Je me demandais si
12 Je veux
13 J'aimerais
14
```

Figure 5.20: Template of prepend request entity

This approach has resulted of approximately eleven thousand of unique samples. The bar chart below, figure 5.21 explains the distribution of banking service intents: transfer money (virement), check balance (consulter solde), check bank's account mouvement (consulter mouvements), and check the history of transfers (consulter historique des virements). the height of each bar represents the number of each intent. As it's shown, each intent has around an equal number of samples, resulting in a balanced dataset. This ensures that the model has an equal chance to learn from all the intents, and leading for a reliable evaluation metrics.

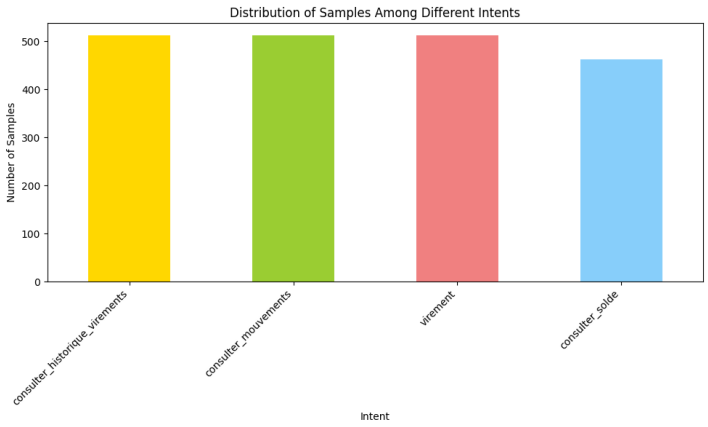


Figure 5.21: Distribution of Different Intents

b. **Tokenization**
The step of tokenization is crucial in NLP tasks. It transfers textual data into numerical format that machines can understand. When loading a tokenizer, it's important to use the same one used during the training of the pre-trained model. In this case, 'distilbert-base-uncased' was the pre-trained model.

5.3.2 Model development

a. **Model architecture**
Keras has been used to build the model architecture, which involves identifying the layers. These layers are the units that define a neural network model. They are stacked together so that each layer processes its inputs, performs computations, and passes the output to the next layer. There are three main layers:

- **Input layers:** the DistilBERT-based model accepts two input layers that were generated in the tokenization phase. Input ids which represent the tokenized sequences, and the input attention layer which guides the model to focus on the important tokens. Both with the size of 128, which corresponds of the maximum size of a sequence.
- **Transformer layer:** a layer that sits between the input and output layers. Responsible for understanding input sequences. It's referred to as the hidden layer because its work isn't observable. It holds the output of the transformer layer which is contextualized embeddings of the input tokens.
- **Output layers:** We have two output layers. The first is the intent output layer, which predicts the intent of the text. The second consists of entity output layer, which predict the entities in a given sequence.

The figure below provides an understanding of the model architecture. It visualizes the flow of data, from input to output, and highlights layers involved in the computation.

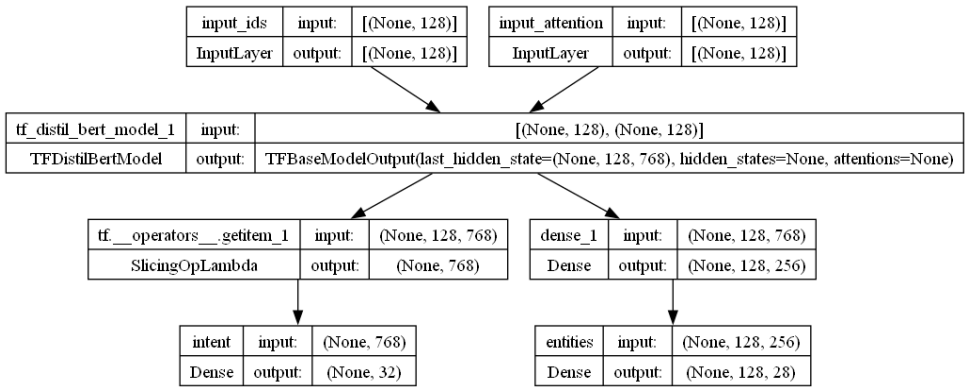


Figure 5.22: Model Architecture Layers

b. **Model Evaluation**
Precision, recall, and categorical accuracy have been used to evaluate our model.

- **Precision:** the proportion of instances that were correctly predicted as positive among all the instances that were classified as positive.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall:** Recall, or sensitivity, is the ability of the model to detect positives correctly.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **Categorical Accuracy:** a metric that measures the proportion of correctly classified instances among all of the other instances in the dataset across all classes.

$$\text{Categorical Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

The table 5.6 shows the results of the evaluation metrics of our model after ~~many trials of~~ fine-tuning ~~and changing hyper-parameters~~.

	Loss	Categorical Accuracy	Precision	Recall
Entities	0.065	0.98	0.99	0.98
Intent	0.18	0.95	0.94	0.94

Table 5.6: Evaluation Results

5.3.3 Model Deployment

Deployment of a machine learning model consist of making the model accessible to the application and grant interaction to get predictions at any time needed. Deploying our model in a web service is the solution that has been opted, making it available via REST APIs. This is done by creating an endpoint, a URL, that defines the location or port of the web service. Flask, figure 5.23, has been the web framework chosen, a framework that simplifies the creation of web services in python.

Figure 5.23: Flask's logo

The following explain the scenario of how our system integrates with the model:

1. Client issues his command, which could range from just consulting his balance to transferring money to one of his beneficiaries. The command is transcribed using a speech-to-text service, facilitated by the speech recognition that was used in React Native. The transcribed command is sent to Spring Boot.

2. Spring Boot sends the prompt to Flask.

3. Spring Boot receives the intent and entity predicted by the model.

4. Spring Boot will execute the task depending on the prediction.

5. Spring Boot sends a string message to React Native. A text-to-speech service will voice the response.

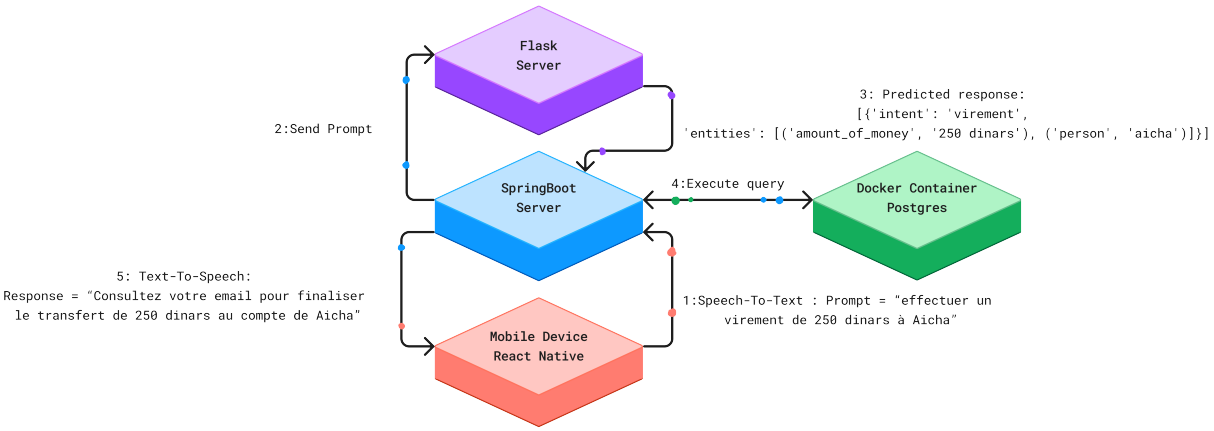


Figure 5.24: Virtual Assistant Integration

5.4 Conclusion

During this chapter, the process that went into the creation of an intent recognition system has been explained, starting from defining the libraries used and outlining the machine learning life cycle. Next chapter will be the implementation of sprint 3, titled 'Operations and Transfers'.

Chapter 6

Sprint 3: Operations and Transfers

6.1 Introduction

This chapter serves as an introduce to the work flow of sprint 3, titled 'Operations and Transfers'. During this sprint, the banking set of services are refined, conceptualized, and realized. The goal will be to establish a fluid user interaction with different banking features integrating accessibility and interface customization to better assist those with disabilities.

6.2 Sprint 3 Backlog Identification

Table 6.1 provides a broad perspective of different tasks that shape sprint 3.

Table 6.1: Sprint backlog

Backlog Item	Priority	Estimation	Planning
As a Client, I can check my bank account's balance.	3	Average	Sprint 3
As a Client, I can change settings.	3	Average	Sprint 3
As a Client, I can check the history of my operations.	3	Low	Sprint 3
As a Client, I can manage my card.	3	Average	Sprint 3
As a client, I can manage beneficiaries.	3	Average	Sprint 3
As a client, I can transfer money to a selected beneficiary.	3	High	Sprint 3
As a client, I can check the history of my transfers.	3	Low	Sprint 3
As a client, I can change my password.	3	Low	Sprint 3

6.3 Sprint 3 Refinement

Herein, a comprehensive and refined overview of the following scenarios are being introduced:

- Check bank account's balance;
- Check history of operations;
- Check history of transfers;
- Manage beneficiaries;
- Transfer money;

- Manage card;
- Change settings;
- Change password.

6.3.1 Refinement of the « Check account's Balance » Use Case

The act of checking the bank account's balance is fundamental in any banking application. It's the first thing observed when a client is redirected to the home page.
The refinement of the « Check account's Balance » use case is portrayed in figure 6.1.

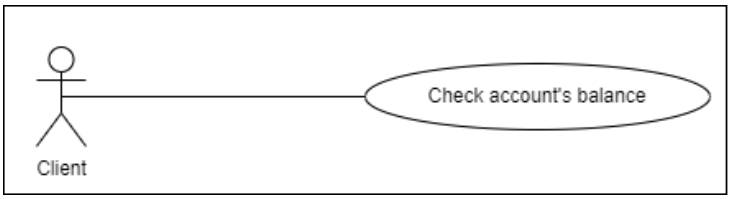


Figure 6.1: Refinement of « Check account's Balance » use case

Table 6.2 defines the client's interaction with this feature. It traces both the pre and post conditions, the main scenario of this use case, and potential errors that may happen.

Table 6.2: « Check account's Balance » use case refinement.

Use case	Check account's Balance
Actor(s)	Client
Pre-condition	Client is authenticated
Post-condition	Account's Balance checked
Main scenario	- The system displays the home screen. - The client checks their account's available balance.
Exception	- The system displays an error message if the balance is not identified for the client. - The system logs the user out if the token is expired.

6.3.2 Refinement of the « Check history of Operations » Use Case

Checking the history of operations is another fundamental action in a banking application. It offers clarity to the client about diverse operations that happened in their bank account. An operation could include actions like depositing money through an ATM or making transfers.
The refinement of the « Check history of Operations » use case is provided in figure 6.2.

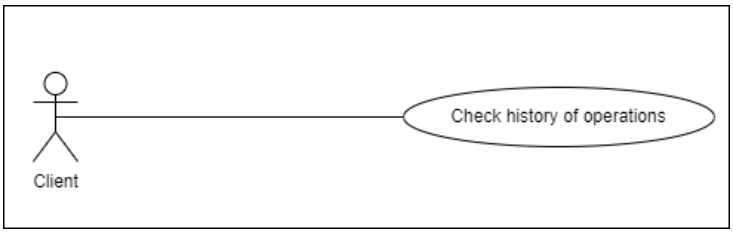


Figure 6.2: Refinement of « Check history of Operations » use case

Table 6.3 provides a detail scenario of the client's interaction with the « Check history of Operations » use case.

Table 6.3: « Check history of Operations » use case refinement.

Use case	Check history of Operations
Actor(s)	Client
Pre-condition	Client is authenticated
Post-condition	History of operations checked
Main scenario	- The system retrieves all operation stored in the databases. - The system displays the interface. - The client consults their account's operations. - The client can filter operations by date.
Exception	- The system displays an error if the server is down. - The system displays an error message if the start date exceeds the end date.

6.3.3 Refinement of the « Manage Card » Use Case

Managing the debit card allows the client to hold control of it remotely. With a touch of a button, they can deactivate and activate their card. In addition, important information is displayed.
The refinement of the « Manage Card » use case is shown in figure 6.3.

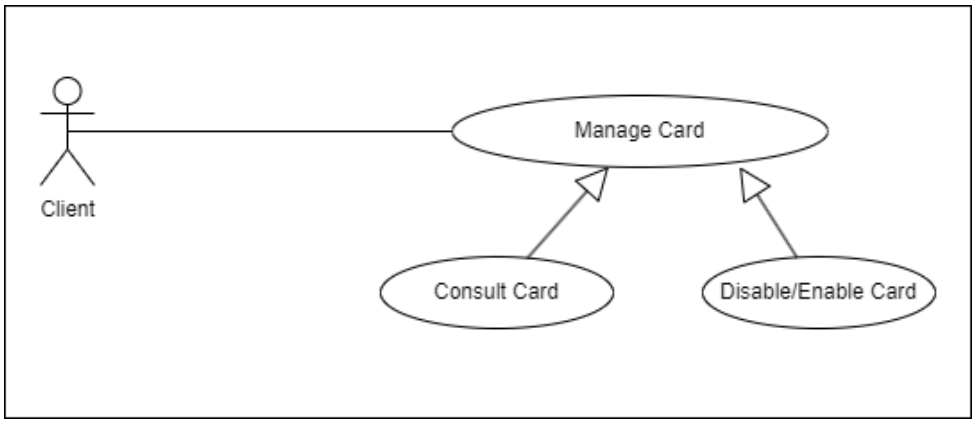


Figure 6.3: Refinement of « Manage Card » use case

Table 6.4 dive into a detailed overview of « Manage Card » scenario.

Table 6.4: « Manage Card » use case refinement.

Use case	Manage Card
Actor(s)	Client
Pre-condition	Client is authenticated
Post-condition	Card Managed
Main scenario	- The system retrieves and displays card information. - The client manages their card information.
Scenarii	- Enable/Disable card. - Consult card.
Exception	- The system displays a loading state if an error occurred while displaying the card's information.

6.3.4 Refinement of the « Check history of transfers » Use Case

Checking history of transfers consist of displaying a list of previous transfers made by the client. This reveals the beneficiaries who have received money from the benefactor, in this case, the client, along with the date of the operation. The refinement of the "Check history of transfers" use case is shown in figure 6.4.

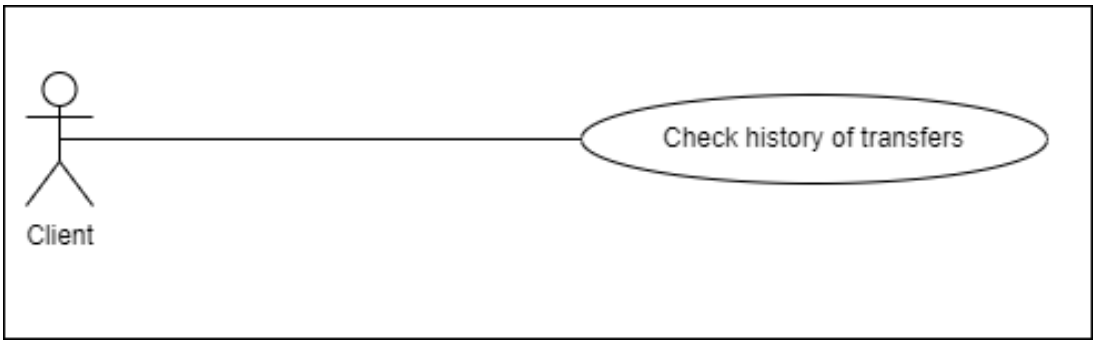


Figure 6.4: Refinement of « Check history of transfers » use case

Table 6.5 outlines a detailed and comprehensive overview of the « Check history of transfers » use case. It specifies the main actor, pre and post conditions of this feature, the main scenario, and illustrates errors that may occur if things went wrong.

Table 6.5: « Check history of transfers » use case refinement.

