Tunisian Republic
Ministry of Higher Education
&
Scientific Research
--
Carthage University

Faculty of Sciences of Bizerte

IT Department

Graduation Project Report

To obtain a Bachelor's degree in Computer Science

Course : Software Engineering and Information System

Entitled:

INVOICING SYSTEM

Realized by:
Houssem Eddine Chibouni

Within:
Zedney Creative

Framed by:
Kawther Baccouch (Zedney Creative)
Mounir Ounissi (FSB)

Année Universitaire 2022-2023

Tunisian Republic
Ministry of Higher Education
&
Scientific Research
--
Carthage University

Frame reserved for the academic supervisor:

Année Universitaire 2022-2023

Appreciation

I would like to express my deepest gratitude and respect to Mr. Mounir Ounissi, my pedagogical supervisor and Professor at the Faculty of Sciences of Bizerte. Throughout the project, Mr. Ounissi provided me with attentive supervision, valuable feedback, and insightful guidance that proved invaluable in the development of my work. His unwavering support, enthusiasm, and personal and professional qualities have been an inspiration to me, and I feel fortunate to have had the opportunity to learn from him.

In addition, I am grateful to Mr. Hamda Maghroum and Fedi Djey, my professional supervisors at the Zedney Creative group, for welcoming me into their team and providing me with a supportive and encouraging environment.

Finally, I would like to express my appreciation to all the instructors of the computer science department at the Faculty of Science of Bizerte. Your dedication and commitment to teaching have played a vital role in my academic growth, and I am grateful for the knowledge and skills that I have gained through your courses and instructions.

Thank you all for your contributions to my education and professional development. I will carry the lessons and experiences that you have imparted to me throughout my career.

Table of Contents

Figures

Table

General Introduction

In this age of technology, digitalization is ubiquitous, and businesses are investing significant resources to streamline their processes and stay ahead of the competition. One area that requires particular attention is the management of invoices and payments. This critical function can be complex and intricate, and businesses must ensure they are organized and efficient to avoid costly mistakes.

Managing invoices and payments requires a keen eye for detail and adherence to legal and regulatory compliance. Businesses must ensure that all their payment processes comply with relevant laws and regulations to avoid legal and financial repercussions.

The recruitment and selection of the right payment management system is also crucial. Companies must identify payment solutions that align with their organizational goals, culture, and values. This process is vital in creating an efficient payment system and driving business growth maintaining a positive relationship with clients and vendors to avoid payment disputes and conflicts.

Efficient invoice and payment management is a critical component of any successful business, and it requires organizations to navigate complex challenges while ensuring prompt and accurate payment processing enabling companies to focus on core business functions and stay competitive.

1. Chapter: Working Context

1.1. Introduction

In this chapter, I will introduce the company Zedney Creative, and then we will present the problem and describe the project to be carried out. Finally, we will define the Scrum agile framework as a framework for carrying out our mission.

1.2. Presentation of the company

Zedney Creative is an IT engineering services company (SSII) founded in 2011 and present in Tunisia (Tunis and Bizerte), France (Paris), Emirates (Dubai) and Saudi Arabia (Riyadh), it offers IT solutions adapted to different businesses and industries.

The company is a partner of choice that provides its customers with digital solutions that are perfectly effective in meeting the challenges of agility, performance, and development.

The company specialize in the fields of information systems, IT development, mobile development, process automation and digitalization, provides solutions to meet the needs of companies, relying on a community of consultants who offer their technical expertise to groups in various fields.

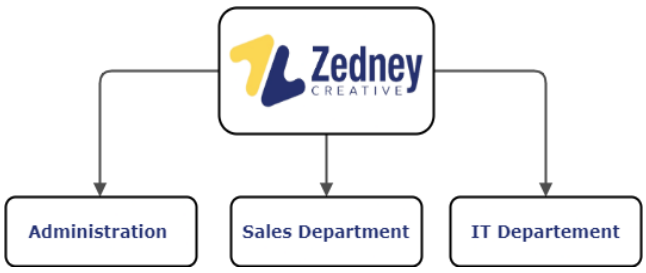


Figure 1 : Company organization chart

This figure shows the hierarchy within a subsidiary founded in Tunisia. There are 3 directions (IT, financial administration and sales) managed by a general management.

My internship takes place within the IT department.

1.3. Issue of the Project

1.3.1. Description of the problem

One of the major problems with invoice management is the potential for errors or inaccuracies. Invoices may contain incorrect information, such as incorrect pricing or quantities, which can lead to overpayment or underpayment. Additionally, invoices may be lost or misplaced, which can cause delays in payment and negatively impact relationships with vendors and clients.

Another challenge with invoice management is the sheer volume of invoices that many businesses receive, managing a high volume of invoices can be time-consuming and require a significant number of resources. This can be particularly challenging for small businesses or those with limited staff and resources. It can also be complicated by the different payment terms and methods used by vendors. Some may require payment by a certain date or using a specific payment method, while others may be more flexible so to keep track of these different requirements and ensuring timely payment will be a great challenge.

A Digital solution can help to streamline and automate many aspects of invoice management, reducing the potential for errors and improving efficiency. the

solution can automate approval workflows, enable integration with accounting systems, customize payment terms and methods...

Developing software for invoices, products and client's management can be a complex and challenging process, we are currently facing hardships concerning data accuracy, integration with other systems and security concerns, therefore the plan is to leverage an existing platform despite its issues and undertake a comprehensive effort to enhance its functionality. Specifically, we aim to improve the core functions of the platform and its underlying architecture to create a more efficient and user-friendly experience.

1.3.2. Proposed Solution

In pursuit of our objective to enhance the platform, we will carefully analyze the existing issues and devise a strategy to address them. By focusing on the core functions that require improvement, we will implement modifications to optimize performance and user experience, we are planning to substitute the architecture of the platform to ensure that it is scalable, flexible, and sustainable for future growth

Through our efforts, we aim to exceeds user expectations and delivers value to our stakeholders by delivering a significantly improved system that ensures:

- ✓ An intuitive and user-friendly dashboard that provides easy access to key information and features, such as invoices, taxes, contacts, and clients.

- ✓ Improved functionality and core features, including the ability to create, manage, and track invoices; calculate taxes automatically; and easily add and manage client and contact information.

- ✓ More efficient and streamlined processes that reduce the time and effort required to create and manage invoices and generate documents.

- ✓ Robust security and data protection measures to safeguard sensitive information, such as client and payment details, and prevent unauthorized access or data breaches.

- ✓ Increased stakeholder engagement and support, by providing timely and accurate information, regular updates, and excellent customer service and support.

1.4. Used Technologies:

1.4.1. Objectives & Goals

Zedney Creative currently possesses a copy of the application that was built by a French team. The development of this application utilized Symfony 4, Vue.js 3, and MySQL as the underlying technologies.

The ongoing migration process entails a shift from the old technologies to adopting the React JavaScript library for front-end development, Django framework for back-end development, and the PostgreSQL database for data storage. This transition involves transferring existing code, restructuring the application architecture, and re-implementing functionality using the new technology stack.

The goal is to leverage the benefits and features offered by React, Django, and PostgreSQL, such as enhanced user experience, improved development efficiency, scalability and security. Throughout the migration, careful planning, testing, and data migration procedures are employed to ensure a smooth and successful transition to the new technology stack.

1.4.2. Previous Technologies:

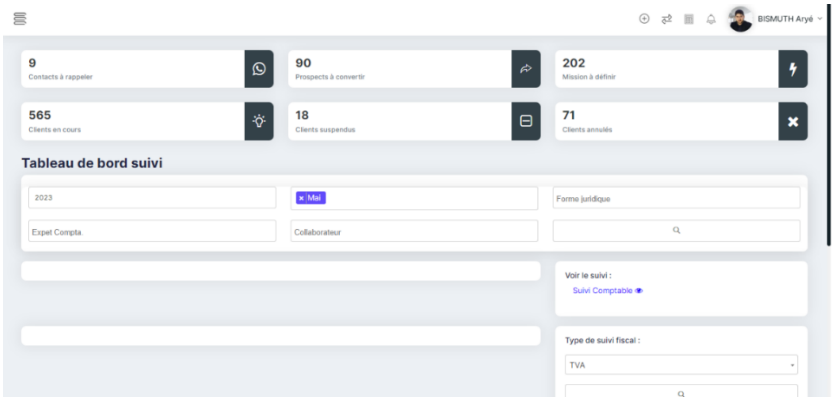


Figure 2 : Screenshot 1

The current dashboard is experiencing functionality issues and is lacking charts/graphs. It requires attention and improvements to ensure proper functionality and include the necessary visual representations of data through charts.

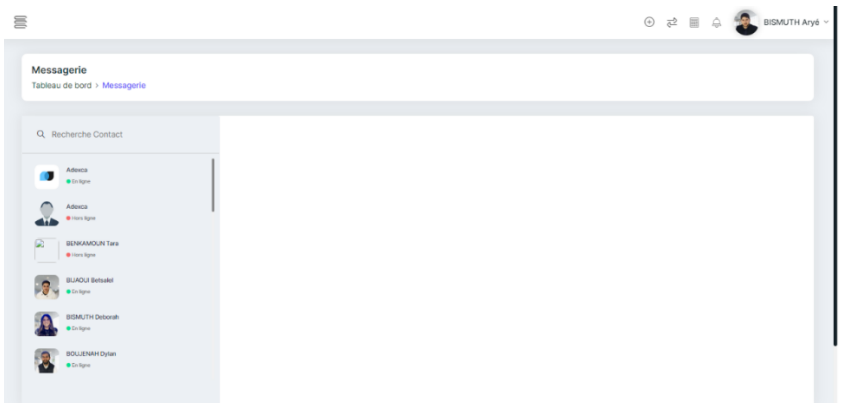


Figure 3 : Screenshot 2

The chat display is ambiguous as it incorrectly shows connected users who are not actually online or actively participating in the chat.

Additionally, the chat functionality is incomplete and not fully operational. There are certain features or capabilities that are missing or not working as intended

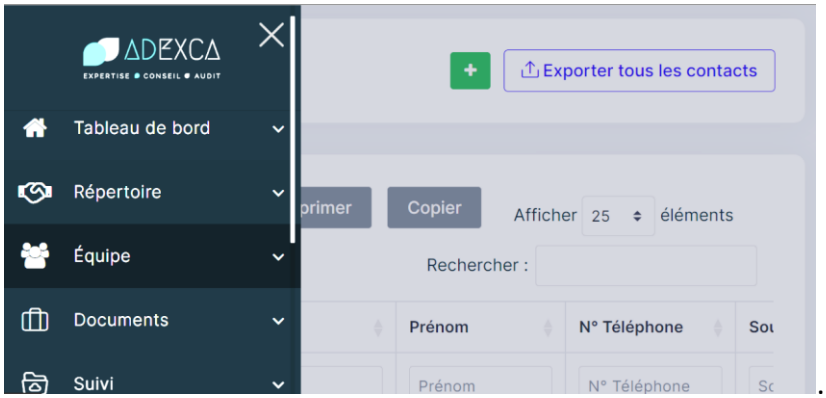


Figure 4 : Screenshot 3

The current data display lacks intuitiveness and ergonomic design, making it difficult for users to fully control and monitor the data effectively. It is crucial to improve the user interface and data presentation to enhance usability and provide a more user-friendly experience.