Use case	Check history of transfers
Actor(s)	Client
Pre-condition	Client authenticated
Post-condition	Transfers are checked
Main scenario	- The system retrieves and displays the history of transfers. - The client checks their history of transfers they made. - The client can filter their transfers by date.
Exception	- The system displays an error if the server is down. - The system displays an error message if the start date exceeds the end date.

6.3.5 Refinement of the « Manage Beneficiaries » Use Case

Beneficiaries are the client's list of individuals who receive money when a transfer occurs. This helps the client to associate a name to the Statement of Banking Identity (Relevé d'Identité Bancaire, RIB), rather than dealing with a long string of numbers. This way makes it easier to retrieve beneficiaries by their associated names.
The refinement of the « Manage Beneficiaries » use case is illustrated in figure 6.5.

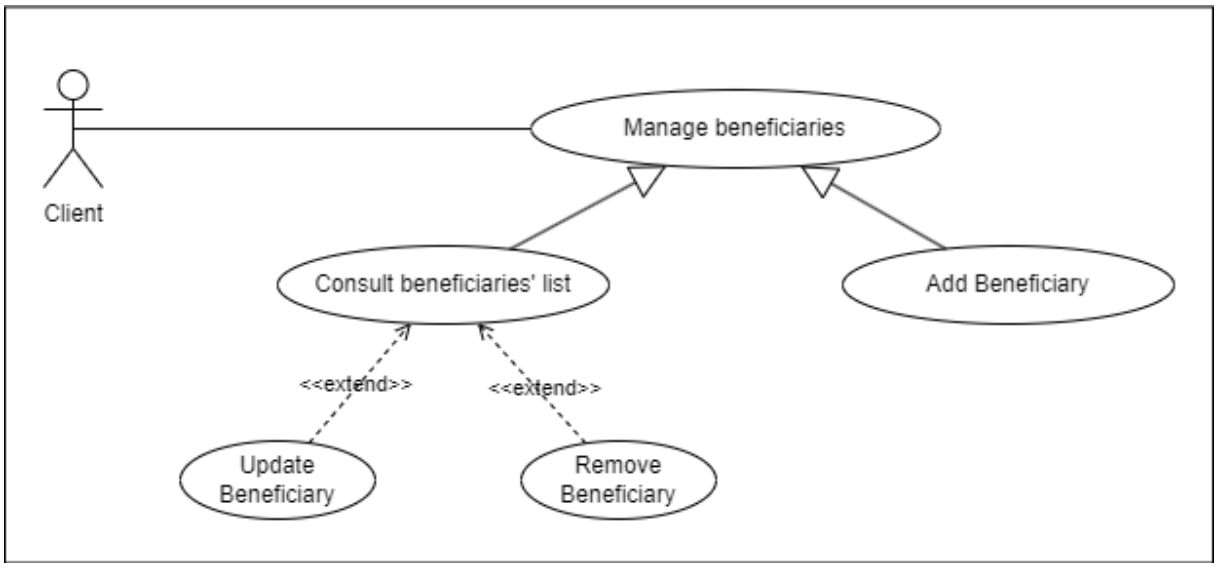


Figure 6.5: Refinement of « Manage Beneficiaries » use case

The table 6.6 illustrates a detailed scenario of client's interaction with « Manage Beneficiaries » use case.

Table 6.6: « Manage Beneficiaries » use case refinement.

Use case	Manage Beneficiaries
Actor(s)	Client
Pre-condition	Client is authenticated
Post-condition	Beneficiaries managed
Main scenario	- The system retrieves and displays the list of beneficiaries. - The client checks their list of beneficiaries.
Sub scenarios	- Add a beneficiary. - Update the name of a beneficiary. - Delete a beneficiary.
Exception	- The system displays a loading state if there is an error while retrieving, adding, updating, and deleting beneficiaries

6.3.6 Refinement of the « Transfer Money » Use Case

Transferring money is a core functionality in a banking application. Clients must first select a beneficiary and the amount of money to transfer. Upon approval, a verification email will be sent to confirm the transaction. To finalize the transfer, the client must click on the button in the email.
The refinement of the « Transfer Money » use case is visualized in figure 6.6.

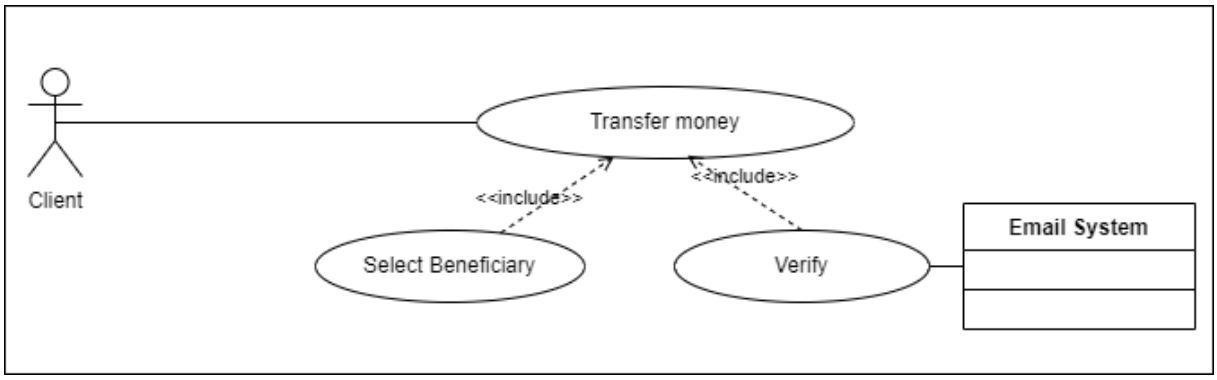


Figure 6.6: Refinement of « Transfer Money » use case

A detailed description of the interaction with « Transfer Money » use case is depicted in table 6.7. It shows that the client must be authenticated as a pre-condition, the various steps of transferring money is explained in the main scenario, specifying the sub-scenarios that should be included, and specify the potential errors that could arise.

Table 6.7: « Transfer Money » use case refinement.

Use case	Transfer Money
Actor(s)	Client, Email System
Pre-condition	Client is authenticated
Post-condition	Money transferred
Main scenario	- The system retrieves all beneficiaries and displays the transfer money screen. - The client selects a beneficiary from their beneficiaries' list. - The client then enters the amount of money to transfer and their motif. - The client clicks on transfer. - A verification email is sent to the client. - The client opens their email and clicks on verify. - The client then is redirected to the application and a validation message pops up on the screen.
Inclusion	- Email verification. - Select a beneficiary.
Exception	- The system displays an error message if the client's available balance doesn't allow them to make a transfer. - The system displays an error message if the client already has an initiated transfer that is not verified.

6.3.7 Refinement of the « Change Password » Use Case

Changing the password allows the client to update it when needed. The refinement of the « Change Password » use case is portrayed in figure 6.7.

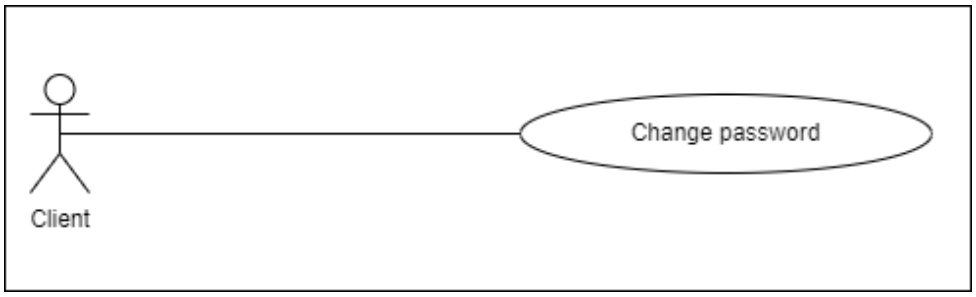


Figure 6.7: Refinement of « Change Password » use case

A detailed description of the « Change Password » scenario provided in table 6.8.

Table 6.8: « Change Password » use case refinement.

Use case	Change Password
Actor(s)	Client
Pre-condition	Client authenticated
Post-condition	Password changed
Main scenario	- The system displays the update password screen. - The client enters their current password. - The client enters their new password and confirm it.
Exception	- The system displays an error message if their current password is wrong. - The system displays an error message if the new passsword and the confirm new password are not matching and do not comply to the structure of a password.

6.4 Sprint 3 Modelling

This section will present the class and sequence diagram of the refined use cases.

6.4.1 « Check account's Balance » Use Case Modelling

Within this section, « Check account's Balance » use case will be conceptualized.

Class Diagram

Figure 6.8 represents the class diagram of the « Check account's Balance » use case.

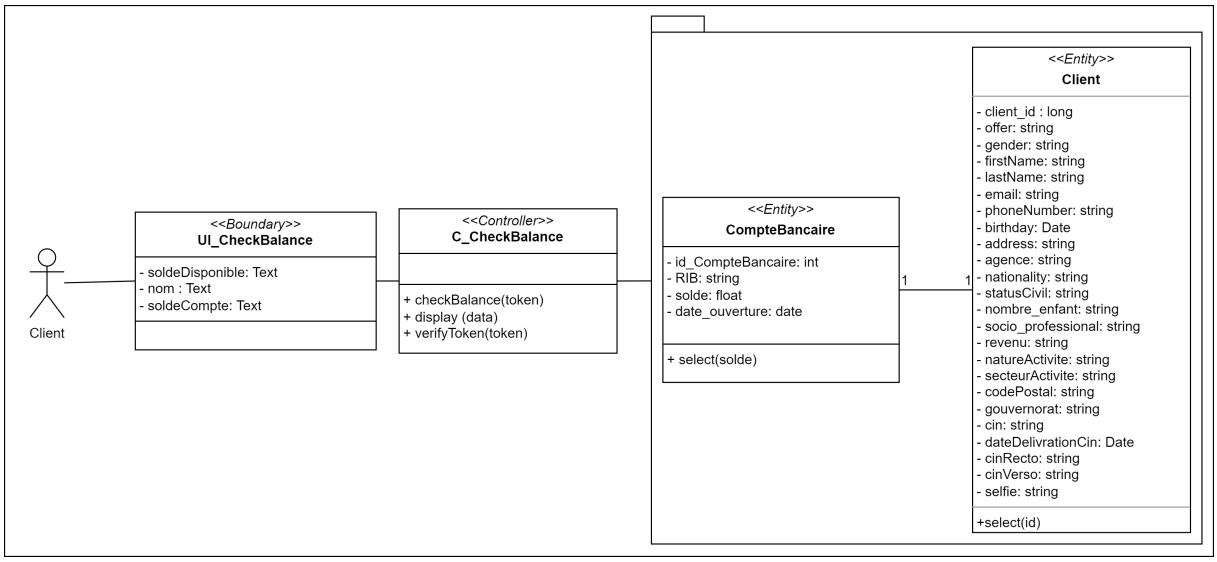


Figure 6.8: « Check account's Balance » Class Diagram

Sequence Diagram

When the home page first loads, the system sends a request to the server to fetch client details, such as the balance and the name. Before retrieving any information, the service checks the validity of the token. If it's expired, then the client will be logged out. The following figure represents the sequence diagram of the « Check account's Balance » use case (figure 6.9).

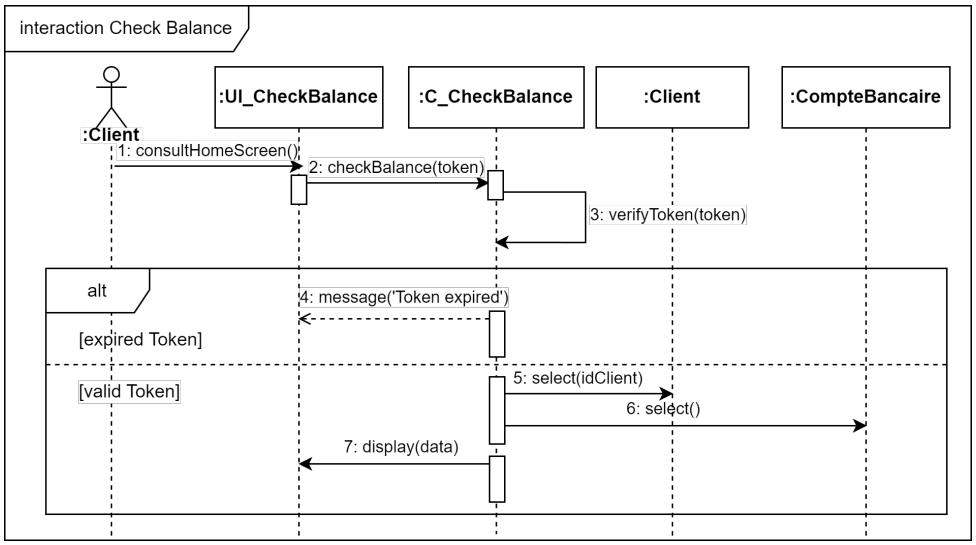


Figure 6.9: « Check account's Balance » Sequence Diagram

6.4.2 « Check history of Operations » Use Case Modelling

In the following, the sequence and class diagram of the « Check history of Operations » will be outlined.

Class Diagram

Figure 6.10 shows the class diagram of the « Check history of Operations » use case.

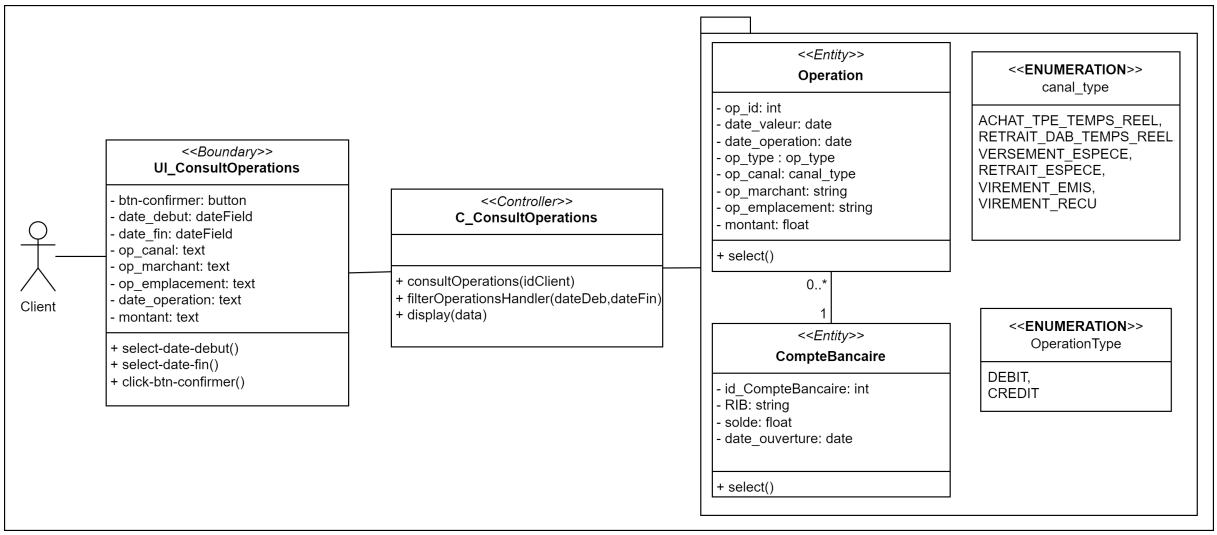


Figure 6.10: « Check history of Operations » Class Diagram

Sequence Diagram

When the 'Consult Operation' interface is opened, a request to retrieve bank account operations will be sent. If there are no operations, the screen will display 'no Operations found'. Otherwise, it will show the list of operations with the ability to filter by dates (figure 6.11).

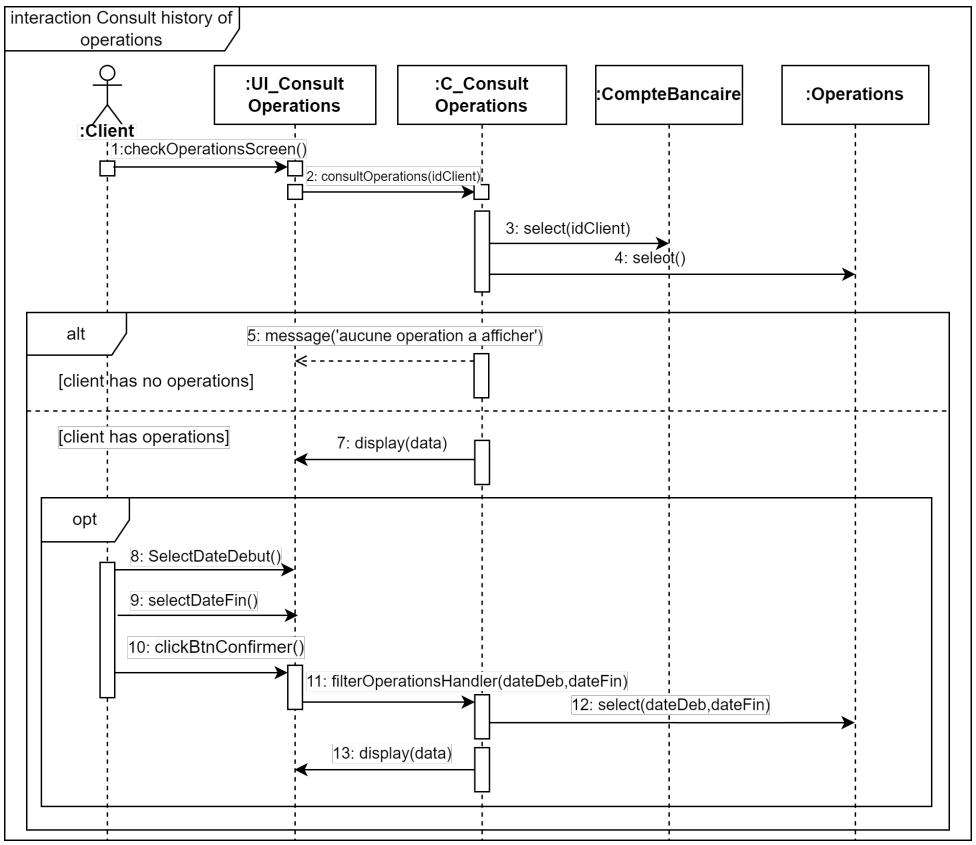


Figure 6.11: « Check history of Operations » Sequence Diagram

6.4.3 « Manage Card » Use Case Modelling:

During This section, the « Manage Card » use case will be conceptualized. The class diagram and sequence diagram will be detailed.

Class Diagram

Figure 6.12 depicts the class diagram of « Manage Card » use case.

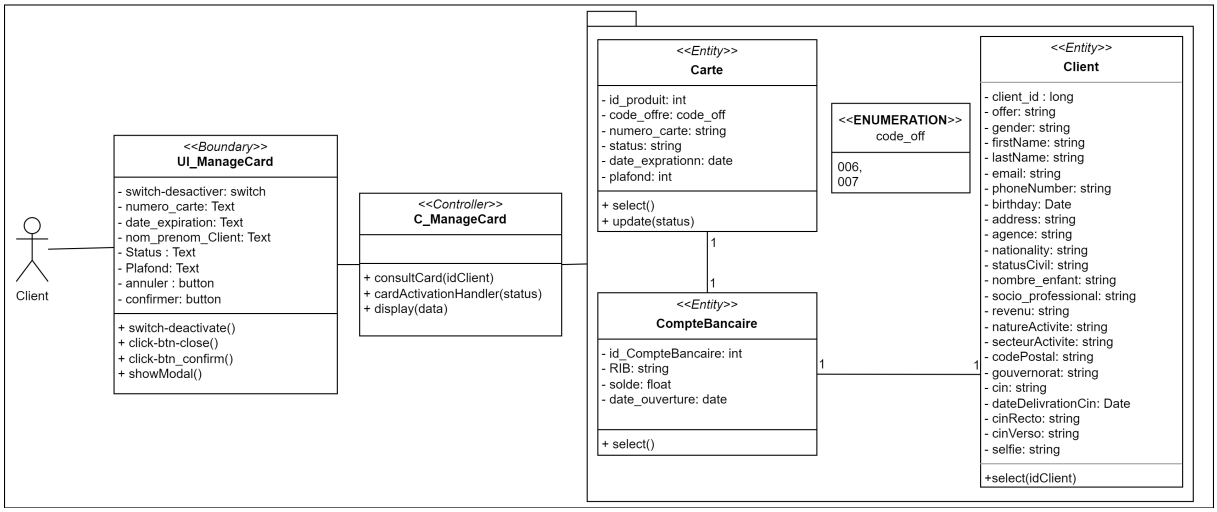


Figure 6.12: « Manage Card » Class Diagram

Sequence Diagram

When the 'Card' screen is accessed, the system will send a request to acquire the needed information. When everything is displayed, user can activate and deactivate their card through a switch button (figure 6.13).

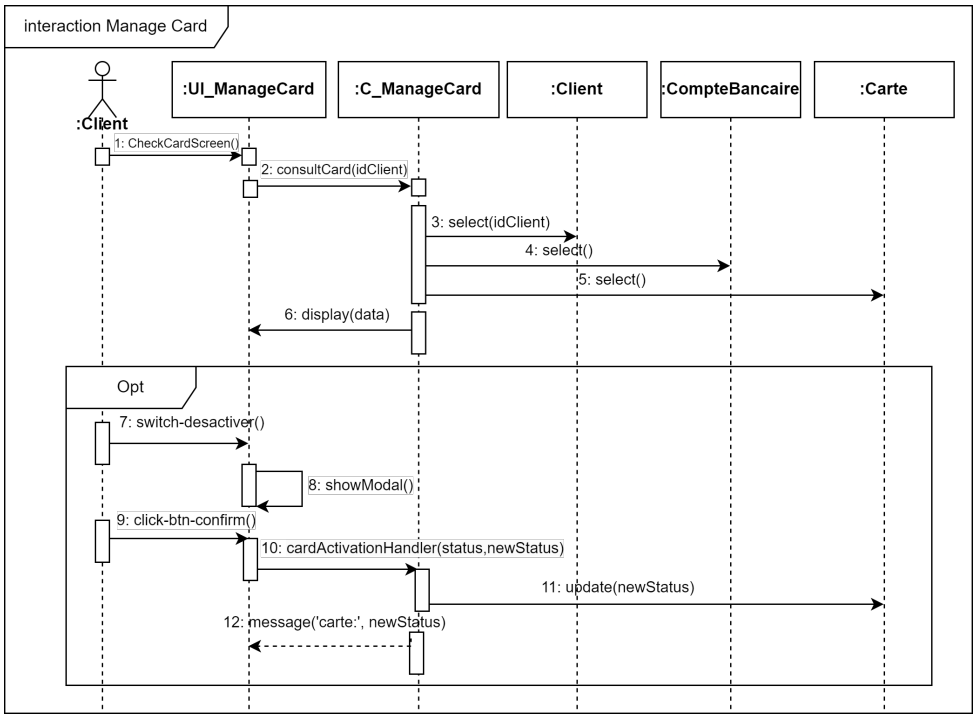


Figure 6.13: « Manage Card » Sequence Diagram

6.4.4 « Check history of transfers » Use Case Modelling

The proceeding section will outline the sequence and class diagrams of the « Check history of transfers » use case.

Class Diagram

A visual representation of the class diagram of the « Check history of transfers » use case is provided in figure 6.14.

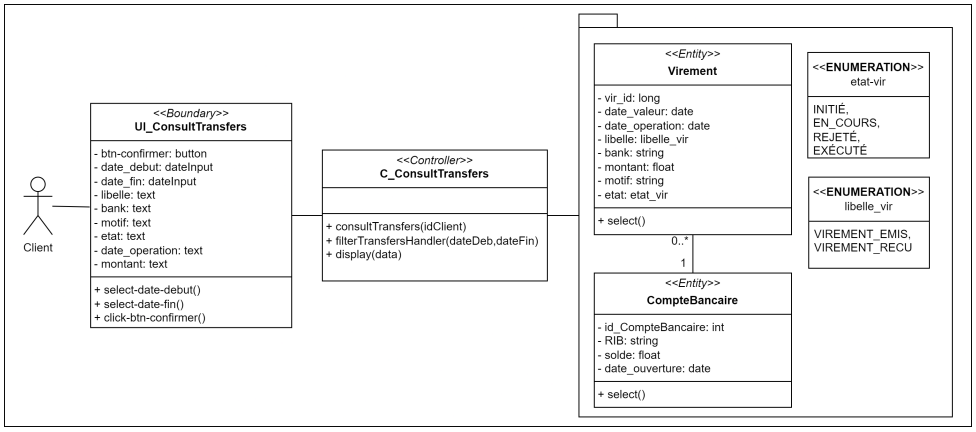


Figure 6.14: « Check history of transfers » class diagram

Sequence Diagram

When the 'Consult transfers' interface is opened, a request to retrieve bank account transfers will be sent. If there are no transfers, the screen will display 'no transfers found'. Otherwise, it will show the list of transfers with the ability to filter by dates (figure 6.15).

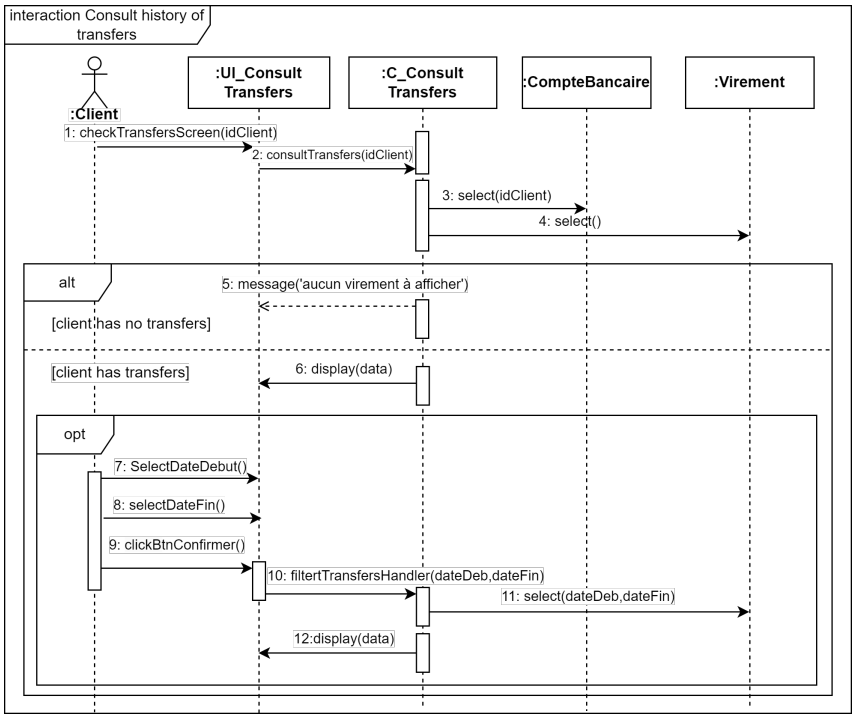


Figure 6.15: «Check history of transfer» Sequence Diagram

6.4.5 « Manage Beneficiaries » Use Case Modelling

The following section will provide the class and sequence diagrams of the « Manage Beneficiaries » use case.

Class Diagram

A visual representation of the class diagram of the « Manage Beneficiaries » use case is provided in figure 6.16.

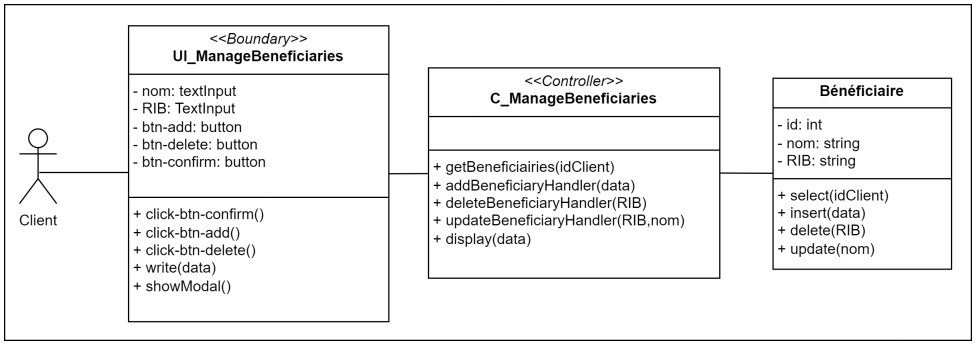


Figure 6.16: « Manage Beneficiaries » class diagram

Sequence Diagram

When the 'Manage Beneficiaries' screen is accessed, a request to get all the client's beneficiaries will be sent. If the client has no beneficiaries, a message with 'no beneficiaries found' will be displayed. Or else, it will display the list of beneficiaries (figure 6.17).

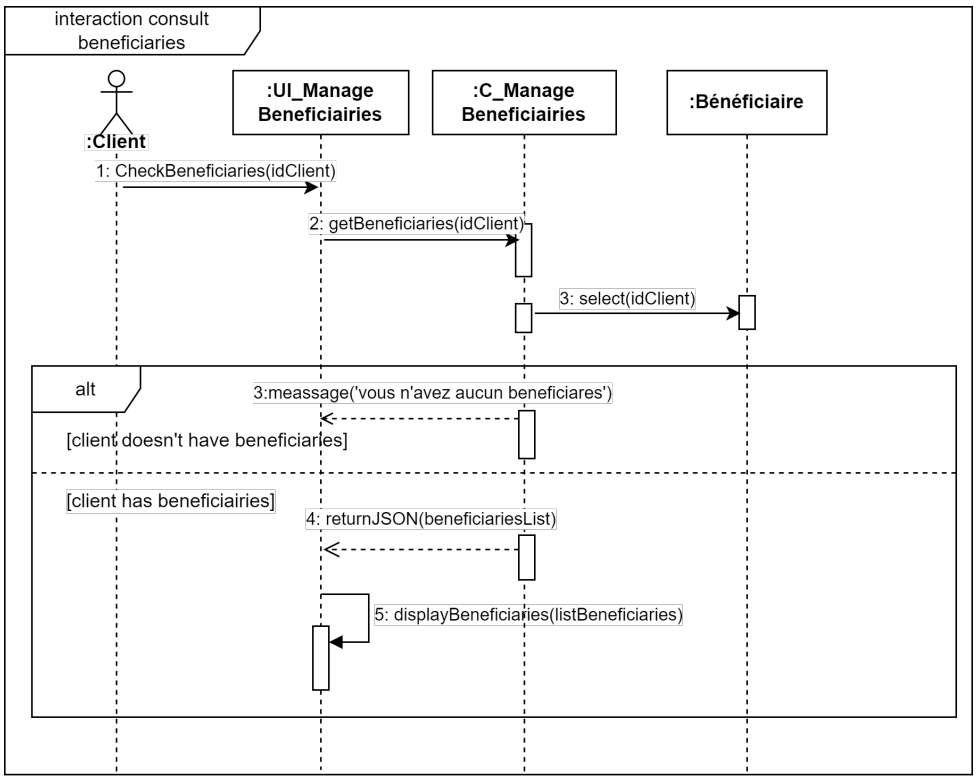


Figure 6.17: « Manage Beneficiaries » Sequence Diagram

Sequence diagram of « Add Beneficiary »

The client can add a new beneficiary by clicking the 'add' button. A pop-up will appear where they can enter the name and bank identity of the new beneficiary (figure 6.18).