1.4.3. Alternatives

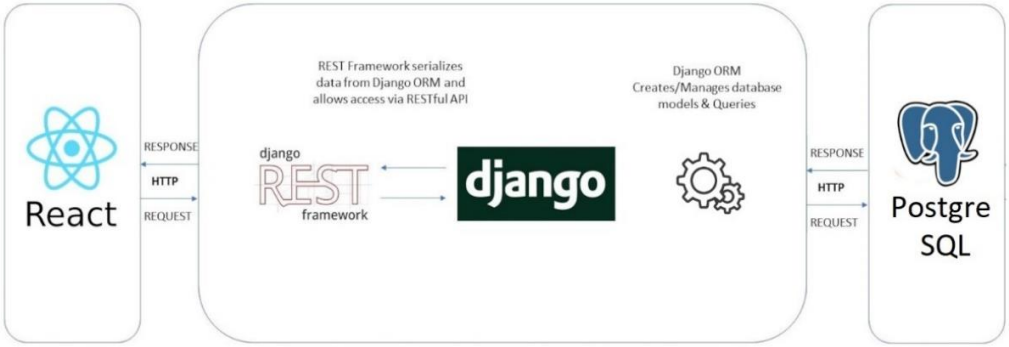


Figure 5 : Project Architecture

Python Vs Symfony: Backend technologies such as Django and PHP are widely utilized for developing the backend components of websites. The backend is a crucial aspect of any website, The evolution of web development leans towards Django as it offers a higher level of versatility, although Django combines the simplicity and elegance of the Python programming language with its extensive range of packages and libraries. This boost will empower the project to be built robust, rich and efficient.

The Python ecosystem provides a wide selection of pre-built modules, making it easier to handle diverse functionalities such as authentication, database management, and API integrations.

Django's flexibility will allow us to adapt modifications in the project requirements and use the extensive Python community for support and collaboration.

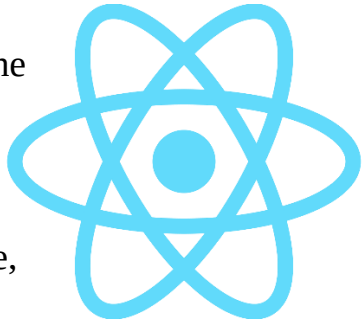
By Using Django Rest Framework (DRF) which is a powerful and popular framework for building Web APIs in Django we will be empowered with a set of tools and utilities that simplify the process of creating RESTful APIs.

React Vs Vue/Twig: Twig and Vue are commonly used in server-side rendering scenarios, where the templates are rendered on the server and then sent to the client.

This approach can provide benefits such as improved SEO and initial page load performance. However, it may have limitations in terms of interactivity compared to client-side rendering frameworks like React.

I chose to migrate towards React for its scalability, reusability, and modern approach which is a great decision in my personal opinion.

React's component-based architecture allows for the development of scalable and modular applications. By breaking the user interface into reusable components, **React** enables efficient development and maintenance, saving time and effort in the long run also

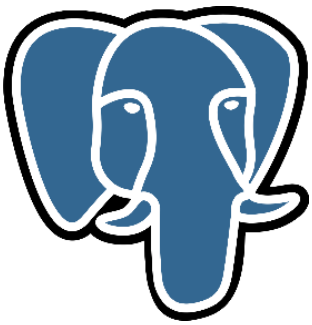


React's virtual DOM and efficient rendering mechanism contribute to its scalability, allowing for smooth performance even with complex and dynamic UIs, additionally, React's ecosystem offers a wealth of libraries and tools that enhance development productivity and provide solutions for various challenges.

Unlike Django Template engine, Vue is known for its ability to handle templates directly through One File Components, however, it is important to note that this approach may require making numerous Symfony API calls. While some developers find success with this setup, others may have a less positive experience and encounter challenges.

When utilizing Vue with Symfony, the heavy reliance on Symfony API calls within Vue templates can lead to suboptimal results. It can introduce complexities and potential performance issues, especially as my project scales or the number of API calls of the different entities increases.

MySQL Vs Postgres: While MySQL offers a straightforward setup and administration process, making it ideal for beginners or those with less complex database requirements.



PostgreSQL is renowned for its advanced features, extensibility, and strict adherence to data integrity. It offers robust support for complex queries, advanced data types, and custom functions, making it suitable for enterprise-level applications and projects that require sophisticated database functionality.

Managing a project encompassing around 80 entities demands meticulous organization and effective documentation. To navigate the complexities efficiently, it was so crucial to establish a systematic approach to replicate the project efficiently.

1.4.4. Architecture

I used The Model/View/Controller (MVC) architecture as a way of organizing an interface graph of a program, consisting in distinguishing three distinct entities which are, the model, the view and the controller each having a specific role in the interface.

- ✓ **Model**: a kernel of the application which manages the data, makes it possible to retrieve the information in the database.

- ✓ **View**: graphic component of the interface which allows to present the data of the model to the user.