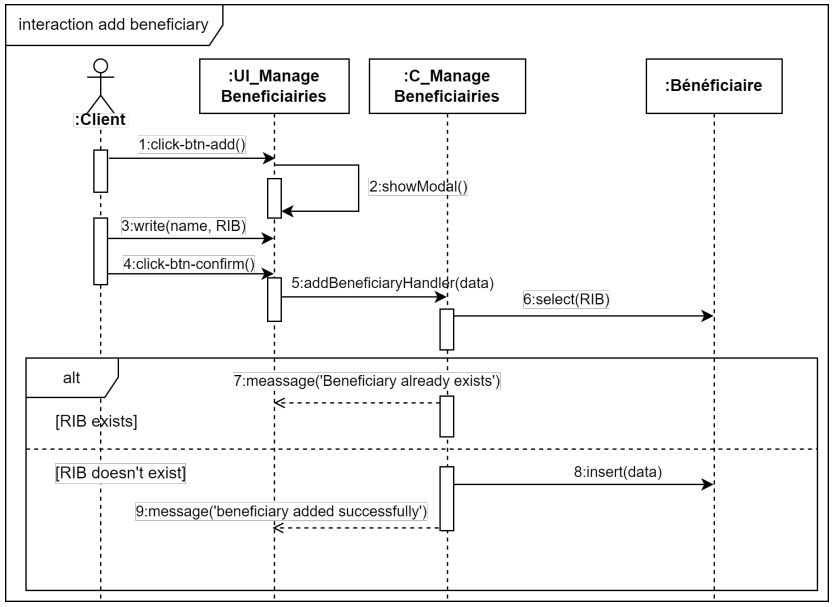


Figure 6.18: « Add Beneficiary » Sequence Diagram

Sequence diagram of « Update Beneficiary »

To update a beneficiary, the client should click on the button associated with the desired person, a pop-up will appear to update the name, and confirm the changes. If there is no error occurred the message with 'beneficiary is updated' will display. Otherwise, it will show the error message (figure 6.19).

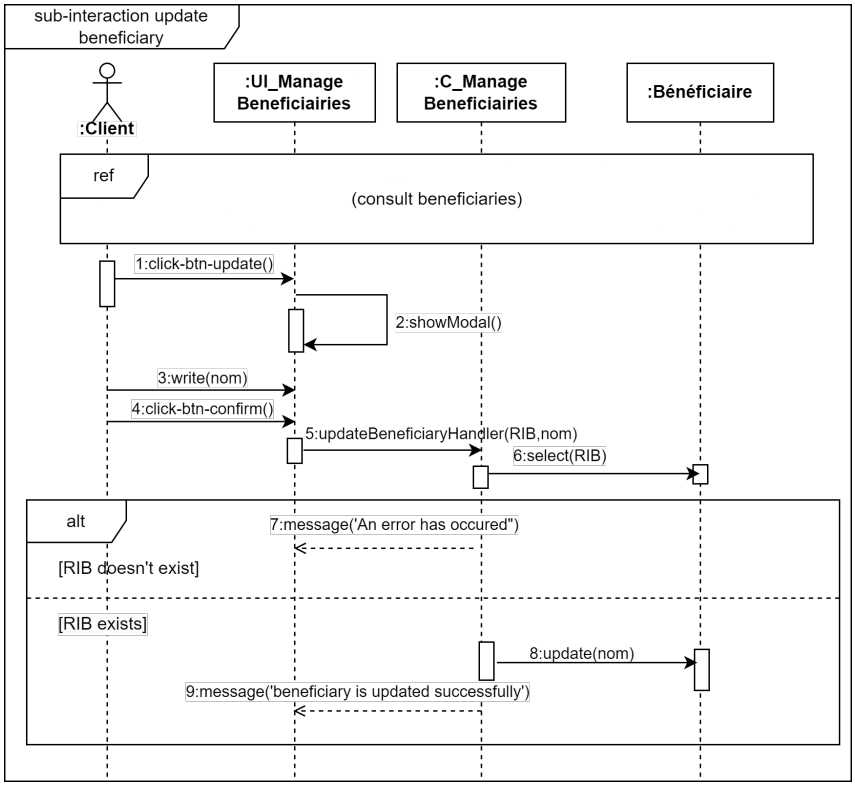


Figure 6.19: « Update Beneficiary » Sequence Diagram

Sequence diagram of « Delete Beneficiary » use case

To delete a beneficiary, the client should click on the button associated with the desired person, and confirm the operation. If there is no error occurred the message with 'beneficiary is deleted successfully' will display. Otherwise it will show the error message (figure 6.20).

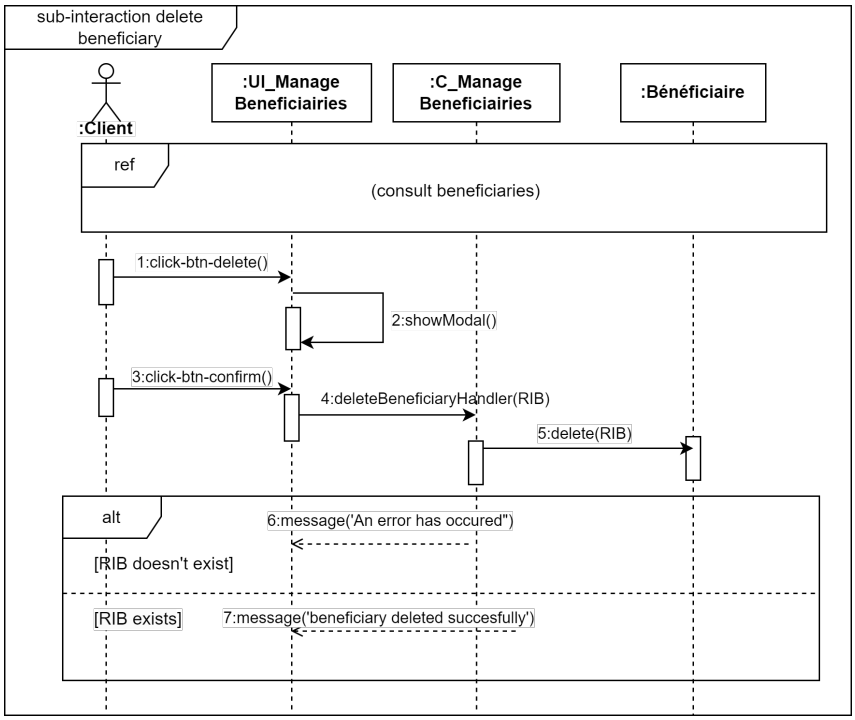


Figure 6.20: « Delete Beneficiary » Sequence Diagram

6.4.6 « Transfer Money » use case Modelling

During this section, the modelling of « Transfer Money » use case will be presented.

Class Diagram

Figure 6.21 represents the class diagram of « Transfer Money » use case.

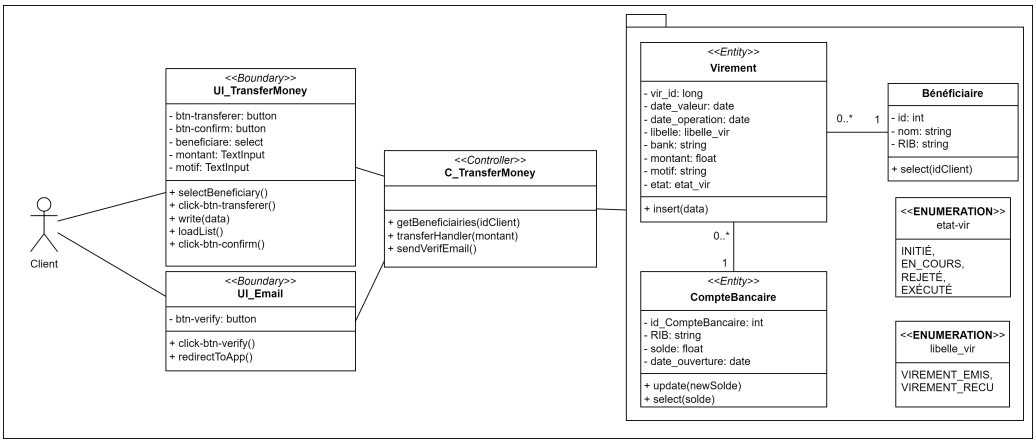


Figure 6.21: « Transfer Money » class diagram

Sequence Diagram

Upon accessing the 'transfer money' screen, a request to fetch the list of beneficiaries will be sent. The client can select one beneficiary, provide a reason (motif) for the transfer, and specify the amount to send. If the client already has an initiated transfer they will get an error message. Otherwise, if the transfer amount is less than the bank account's balance, a validation email will be sent with a feedback. In the email, once the 'verify' button is clicked, the status of the transfer would change from 'initiated' to 'executed' and the balance will be updated (figure 6.22).

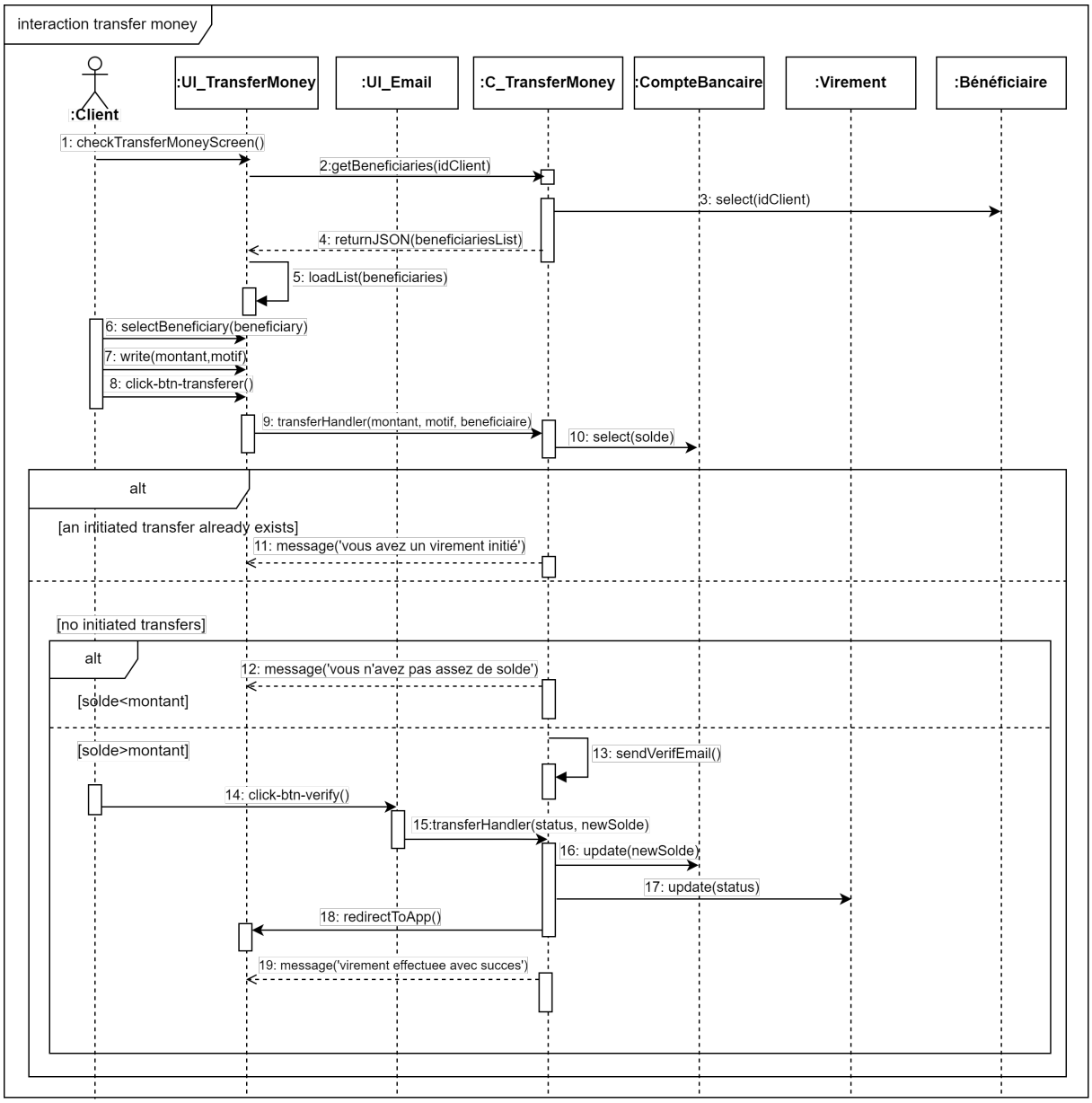


Figure 6.22: « Transfer Money » Sequence Diagram

6.4.7 « Change Password » Use Case Modelling

During this section, the « Change Password » use case will be conceptualized.

Class Diagram

Figure 6.23 represents the class diagram of the « Change Password » use case.

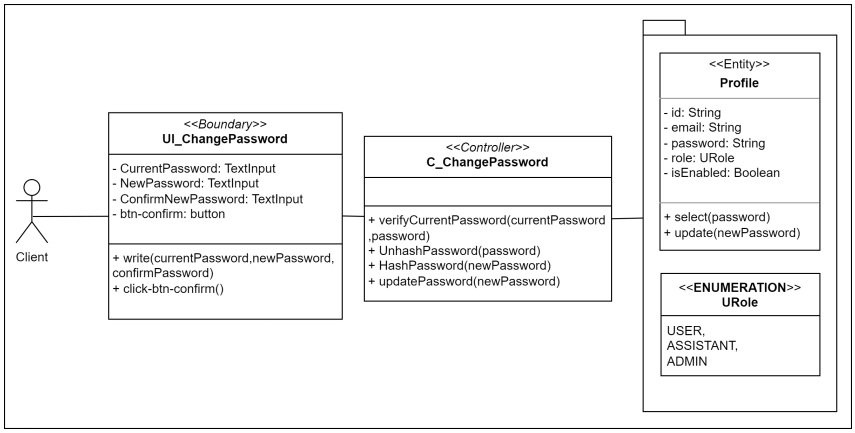


Figure 6.23: « Change Password » class diagram

Sequence Diagram

In order to update their password, the client must enter his current password, the new one and confirm it. If the current password matches the password stored in the database, the password will be changed and a success message will be displayed. Otherwise, it will display 'Password is wrong' (figure 6.24).

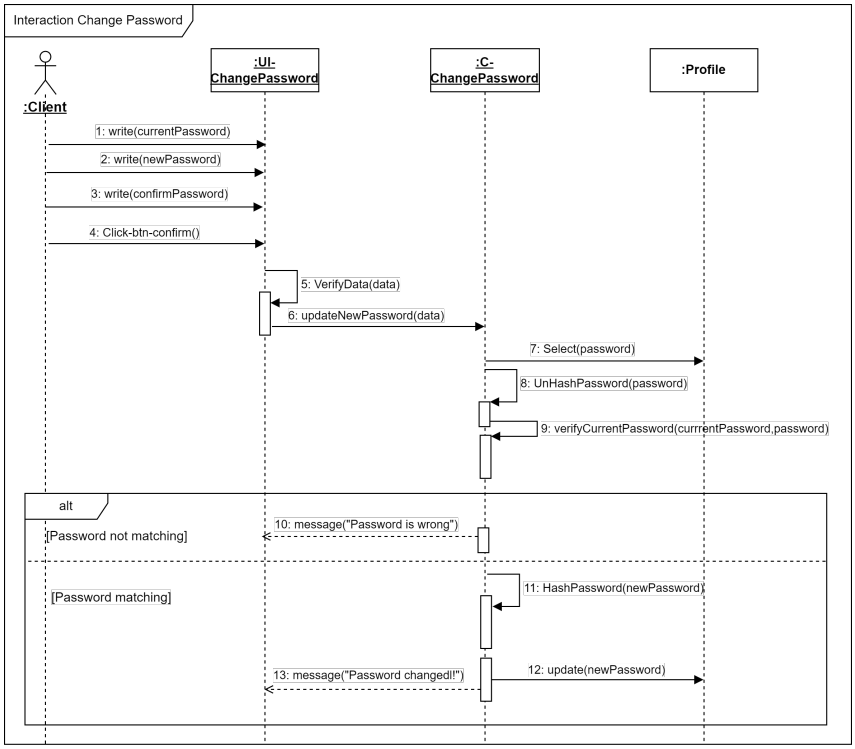


Figure 6.24: « Change Password » Sequence Diagram

6.5 Sprint 3 Class Diagram

A visual representation of the global class diagram of sprint 3 is provided in figure 6.25

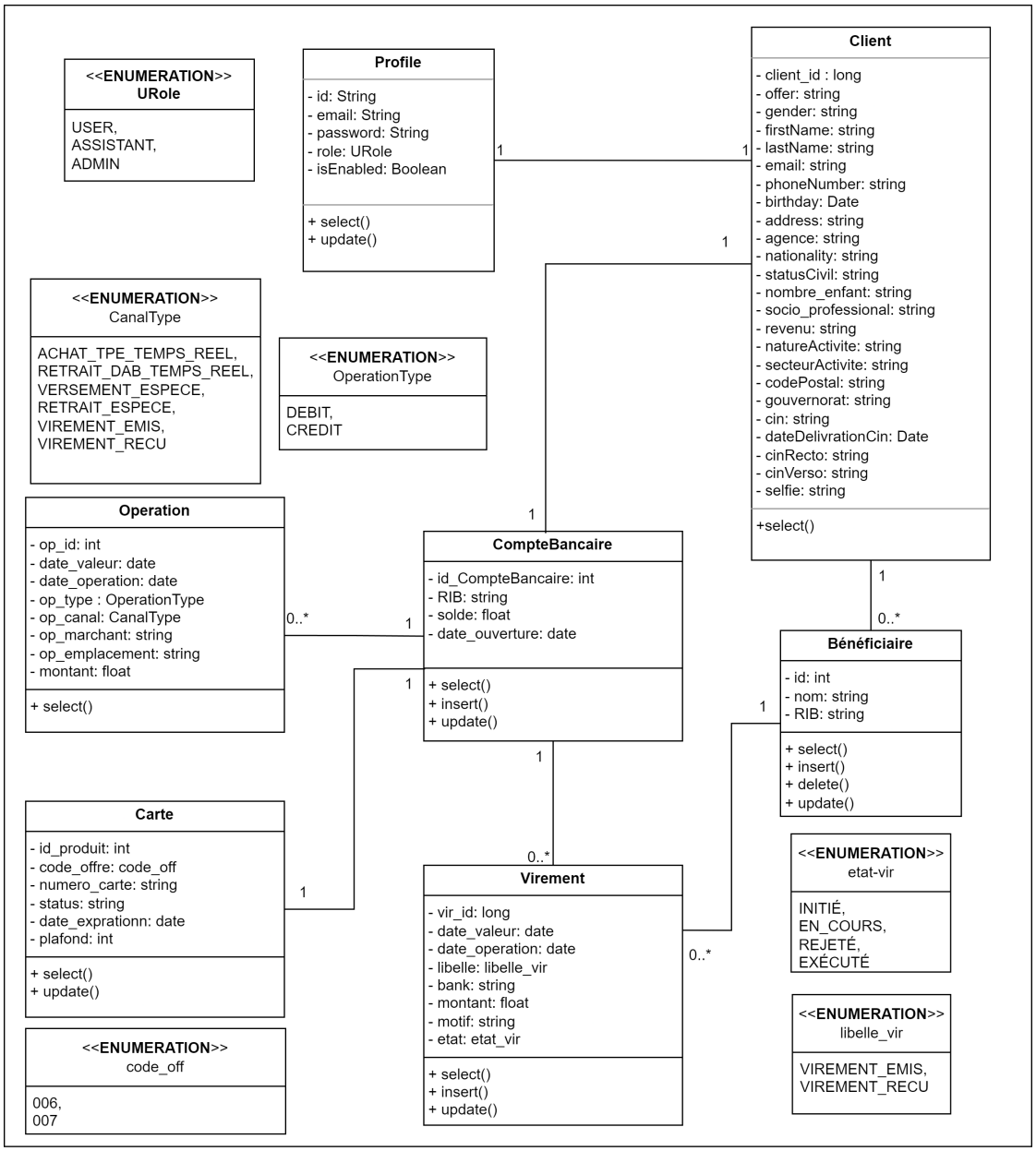


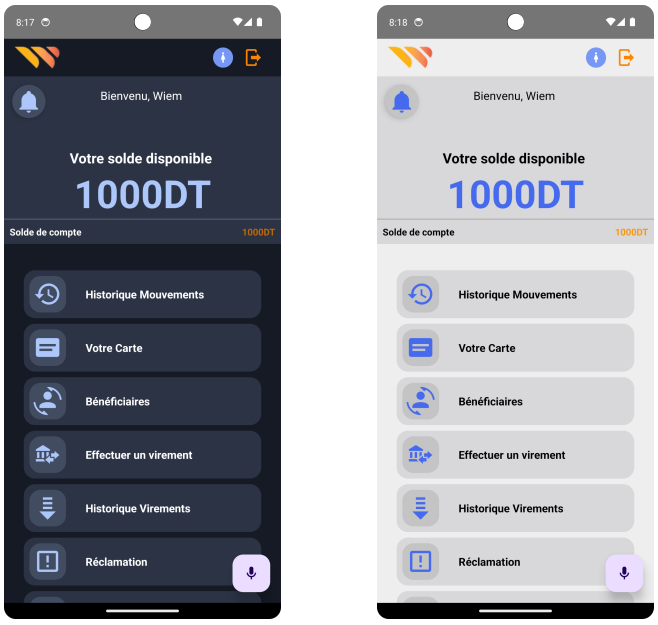
Figure 6.25: Sprint 3 Global Class Diagram

6.6 Realization

This section presents the realization of the use cases identified for Sprint 3, including screenshots of the implemented features.

6.6.1 «Check Balance» Use Case Realization

When the client is signed in, they will be redirected to the home page, where he can check their balance and access to other features of the application (figure 6.26).



(a) Dark mode (b) Light mode

Figure 6.26: «Check account's balance» Use Case Realization

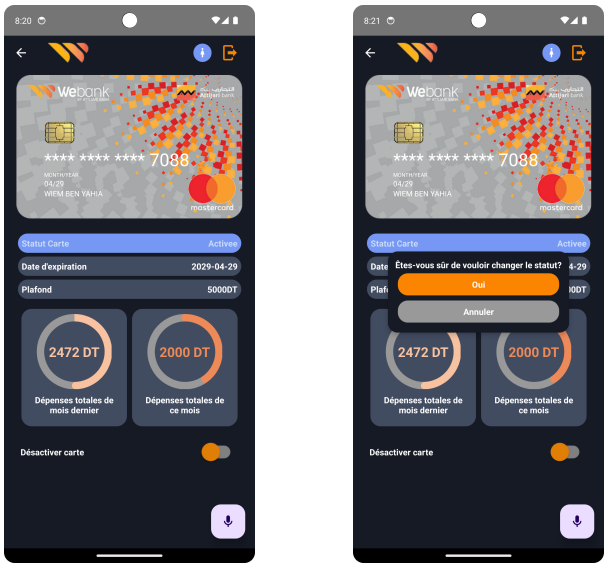
6.6.2 «Consult Operations» Use Case Realization

When clicking on 'Operations' button, the client can check and filter their list of operations (figure 6.27).

Figure 6.27: «Consult operations» Use Case Realization

6.6.3 «Check Card» Use Case Realization

Clicking on the 'Card' button on the home page will lead to the screen in figure 6.28, where they can check various information about the card and change the status of their card.



(a) Card interface (b) Card status modal

Figure 6.28: «Manage card» Use Case Realization

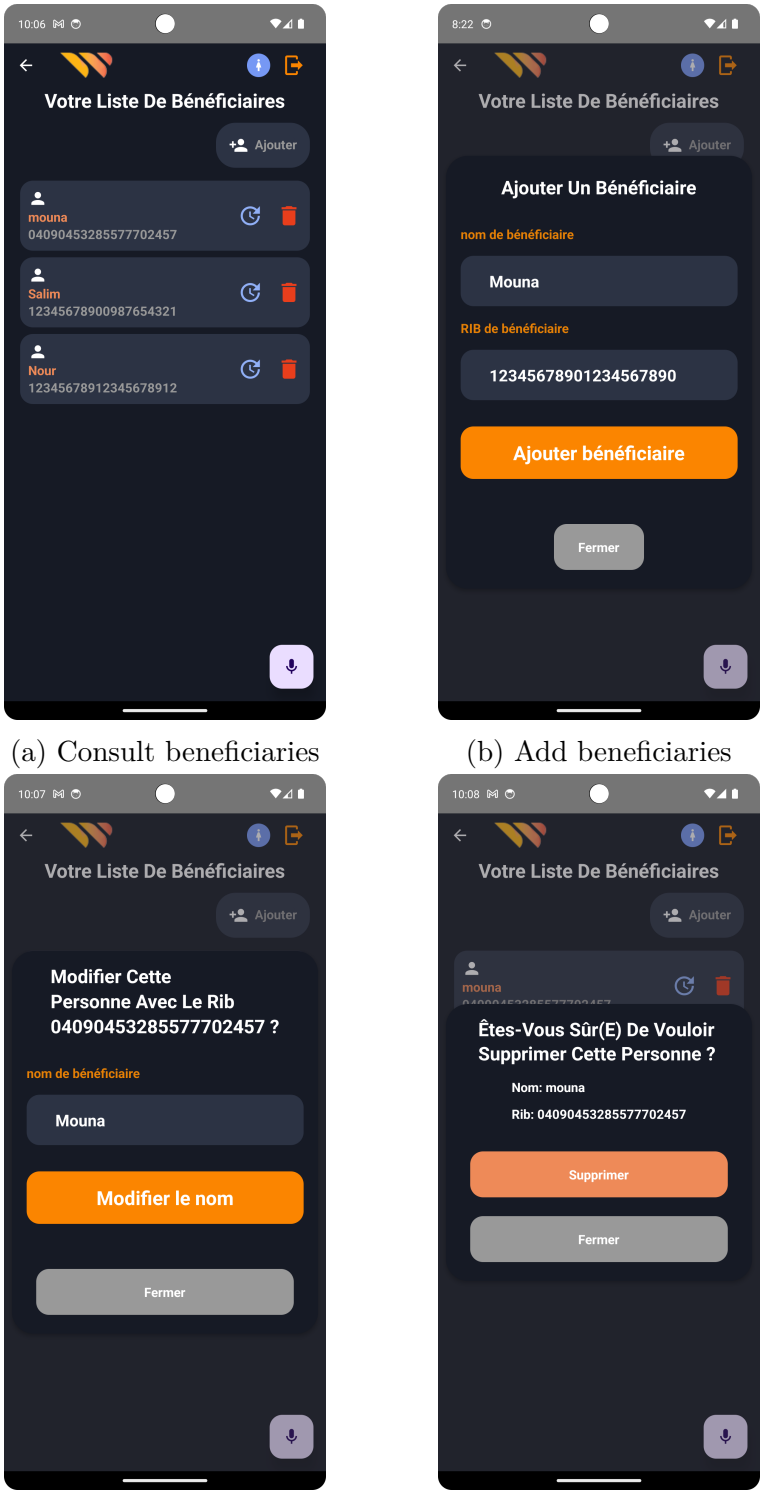
6.6.4 « Check history of transfers » Use Case Realization

The client can check and filter the history of transfers as visualized in figure 6.29.

Figure 6.29: «Check history of transfers» Use Case Realization

6.6.5 « Manage Beneficiaries » Use Case Realization

Upon clicking on 'Manage Beneficiaries' button on the home page, the client can check their beneficiaries (figure 6.30a), add (figure 6.30b), update (figure 6.30c), and delete a beneficiary (figure 6.30d).



(a) Consult beneficiaries (b) Add beneficiaries

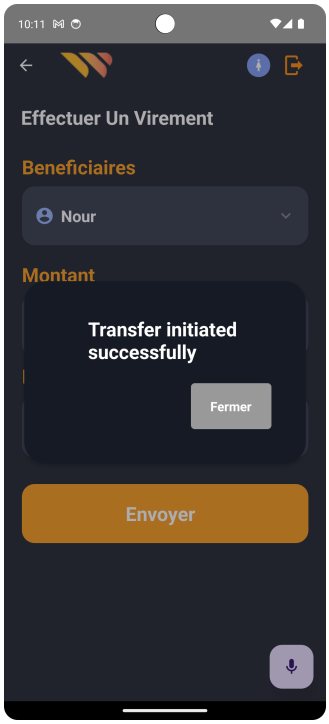
(c) Update beneficiary (d) Delete beneficiary

Figure 6.30: « Manage Beneficiaries » Use Case Realization

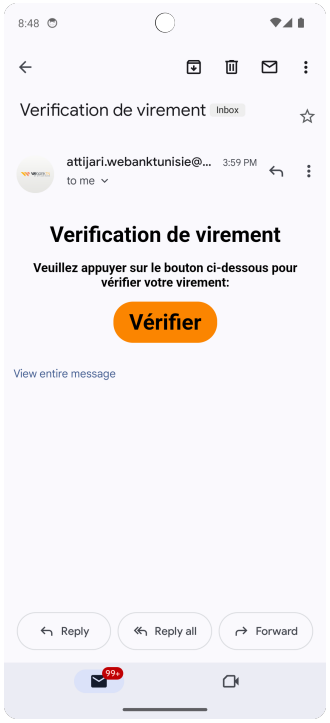
6.6.6 « Transfer Money » use case Realization

Clicking on 'Transfer Money' button will lead the client to screen presented in figure 6.31a. The client has to select a beneficiary and fill other information, and upon clicking the 'send' the modal in figure 6.31b, the client then will be sent a verification email which will redirect them back to the app with the information validation screen shown in figure 6.31d.

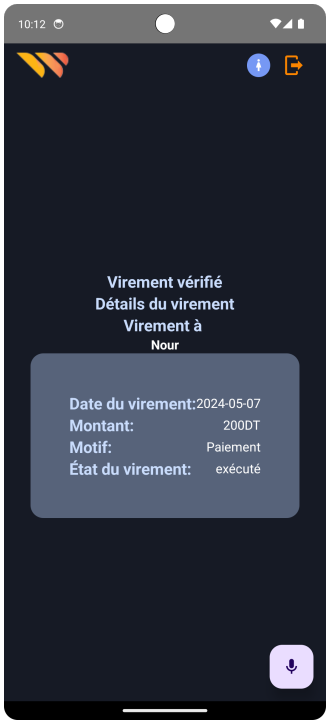
(a) Transfer Money interface



(b) Confirmation Modal



(c) Transfer Email Validation

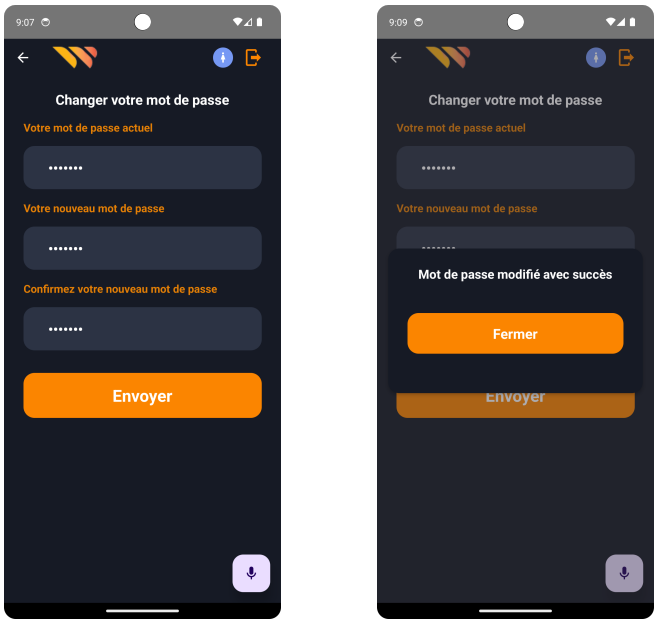


(d) Validation

Figure 6.31: Transfer Money Process

6.6.7 « Change Password » Use Case Realization

The client can update their password by entering their current password and their new password respectively as visualized in figure 6.32.