- ✓ **Controller**: a component responsible for decision-making, manages the business logic, it acts as the intermediary between the model and the view.

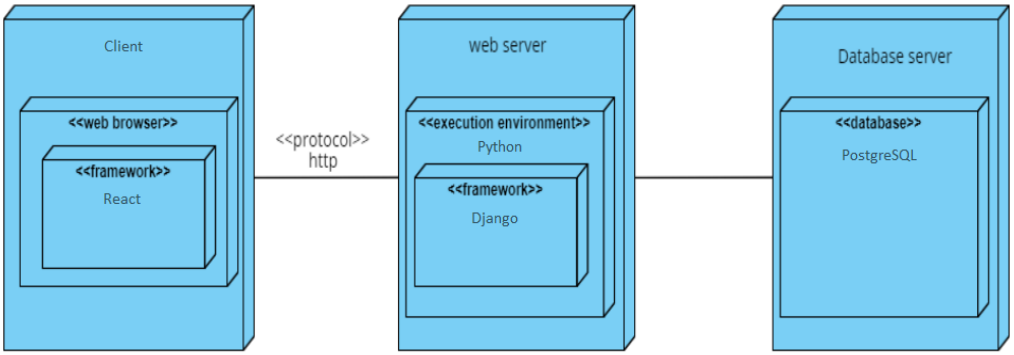


Figure 6 : Deployment Diagram

1.5. Used Methodologies

1.5.1. Project management approach

I opted for an agile approach to project management based on the comprehensive indicators. This decision aims to achieve two main goals: first, to successfully implement as many requested functionalities as possible, and second, to effectively adapt to changes and emerging needs.

Agile software development is a way of organizing the development process, emphasizing direct and frequent communication – preferably face-to-face, frequent

deliveries of working software increments, short iterations, active customer engagement throughout the whole development life-cycle and change responsiveness rather than change avoidance. This can be seen as a contrast to waterfall-like processes which emphasize thorough and detailed planning and design upfront and consecutive plan conformance. [1]

Agile is a project management and software development approach that operates in iterations, enabling teams to deliver value to customers more efficiently and with fewer challenges. Rather than relying on a single, extensive launch, an agile approach delivers work in smaller, more manageable increments that are readily usable by stakeholders.

While agile concepts and tools can be traced back to IT, the application of agile practices has expanded to various industries beyond the realm of IT. Today, agile methodologies are available and utilized in diverse fields, ranging from innovative services to research and development in industries. Some notable agile methods include Scrum, XP (Extreme Programming), RAD (Rapid Application Development), and DSDM (Dynamic Systems Development Method).

1.5.2. Presentation of the used framework

After reviewing the characteristics of the most used agile methods I decided to choose the Scrum method. In fact, scrum indicates that the size of the team can be reduced and this is my case; another reason is that scrum is flexible in terms of the duration of the sprint (between 2 and 4 weeks).

Scrum is a framework for developing and sustaining complex products. Ken Schwaber and Jeff Sutherland developed Scrum; the Scrum Guide is written and provided by them. Together, they stand behind the Scrum Guide. [2]

1.6. Presentation and application of Scrum

1.6.1. Introduction

I will present the product backlog and the list of actors acting in this project. I will also present the definition of done, one of the artifacts of the Scrum method.

1.6.2. The development Team

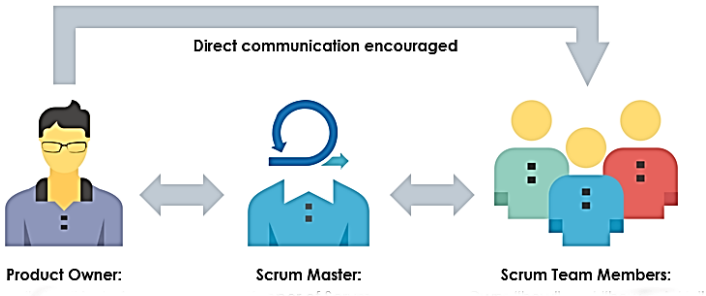


Figure 7 : Scrum Actors

In our development team, Mr. Ounissi serves as the product backlog manager, diligently curating and organizing the list of features and tasks essential to our project's success.

Working closely with him, Mr. Hamda Maghroum takes on the role of the product owner, acting as the bridge between the customer and our team.

With a deep understanding of their needs and aspirations, Mr. Hamda ensures that development efforts align with the product vision. I relied on Mr. Ounissi's

expertise in backlog management and collaborate closely with Hamda to prioritize tasks and refine requirements.

Together, we embraced an agile approach, constantly adapting to changing demands and delivering value. Through the collective efforts of Mr. Ounissi, Mr. Hamda Maghroum, and my dedicated work, we strived to create a successful and impactful product.

1.6.3. Done

"Done" are the acceptance criteria in the Product Backlog. Specifically, in this project, "done" consists of:

- ✓ Successfully completing the UML [3] diagrams accurately and comprehensively.

- ✓ Successfully completing the validation stage of functionalities.

1.6.4. Le Time Box

The Time box of a sprint is defined by the team and does not change. During the project; it allows to define an estimate of the duration of each sprint and to produce potentially deliverable increments with the most business value, particularly, in our project the time box is 2 weeks.

1.6.5. Why Scrum

The choice of Scrum for our project was based on its strength. It can be described as a flexible method and its good for its responsiveness to our needs for improving the outdated application.