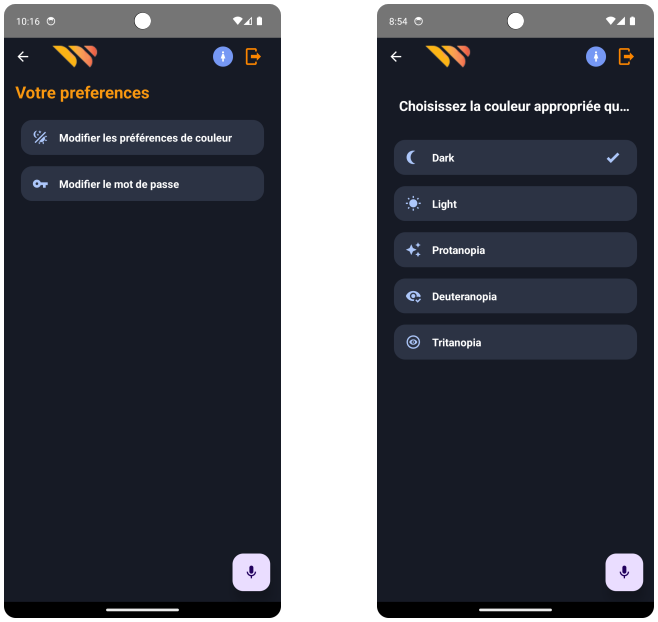


(a) Change password (b) Change password modal

Figure 6.32: « Change password» Use Case Realization

6.6.8 «Change Settings» Backlog Item Realization:

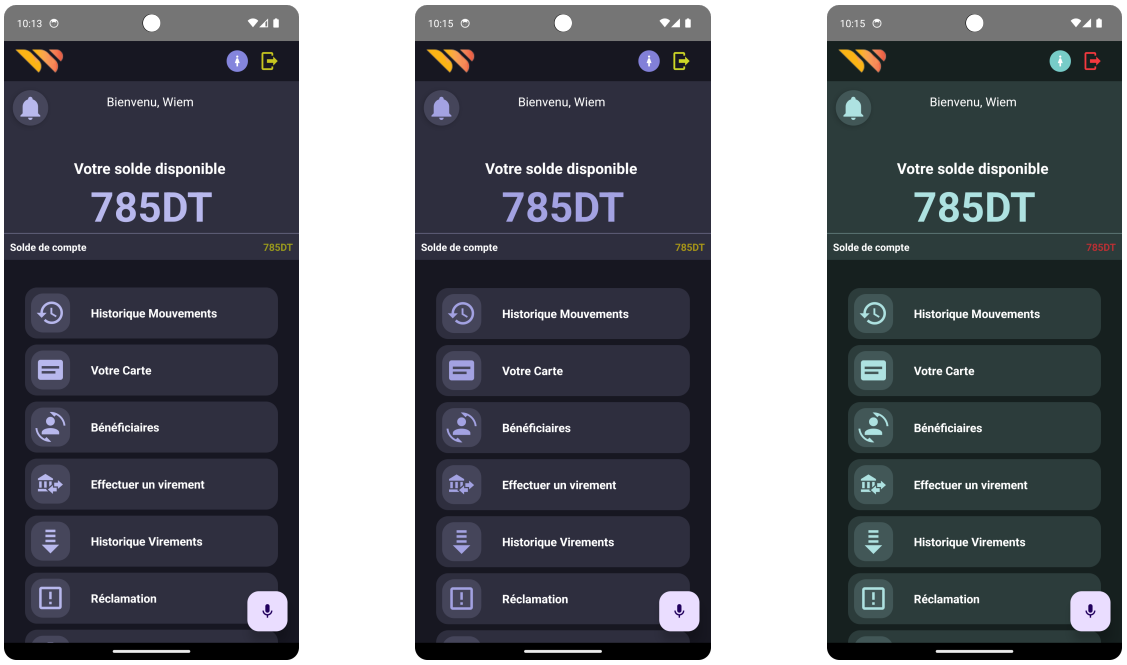
The client can access the settings interface in which they have the option to modify the color mode or change their password as shown in Figure 6.33.



(a) Settings Interface (b) Color Preferences

Figure 6.33: Change Settings

Figure 6.34 below shows the additional color modes that are especially made for those with color blindness.



(a) Home - Protanopia (b) Home - Deuteranopia (c) Home - Tritanopia

Figure 6.34: Color Blind Modes

6.7 Conclusion

This chapter presented a well constructed modelling phase for "Sprint 3: Balance and transactions" explaining each use case in detail and then providing breakthrough over the implementation.
The next chapter will be a detailed overview of the next sprint titled 'Administration and client relations'.

Chapter 7

Sprint 4: Administration and Client Relations

7.1 Introduction

This chapter will provide a comprehensive introduction to the final sprint, 'Administration and Client Relations'. As the title suggests, it will focus on features that include the client's relationship with the bank assistant and provides access for assistants to manage client requests.
This chapter will begin with specifying the product backlog, refining and conceptualizing use cases, and ends with the implementation of each use case.

7.2 Sprint 4 Backlog Identification

The items planned for this sprint are presented in Table 7.1.

Table 7.1: Sprint 4's Backlog

Backlog Item	Priority	Estimation	Planning
As a client, I can contact bank's assistant.	4	Average	Sprint 4
As a client, I can check my notifications.	4	Average	Sprint 4
As an Admin, I can manage assistants	4	Average	Sprint 4
As a (Assistant + Admin), I can view dashboard.	4	Average	Sprint 4
As a (Assistant + Admin), I can manage clients request.	4	Average	Sprint 4

7.3 Refinement of Sprint 4

During this section, the following features will be refined:

- Contact assistants;
- Check notifications;
- Manage assistants;
- View Dashboard;
- Manage clients request.

7.3.1 Refinement of the « Contact Assistants » use case

When the clients face issues navigating the application, they need a way to contact assistant to provide guidance.

Figure 7.1 illustrates the refinement of the « Contact Assistants » use case.

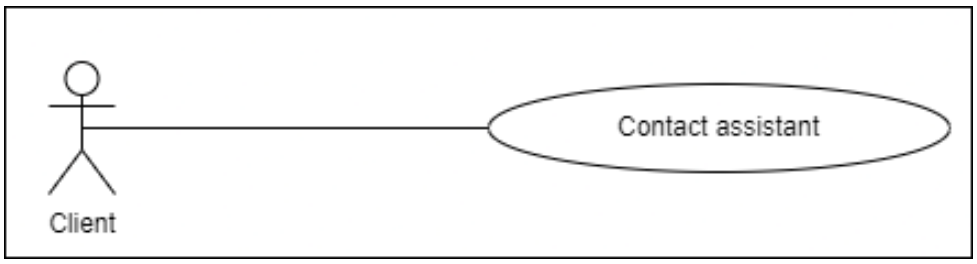


Figure 7.1: Refinement of the « Contact Assistants » use case

The following Table presents the refinement of the « Contact Assistants » use case:

Table 7.2: « Contact Assistants » use case refinement.

Use case	Contact assistants
Client(s)	Clients
Pre-condition	Client is authenticated
Post-condition	Client sent a contact to an assistant
Main scenario	- The system displays the interface. - The client select an object about their claim. - The client write their claim. - The Client clicks the send button to send their claim to the assistant.
Exception	- The system displays an error message if there is an issue with sending the claim messages.

7.3.2 Refinement of the « Check notifications » use case

A notification is sent to the client when a money transfer operation is successful or when the client receives a response from the assistant.

Figure 7.2 presents the refinement of the « Check notifications » use case.

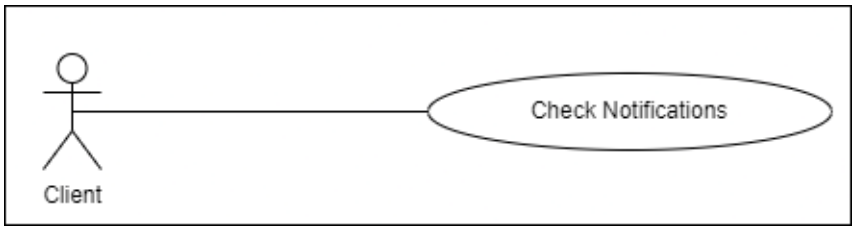


Figure 7.2: Refinement of the « Check notifications » use case

Table 7.3 presents the refinement of the « Check notifications » use case.

Table 7.3: « Check notifications » use case refinement.

Use case	Check notifications
Client(s)	Clients
Pre-condition	Client is authenticated
Post-condition	Client has checked all his notifications.
Main scenario	- The system loads the notifications. - The Client checks all their notifications.
Exception	- The system displays an error message if there are issues with retrieving or displaying notifications.

7.3.3 Refinement of the « Manage Assistants » use case

The administrators possess control over assistants within the system. This includes adding and removing an assistant.
Figure 7.3 illustrates the refinement of the « Manage Assistants » use case.

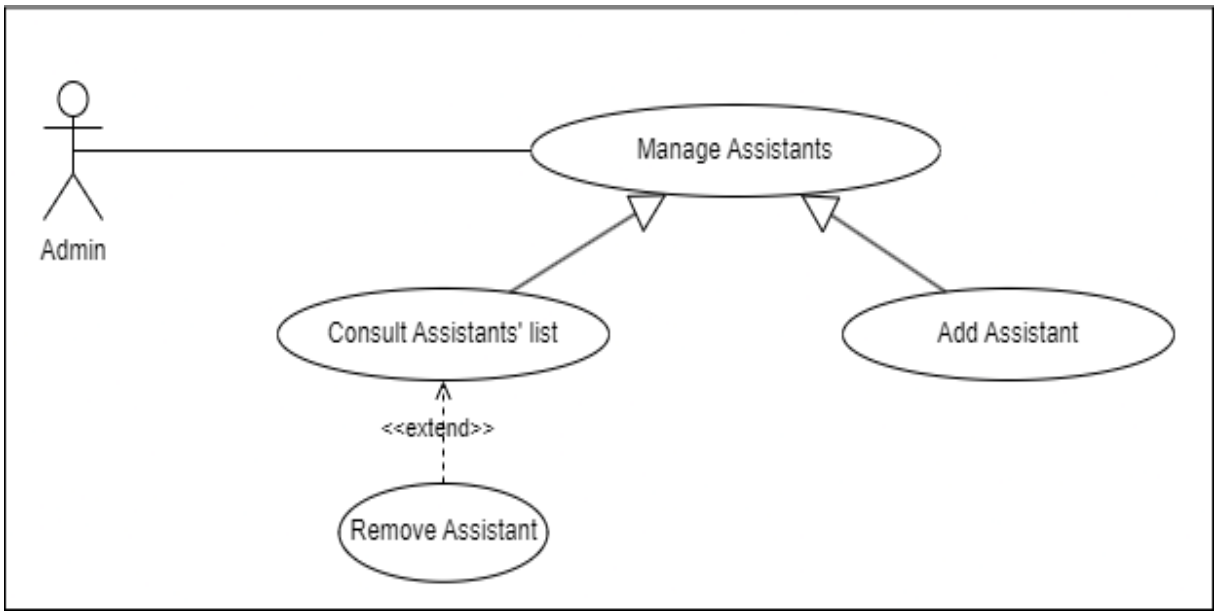


Figure 7.3: Refinement of the « Manage Assistants » use case

Table 7.4 presents the refinement of the « Manage Assistants » use case:

Table 7.4: « Manage Assistants » use case refinement.

Use case	Manage Assistants
Client(s)	Admin
Pre-condition	Admin is authenticated.
Post-condition	Admin has successfully added or deleted an assistant.
Main scenario	- The system loads the list of assistant in the interface and displays them. - Admin adds a new assistant. - Admin deletes an assistant.
sub-scenarios	- Add an assistant. - Remove an assistant.
Exception	- The system displays an error message if there are issues with retrieving or displaying assistants.

7.3.4 Refinement of the « Manage Clients requests » use case

Administrators and bank assistants can check client requests and respond to them. Figure 7.4 provide a refinement of « Manage Clients requests » use case.

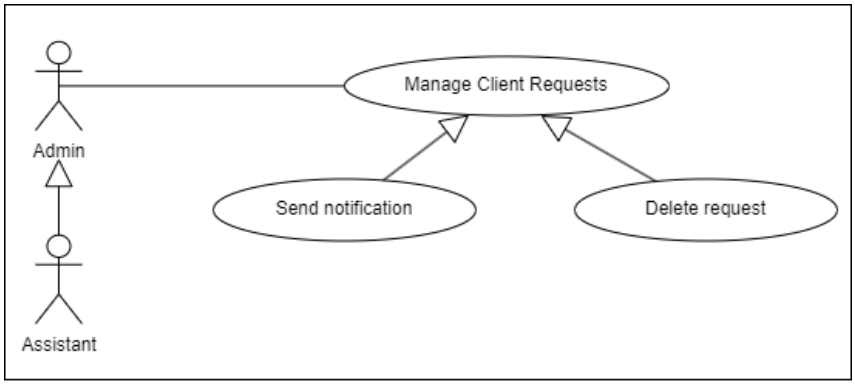


Figure 7.4: Refinement of the « Manage Clients requests » use case

Table 7.5 provides an overview of administrators and assistants interaction with « Manage Clients requests ». It specifies the pre and post condition, the main and sub scenario, errors that could arise during this interaction.

Table 7.5: « Manage Clients requests » use case refinement.

Use case	Manage Clients requests
Client(s)	Admin and Assistant
Pre-condition	Actors is authenticated.
Post-condition	Actors has successfully managed a client request.
Main scenario	- The system loads the list of clients request in the interface and displays them. - Actors responds to client's request. - Actors deletes a clients request.
sub-scenarios	- Responds to a client's request. - Delete a request.
Exception	- The system displays an error message if there are issues with retrieving or displaying client's request.

7.4 Modelling of Sprint 4

In this section, a presentation of the class and sequence diagrams of Sprint 4 use cases will be provided.

7.4.1 « Contact Assistants » Use Case Modelling

The following section will provide the modelling of « Contact Assistants » use case.

Class Diagram

Figure 7.5 presents the class diagram of « Contact Assistants » use case.

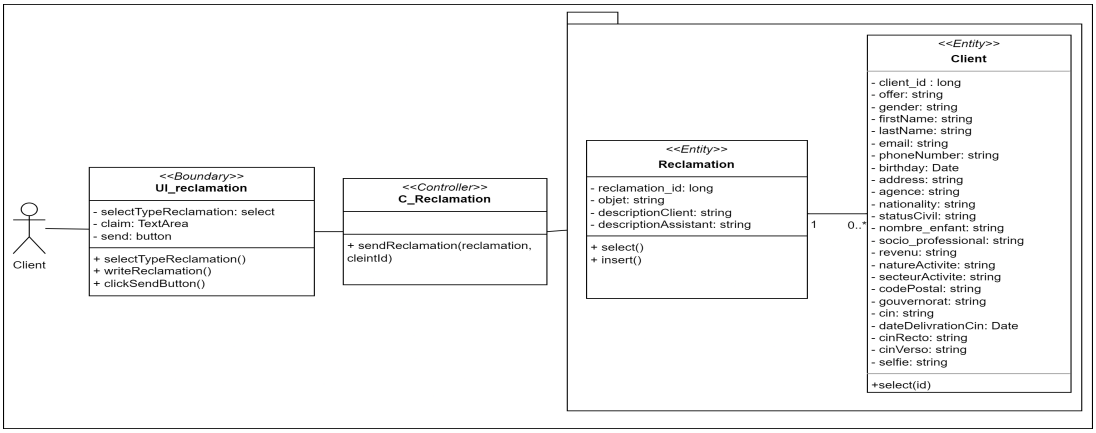


Figure 7.5: Class diagram for the « Contact Assistants » Use Case.

Sequence diagram for the « Contact Assistants » use case

To contact the assistant, the client must select the subject of their inquiry and a description of their needs (figure 7.6).