A great ability to adapt to change thanks to short iterations. The most important thing is that Scrum brings together both theoretical and practical sides and comes very close to reality.

1.6.6. Modeling and design method

The success or failure of software development largely refers to the modeling phase. Before blindly attacking the code, the modeling of the system greatly facilitates its implementation and eliminates the risk of shipwreck of the project The UML [3] is a graphical computer modeling language which allows to popularize the aspects related to the design and the architecture, specific to the software, to the customer. Also, it provides a quick understanding of the program to other external developers in case of recovery of the software and facilitates its maintenance.

1.6.7. Product backlog

A product backlog is a list of the new features, changes to existing features, bug fixes, infrastructure changes or other activities that a team may deliver in order to achieve a specific outcome. The product backlog is the single authoritative source

for things that a team works on, the study of the existing allowed me to identify the services and the needs of the client.

To meet project requirements efficiently, the identified needs were organized and prioritized based on their importance. This process involved grouping and sorting the needs according to their relative priority, once prioritized, the needs were then distributed across the various sprints.

Figure 8 : Scrum Values

1.7. Conclusion

In conclusion of this preliminary study, the improvement of functioning within the company directs us towards the upgrading and fixing of the invoice project, among the first steps for the fulfilling our task, it is essential to define the different functionalities and to create the product backlog.

2. Chapter: Product Backlog Planning

2.1. Introduction:

This chapter focuses on determining the functionality of each user within the project, identifying different user types and actors, and creating a comprehensive product backlog. The functionality analysis and user categorization are crucial for understanding user needs, while the product backlog serves as a prioritized list of features and tasks for development.

2.2. Identification of User Profiles

In our system, there are two distinct roles that individuals can assume: an admin of the cabinet or a collaborator who's working to contribute in the management tasks.

The admin of the cabinet holds a position of authority and responsibility, entrusted with overseeing the overall operations, decision-making processes, and strategic direction of the system. They are responsible for managing resources, coordinating activities, and ensuring effective communication within the organization.

On the other hand, the collaborator plays a crucial role in our application, actively contributing to its development and growth. They work in tandem with the admin, providing valuable input, executing tasks, and actively participating in collaborative efforts. Collaborators bring their expertise, skills, and ideas to the table, making valuable contributions to the application's progress.

Both users can perform a large set of tasks based on the tab they choose:

- ✓ The user can perceive an overview of cabinet's financial activities.
- ✓ The user can manage and organize the cabinet contacts.
- ✓ The user can store and manage the customers information's.
- ✓ The user can track and manage prospects which they are more assured contacts and business opportunities.
- ✓ The user can manage the "Facturation" rates applicable to the cabinet's products or services.
- ✓ The user can manage articles and products.
- ✓ The user can group items by categories for more efficient management.
- ✓ The user can manage the information and permissions of collaborators.
- ✓ The user can witness the social aspects of cabinet's business, such as payments and benefits.
- ✓ The user can easily track the cabinet's tax obligations and pay taxes.

- ✓ The user can manage documents such as legal sheets, couriers and internal document.

- ✓ The user can communicate with administrators and other collaborators via messages.

2.3. Non-functional needs

Non-functional needs are needs that have a visible aspect for the user, but which are not directly related to the behavior of the system.

The goal is to create a versatile solution that is both high-performing and durable. Merely focusing on functionality and operations does not ensure user satisfaction and loyalty. Therefore, I had to consider non-functional criteria during the design and implementation of the solution. Some of these requirements include:

2.3.1. Performance

The application should provide quick response times to ensure a comfortable and user-friendly experience, A high-performance invoice system would typically exhibit the following characteristics:

- Responsiveness: The system should have low response times, ensuring that users can quickly access and interact with invoices without experiencing delays or lags.

- Processing Speed: The system should process invoices swiftly, including tasks such as generating invoices, calculating totals, applying discounts, and updating records. This allows for smooth and efficient invoicing processes.

- Reliability: The system should consistently perform its tasks accurately and reliably, reducing the chances of errors or discrepancies in invoice generation, calculations, or data storage.

2.3.2. Modularity

It should be easy to add new services or components without significant changes to the existing structure.

- Functional Modularity: the invoice system is divided into separate functional components or modules based on their specific tasks or responsibilities. Each module focuses on a specific function, promoting a clear separation of concerns and facilitating easier development, testing, and maintenance.

- Data Modularity: Data modularity involves organizing and structuring the data used in the invoice system. Each entity has its own set of attributes and relationships, and the data is stored and managed separately for each entity. This approach allows for efficient data retrieval, manipulation, and integrity.

- Service-Based Modularity: the invoice system is built as a collection of loosely coupled services that communicate with each other through well-defined interfaces. Each service represents a specific functionality, such as invoice generation, payment processing and reporting.

2.3.3. Maintainability:

The code should be well-commented, easily maintainable, and built upon established "best practices.

- Code Readability and Documentation: Writing clean, well-structured, and self-explanatory code is essential for maintainability.

- Testability: Incorporating automated testing practices, such as unit testing and integration testing, improves the maintainability of the invoice system. A comprehensive test suite allows for confident refactoring and modification of the codebase.

- Error Handling and Logging: Implement robust error handling mechanisms in the invoice system to gracefully handle exceptions and error scenarios. Proper logging of errors, warnings, and information can assist in diagnosing issues and providing insights into system behavior during maintenance and debugging.

- Version Control: [4] Utilize a version control system to manage code changes, track modifications, and collaborate with other developers effectively.