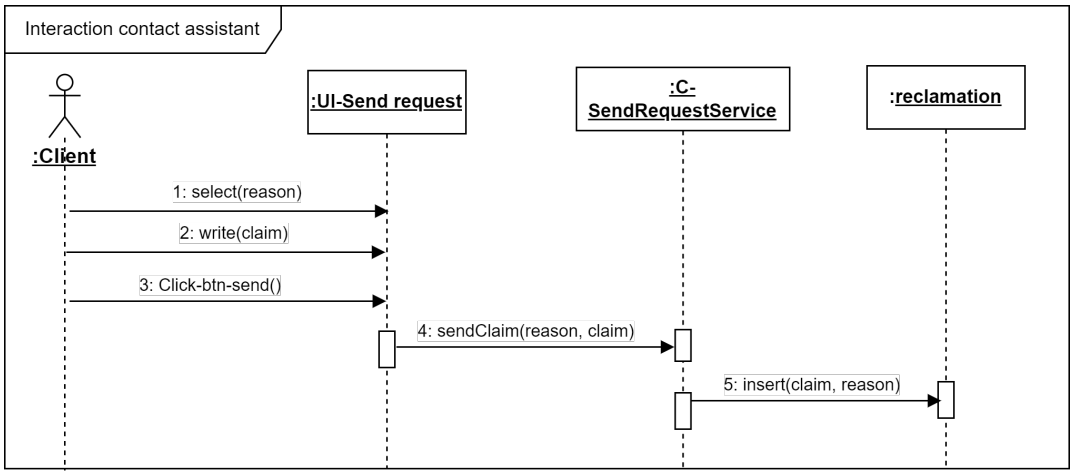


Figure 7.6: « Contact Assistants » Sequence Diagram

7.4.2 « Check notifications » Use Case Modelling

In the following section, the « Check notifications » use case will be conceptualized.

Class Diagram

Figure 7.7 visualize the class diagram of « Check notifications » use case.

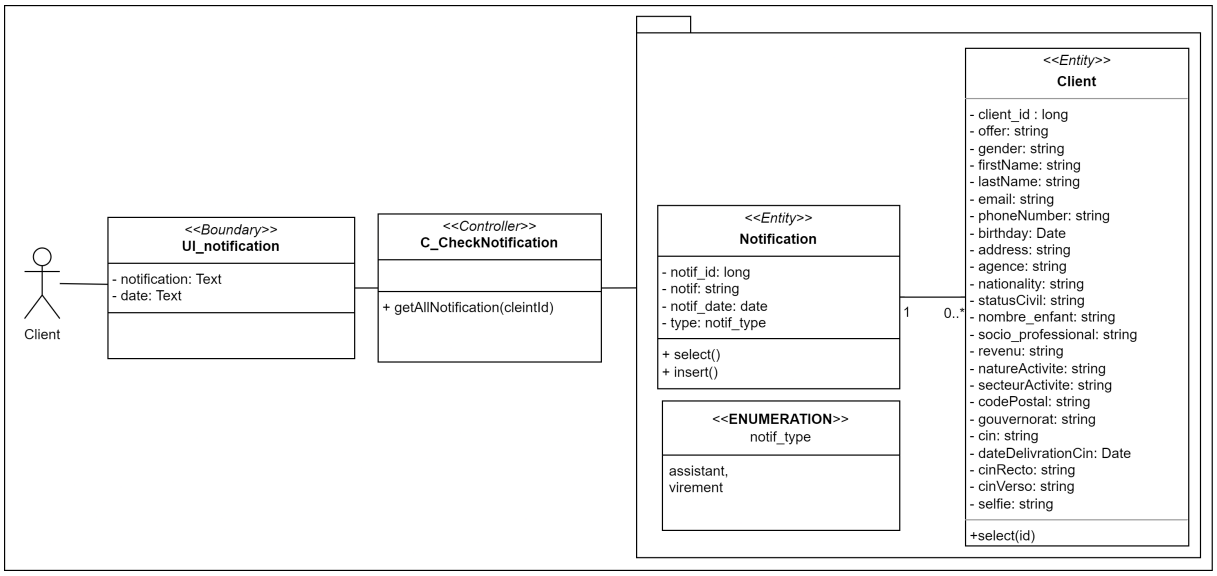


Figure 7.7: Class diagram for the « Check notifications » use case.

Sequence diagram for the « Check notifications » use case

Figure 7.8 provides a visualization of the sequence diagram of the « Check notifications » use case.

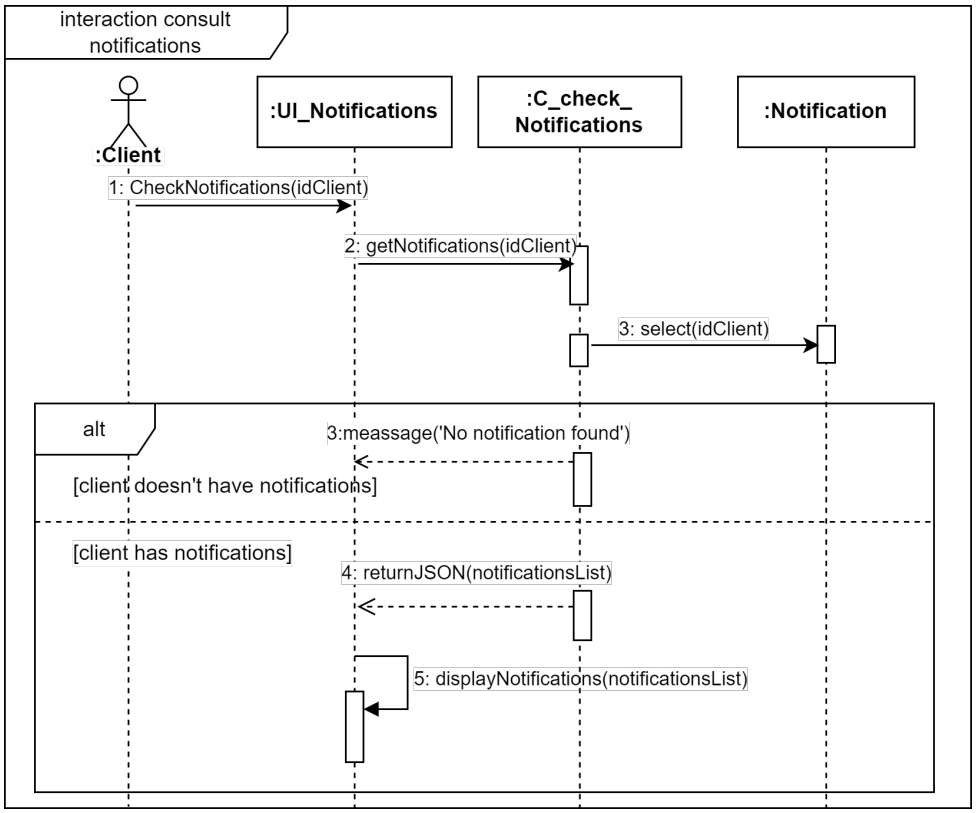


Figure 7.8: « Check notifications » Sequence Diagram

7.4.3 « Manage Assistants » Use Case Modelling

This part will focus on the class and sequence diagrams of manage assistants.

Class Diagram

Figure 7.9 illustrates the class diagram for « Manage Assistants » use case

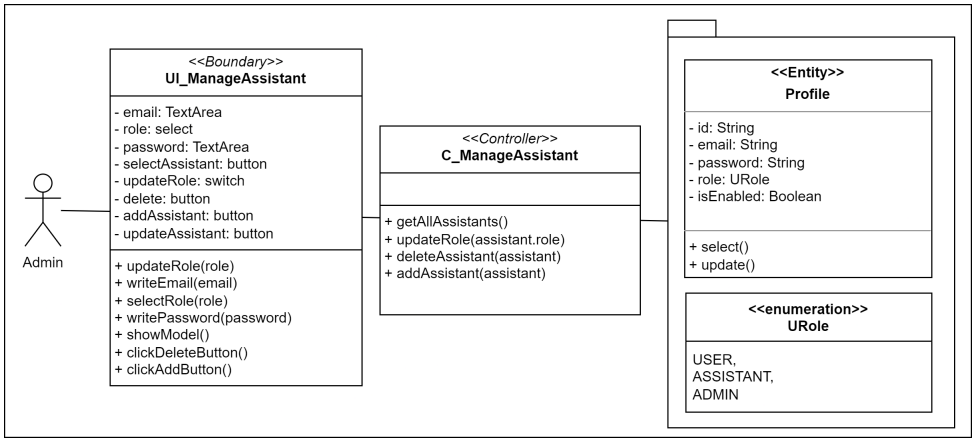


Figure 7.9: Class diagram for the « Manage Assistants » use case.

Sequence Diagrams

Administrators can check and provide access to assistants, as well as add or delete them (figure 7.10).

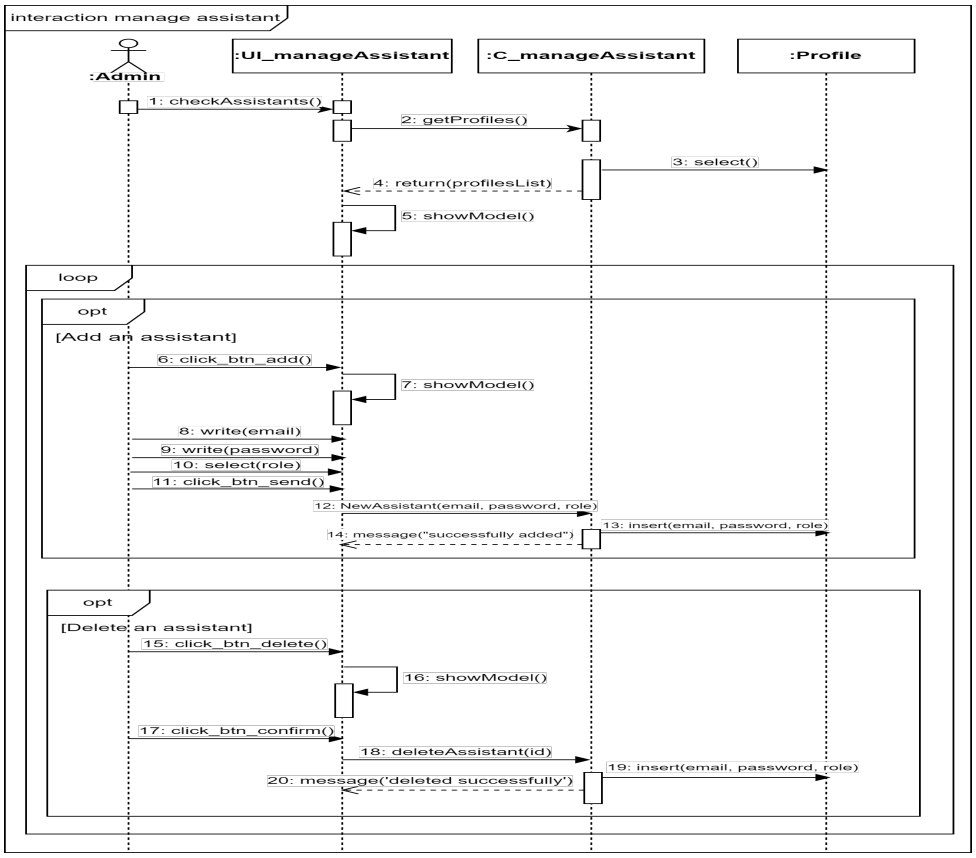


Figure 7.10: « Manage Assistants » Sequence Diagram

7.4.4 « Manage Clients Request » Use Case Modelling

The following section will provide an overview of the class and sequence diagrams of the « Manage Clients Request » use case.

Class Diagram

Figure 7.11 presents the class diagram for « Manage Clients Request » use case

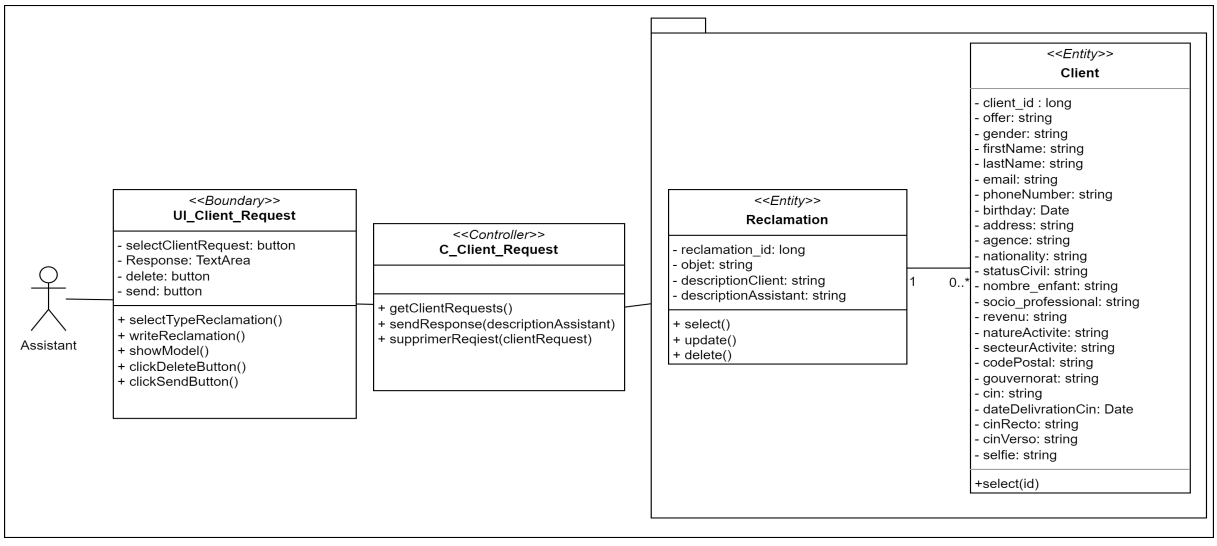


Figure 7.11: Class diagram for the « Manage Clients Request » use case.

Sequence Diagrams

Upon entering the interface, all requests issued by the client will be fetched. Administrators and assistants can respond or delete a request. when a response is sent, the client will receive it as a notification (figure 7.12).

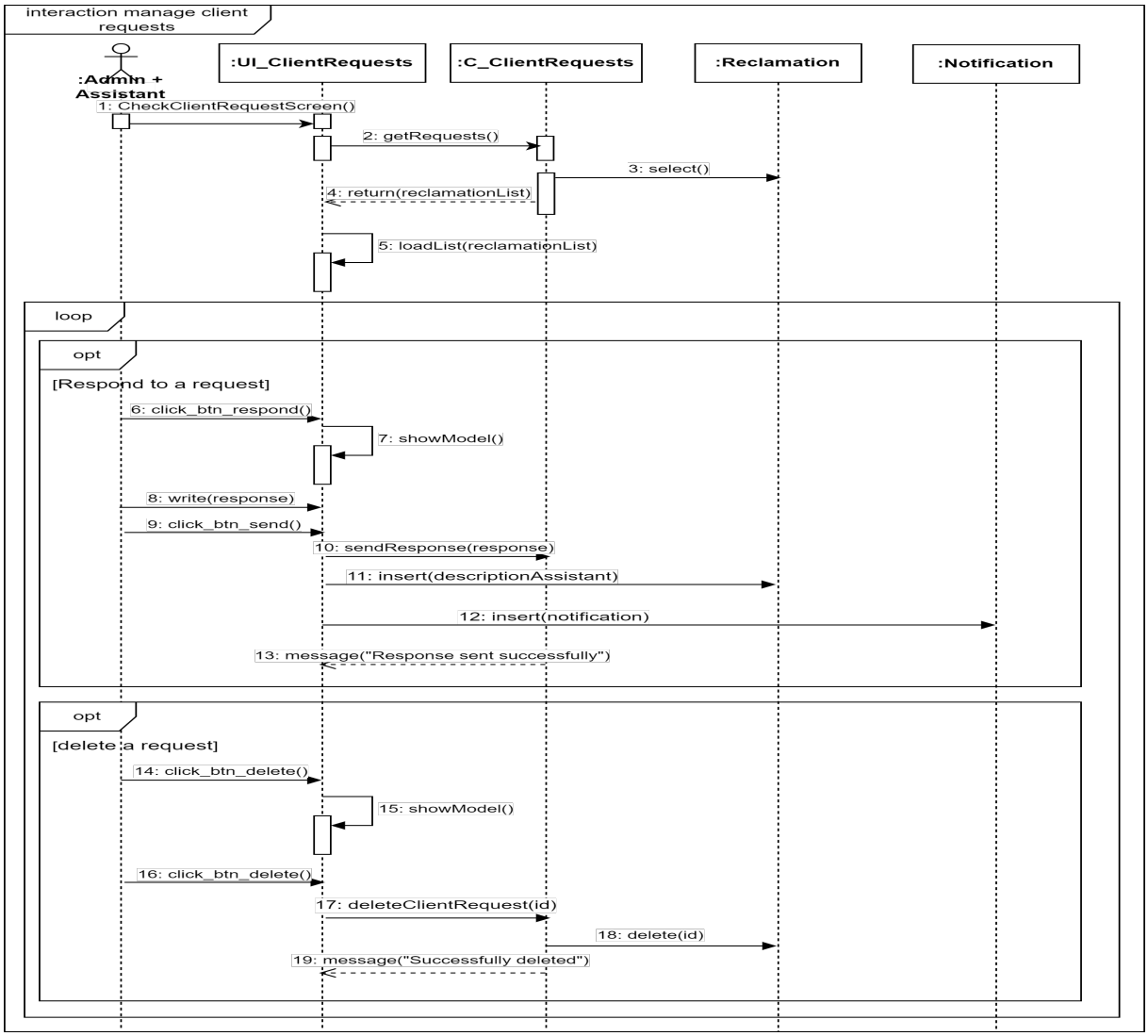


Figure 7.12: « Manage Clients Request » Sequence Diagram

7.5 Global Class Diagram

Figure 7.13 represents the global class diagram of the application. It envisions all the classes starting from sprint 1 and their relationship with each other giving a full generalized overview of the application.