- Dependency Management: Keep track of the external libraries, frameworks, or dependencies used in the invoice system. Regularly update and manage these dependencies to benefit from bug fixes, performance improvements, and security patches.

- Adherence to Best Practices and Standards: Following established software development best practices and industry standards enhances the maintainability of the invoice system.

2.4. User Story & Technical Story

Every feature request is transformed into a concise narrative or short story. The Product Backlog consists of User Stories (US) and Technical Stories (TS), prioritized based on their business value. These stories essentially convey:

- ✓ The intended recipient of the feature (the end user)
- ✓ What the user wishes to accomplish
- ✓ The underlying motivation behind their desire to do so

It is confirmed by acceptance criteria written at the same time as the story and expressed in this way: As a **<role>**, I want to **<do something>** to achieve **<business value>**

2.5. Implementation of Product Backlog

The table below presents these needs which will be the subject of our work.

ID	Type of Story	Story	Estimated Days	Priority	Sprint
1	TS1	As a Developer I want to Adapt and familiarize myself with the outdated project technologies in order to become proficient with them.	14	9	1
2	TS2	As a Developer I want to transfer the previous models from the outdated database and ensuring all necessary actions are carried out accordingly.	6	7	2
3	US1	As an Administrator I can access an intuitive interface and modify my personal informations.	8	9	2
4	TS3	As a Developer I want to strategically organize the components within the forms to promote efficient and effective organization.	7	8	3
5	US2	As administrator I can securely log in to my account and securely access my information.	2	6	3
6	US3	As an administrator I can securely modify my personal information, which includes updating my profile picture, email address, and password.	1	5	3
7	US4	As an administrator I can include users, referred to as collaborators, and assign them various roles within the enterprise, as well as the ability to modify them when needed.	4	7	3
8	US5	As an Administrator I can effectively manage and add contacts, clients, and prospects within the system.	7	9	4
9	US6	As a User I can handle a variety of documents, including letters and couriers, legal documents, and internal documents.	7	7	4
10	US7	As a User I can effectively handle tax management, invoice processing.	14	8	5

Table 1 : Product Backlog

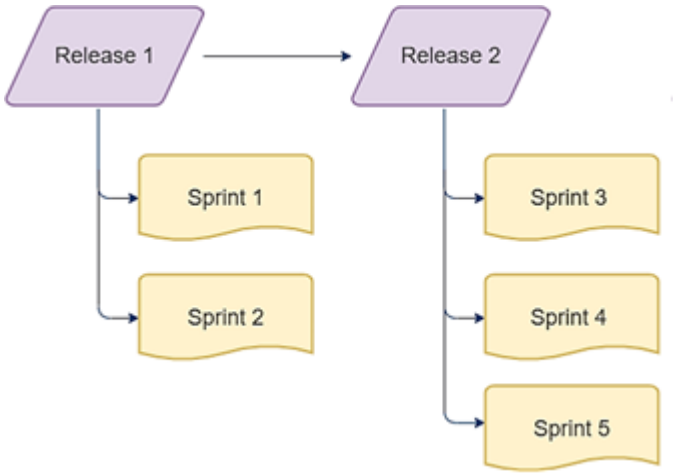


Figure 9: Organization of releases

We have chosen to divide the Sprints that we have identified into three Releases presenting as follows:

By organizing the sprints into these releases, we can effectively plan and prioritize the development and delivery of features, ensuring a systematic and structured approach to the project while assuring the completion of the project in time.

2.6. Conclusion

In conclusion, the product backlog is a vital component of the agile development process for the invoice system.

It serves as a comprehensive repository of user stories and technical stories, prioritized based on their business value. The backlog captures the desired features, functionalities, and improvements that are ready to be implemented throughout the development lifecycle.

3. Chapter: Release 1

3.1. Introduction:

This chapter will include the backlog, the different diagrams and the description for the realization of the first release with its graphical interfaces.

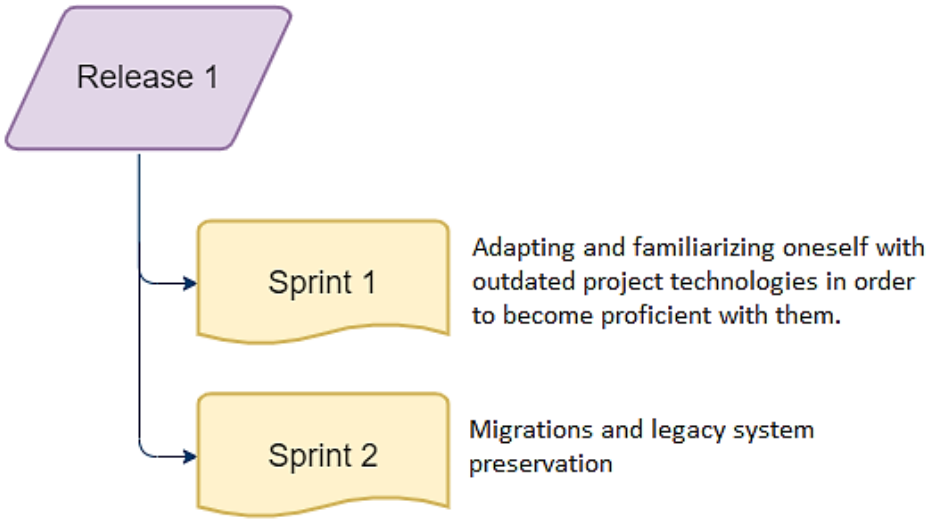


Figure 10 : Release 1

3.2. Sprint 1: « Adaptation to the old Technologies »

3.2.1. Objectif

The goal of this Sprint is to familiarize ourselves with the technologies utilized in the old project namely **Symfony**, **Vue.js**, **Twig**, and **MySQL**. The primary goal is to understand the interactions of these technologies and their implementation within the existing codebase, enabling us to gain a thorough understanding of the existing codebase.