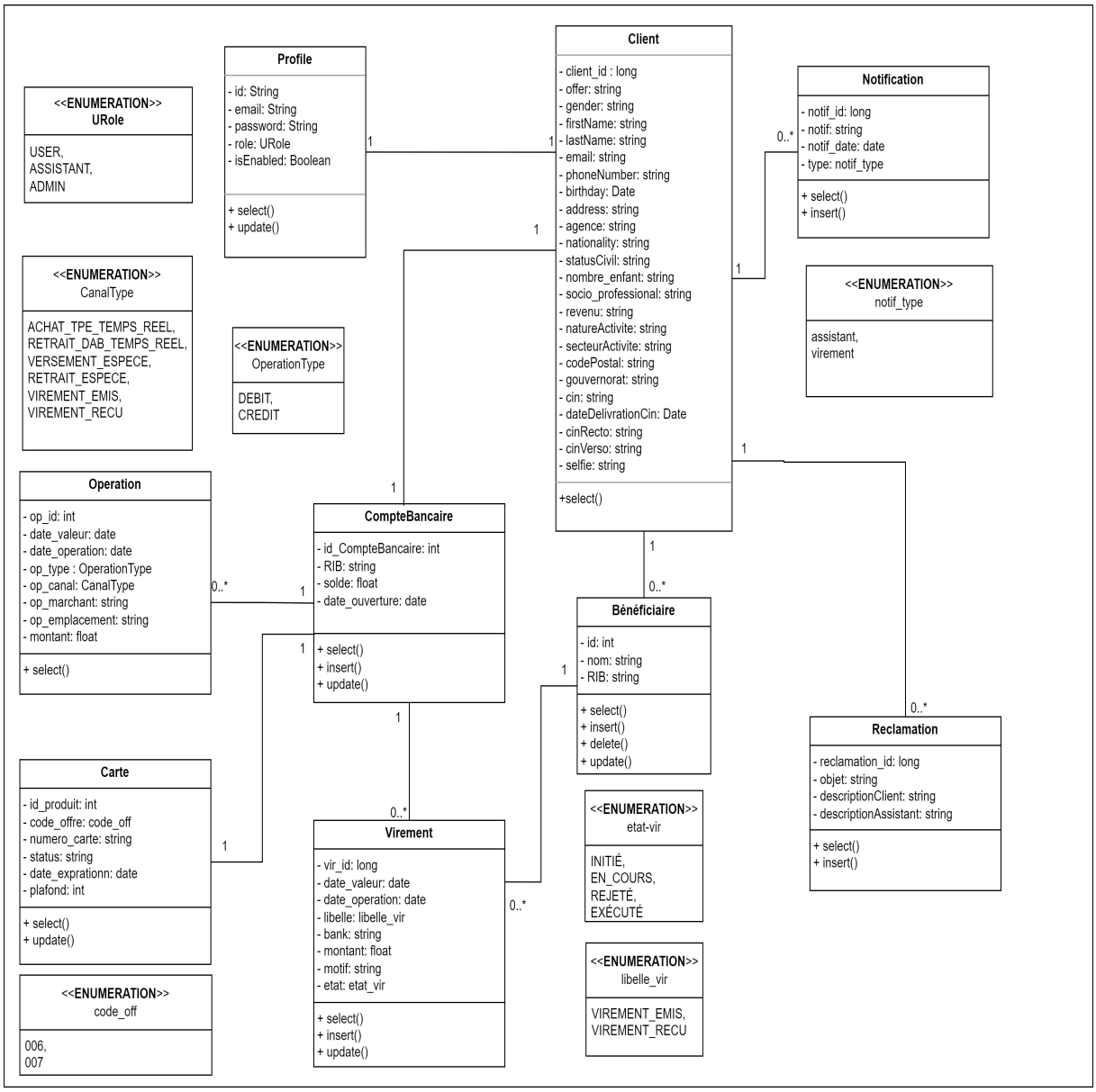


Figure 7.13: Global Class Diagram

7.6 Realisation

This segment will provide the implementation of sprint 4 use cases.

7.6.1 Realization of « Contact Assistants » Use Case

As shown in Figure 7.14, the client can send a claim to the bank assistant's by selecting the claim object and writing a small description then clicking on 'send'.

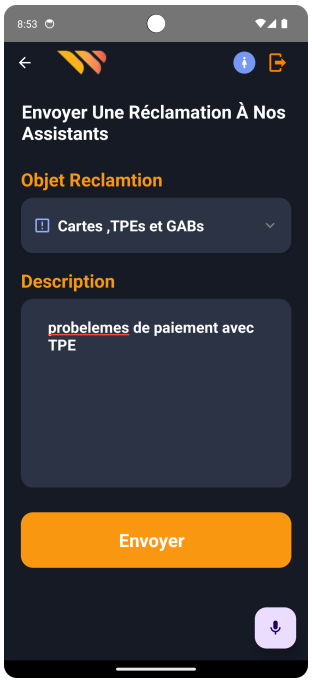


Figure 7.14: « Contact Assistants » use case realization

7.6.2 Realization of « Check Notifications » Use Case

The client can view all the delivered notifications through the interface shown in figure 7.15.

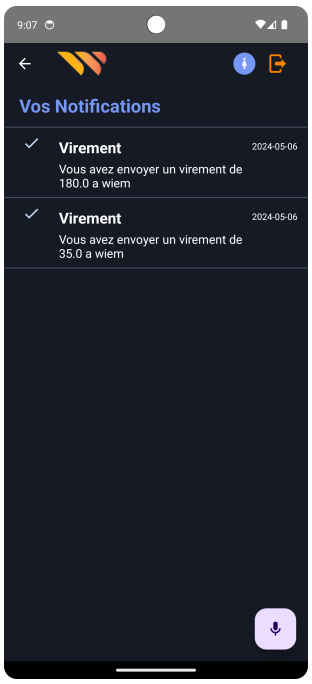


Figure 7.15: « Check Notifications » use case realization

7.6.3 Realization of « View dashboard » Use Case

The Admin and assistants can consult the state of the application and clients by accessing and viewing the dashboard depicted in figure 7.16, this dashboard gives insights about

key information that help in improving the quality of the services.

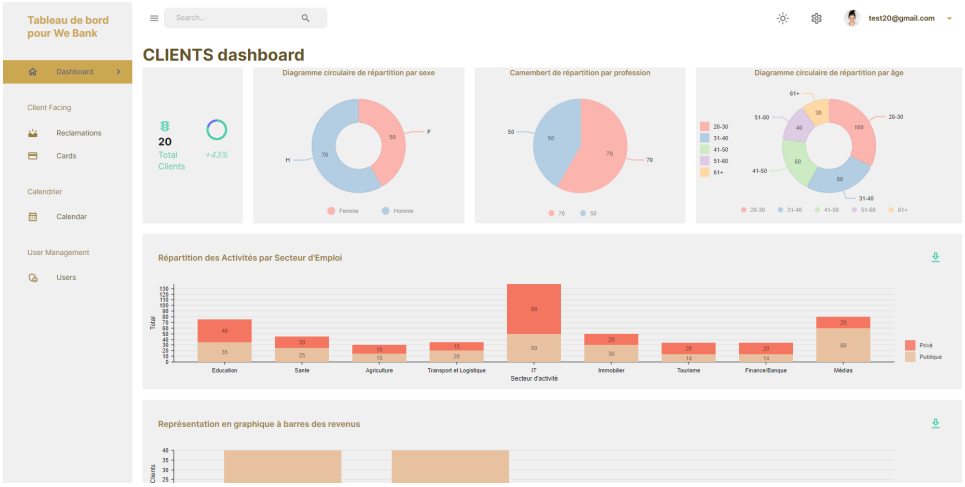


Figure 7.16: « View dashboard » use case realization

7.6.4 Realization of « Manage Assistants » Use Case

The Admin can manage assistants by adding, updating or deleting them, this can be done through the different buttons shown in figure 7.17.

Figure 7.17: « Manage Assistants » use case realization

7.6.5 Realization of « Manage Client's requests » Use Case

Figure 7.18 illustrates the interface that allows administrators and assistants manage client's requests by clicking on 'reply' or 'delete'.

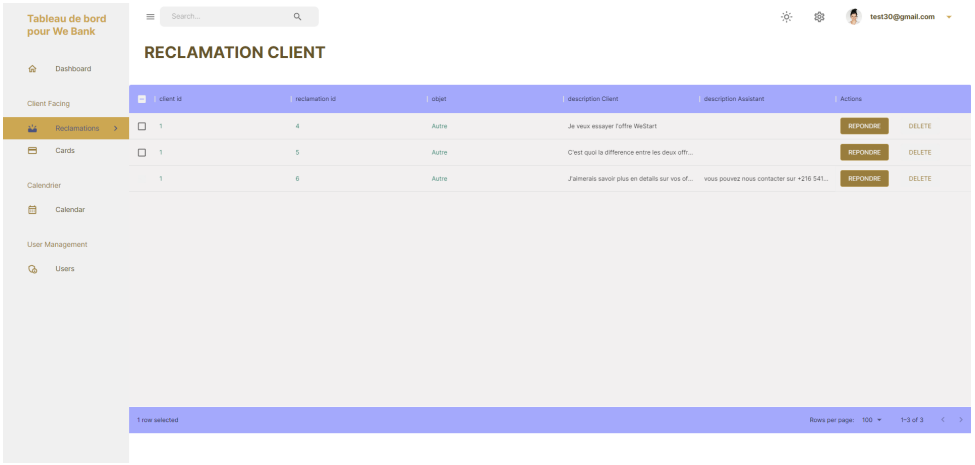


Figure 7.18: « Manage Client's requests » use case realization

7.7 Conclusion

This chapter was a walk-through of the last sprint 'Administration and Client Relations'. The use cases were refined, conceptualized, and a visualization of the interfaces were presented.

General Conclusion

Throughout the duration of our internship at Attijari Bank, we have realized multiple tasks, from research to execution, resulting in successfully building an accessible banking application. The aim of our project was mainly to improve the usability of the given existing application and in order to do this, a thorough research has to be done, which wasn't limited to papers or articles but it also included interacting and learning from people.

The execution was done through mobile and web development with the inclusion of machine learning this is to allow for a seamless and simple user interface and experience which is important in our case. In addition, for creating the virtual assistant, we have built a model based on BERT for intent classification. Lastly, we deployed it using Flask for its simple implementation.

This experience allowed us to better improve our development skills and more specifically we learned how to incorporate accessibility techniques so that the application becomes compatible with screen readers, moreover, we have learned how to work in a team, and better communicate.

On another hand, some areas could still be improved, such as adding other accessibility functionalities and using more tools or different approaches to improve the virtual assistant such as using Rasa.

Ultimately, we can add the latest accessibility and testing functionalities so that the application stays up-to-date.

Netography

[1] Attijari bank. attijaribank.com.tn. Consulted on March 10, 2024.

[2] Pandas. https://pandas.pydata.org/about/index.html. Consulted on March 24, 2024.

[3] The scrum guide. Scrum.org. Consulted on March 10, 2024.

[4] Intent detection module for a conversational assistant. https://dspace.cvut.cz/bitstream/handle/10467/100875/F3-DP-2022-Ozerova-Daria-masters_thesis.pdf. Consulted on April 29, 2024.

[5] Neural network. https://www.databricks.com/glossary/artificial-neural-network. Consulted on April 22, 2024.

[6] Understanding long short-term memory neural network. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Consulted on April 22, 2024.

[7] Attention is all you need. https://arxiv.org/pdf/1706.03762.pdf. Consulted on April 22, 2024.

[8] Agile. agilealliance.org/agile101/. Consulted on March 10, 2024.

[9] Illustrated guide to transformers- step by step explanation. https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0. Consulted on April 29, 2024.

[10] Bert explained: State of the art language model for nlp. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b62702. Consulted on April 29, 2024.

[11] Visual studio code. code.visualstudio.com/docs. Consulted on March 10, 2024.

[12] Intellij idea. https://www.jetbrains.com/idea/. Consulted on March 10, 2024.

[13] Postman. postman.com/company/about-postman/. Consulted on March 10, 2024.

[14] Draw.io. drawio.com/about. Consulted on March 10, 2024.

[15] React native. https://reactnative.dev/. Consulted on March 12, 2024.

[16] React. https://react.dev/. Consulted on March 12, 2024.

[17] Redux. https://redux.js.org/. Consulted on March 12, 2024.

[18] Material ui. https://mui.com/material-ui/getting-started/. Consulted on March 12, 2024.

[19] React native paper. https://reactnativepaper.com/. Consulted on March 12, 2024.

[20] Axios. https://axios-http.com/docs/intro. Consulted on March 15, 2024.

[21] Spring boot. https://spring.io/projects/spring-boot. Consulted on March 17, 2024.

[22] Json web token. https://spring.io/projects/spring-boot. Consulted on March 17, 2024.

[23] Postgresql. https://www.postgresql.org/docs/current/intro-whatis.html. Consulted on March 17, 2024.

[24] Docker. https://docs.docker.com/get-started/overview/. Consulted on March 17, 2024.

[25] Numpy. https://numpy.org/. Consulted on March 24, 2024.

[26] Hugging face. https://huggingface.co/docs/transformers/index. Consulted on March 24, 2024.

[27] Tensorflow. https://www.tensorflow.org/about. Consulted on March 24, 2024.

[28] Keras. https://keras.io/. Consulted on March 24, 2024.

Abstract

To ensure that the application reaches all its customers, Attijari Bank had decided to redesign their mobile application to include international accessibility guidelines and integrate a virtual assistant to assist people with disabilities. This decision not only improve their market reach but also strengthen inclusivity and diversity, reflecting the bank's commitment to serve all its customers equally.

The client can access various services provided by the bank, entirely online. All features of the app are compatible with screen readers additionally, the basic functionalities and services are accessible via voice commands by the virtual assistant.

Keywords: Accessibility, Virtual Assistant, Mobile Development, Web Development, Databases, Machine Learning.

Résumé

Attijari Banque a décidé de restructurer leur application WeBank en appliquant les mesures internationales de l'accessibilité et en intégrant une assistante virtuelle pour mieux assister les personnes avec des handicaps. Cette décision assure que l'application atteint tous les clients de la banque, tout en reflétant leur engagement à promouvoir l'inclusivité et la diversité. Le client a accès à plusieurs fonctionnalités et services offerts par la banque, totalement en ligne. Les fonctionnalités de l'application sont compatibles avec les lecteurs d'écran; en outre, les services de base sont accessibles en utilisant l'assistante virtuelle.

Mots-clés: Accessibilité, Assistante Virtuelle, Développement Mobile, Développement Web, Bases de Données, Machine Learning.