By dedicating this Sprint to technology familiarization, we aim to assess the current state of the codebase, identify any dependencies or limitations, establishing

a solid foundation of knowledge that will facilitate future development tasks and decision-making.

By understanding the intricacies of the old project's technologies, we will be better equipped to plan and execute subsequent development Sprints effectively, this familiarity will allow us to make informed decisions regarding code refactoring, improvements, and potential enhancements, ensuring a smooth transition to the new app while preserving the invoices and clients' essential functionalities.

3.2.2. Sprint Backlog

Story ID	Task ID	Tasks	Estimated Days	Priority
1	1	Learning about the necessary technologies and gaining familiarity with php composer.	8	9
1	2	Installation of packages while disregarding any deprecated dependencies.	6	7

Table 2 : Sprint 1 Backlog

3.2.3. Sprint Analysis

3.2.3.1. Task 1.1: Discovery

- ✓ *Composer*

Composer is a PHP tool that facilitates dependency management within your projects. It provides a straightforward way to declare the libraries or packages that your project relies on, and Composer takes care of installing or updating them accordingly. [5]

It's important to note that Composer operates differently from traditional package managers like Yum or Apt.

While it handles packages, it focuses on managing dependencies on a per-project basis.

This means that Composer installs the libraries specifically for your project, storing them in a designated directory such as "vendor."

- ✓ *FOSUserBundle*

The **FOSUserBundle** [6] adds support for a database-backed user system in Symfony2+. It provides a flexible framework for user management that aims to handle common tasks such as user registration and password retrieval, its features include:

- Users can be stored via Doctrine ORM or MongoDB/CouchDB ODM
- Registration support, with an optional confirmation per email
- Password reset support
- Unit tested

The bundle offers pre-mapped base classes that simplify entity creation by providing default mappings for common fields. The previous Dev Team performed:

- The inheritance from the BaseUser class found in the Model folder.

- The Mapping of the id field, ensuring that it would be marked as protected since it is inherited from the parent class.

✓ *Vue.Js Rendering with Twig*

Vue can take charge of the frontend and seamlessly communicate with the rendered DOM by Twig in order to ensure the dynamic behavior to the page using the Vue application, it is crucial that the client-side rendered DOM and the server-side rendered DOM are in synchronous.

This approach also feasible to utilize Vue variables in conjunction with Twig's, though this require a minor configuration adjustment and this configuration can either be handled individually for each Vue instance or set up globally.

This solution has several drawbacks, summarized as follows:

- The lack of the ability to highlight or navigate to Vue variables directly within the Twig template.

- When including another Twig template containing additional Vue components, the variables and methods of the deeper components may not function correctly within the given component. The parent component may intercept the logic calls, causing conflicts or unexpected behavior.

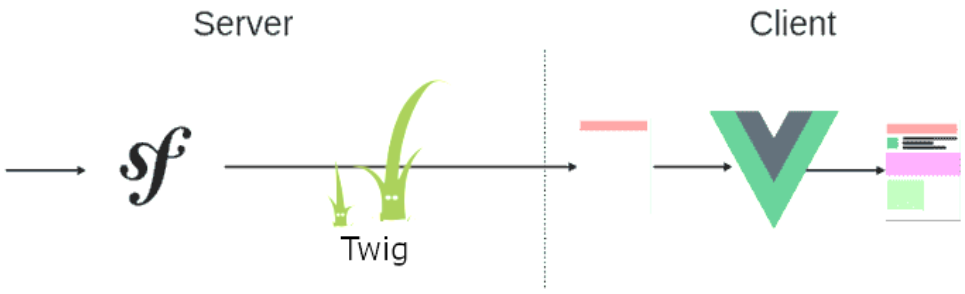


Figure 11 : Symfony & Vue Communication

✓ *Doctrine ORM:*

Symfony provides all the tools you need to use databases in your applications thanks to Doctrine, the best set of PHP libraries to work with databases. These tools support relational databases like MySQL and PostgreSQL and also NoSQL databases like MongoDB. [7]

Doctrine ORM simplifies and streamlines the process of working with databases by mapping database tables to PHP objects and vice versa.

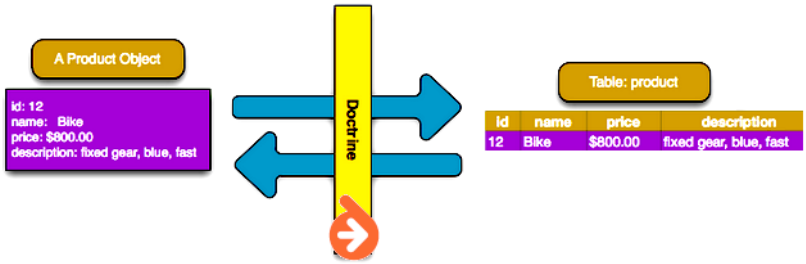


Figure 12 : Doctrine Object Mapping

✓ *GOSWebSocketBundle*

The **GosWebSocketBundle** [8] is a Symfony bundle that brings together WebSocket functionality in a user-friendly application architecture.

The bundle follows an event-driven architecture, allowing the definition of event listeners and handlers to process WebSocket messages and perform specific actions.

This approach promotes decoupling and enables the development of scalable and high-performance WebSocket applications.

3.2.3.2. Task 1.2: Dependencies

By employing the composer dependency manager, we gained immediate access to packages, frameworks, and various components to install the missing requirements.

This eliminates the need to download them individually except for one package called "**DOMPDF"** [9] which a CSS 2.1 compliant HTML layout and rendering engine written in PHP. It is a style-driven renderer: it will download and read external stylesheets, inline style tags, and the style attributes of individual HTML elements. It also supports most presentational HTML attributes.

When installing packages in Symfony while disregarding any deprecated dependencies, it involves intentionally ignoring warnings or deprecation notices related to outdated or deprecated features or dependencies within the packages being installed.

When we choose to disregard deprecated dependencies, we are essentially opting to continue using outdated components without making the necessary updates or adjustments to align with the latest best practices or recommendations.

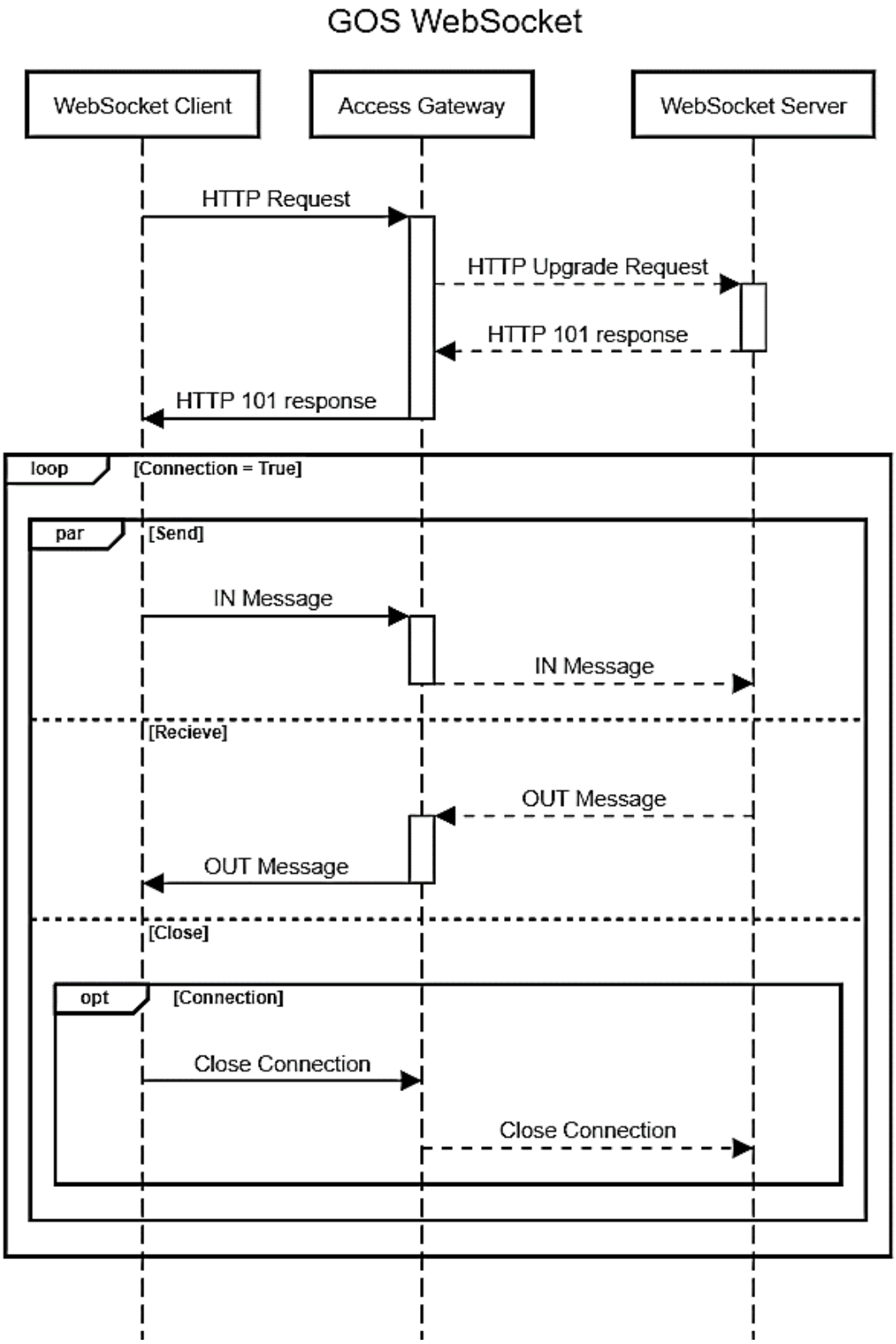


Figure 13 : GOS Web Socket

3.2.4. Sprint Implementation:

```
composer install --ignore-platform-reqs
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
Package operations: 30 installs, 0 updates, 0 removals
  - Installing topthink/think-installer (v1.0.12): Downloading (100%)
  - Installing symfony/options-resolver (v3.4.39): Downloading (100%)
  - Installing endroid/qr-code (1.9.3): Downloading (100%)
  - Installing guzzlehttp/promises (v1.3.1): Downloading (100%)
  - Installing ralouphie/getallheaders (3.0.3): Downloading (100%)
  - Installing psr/http-message (1.0.1): Downloading (100%)
  - Installing guzzlehttp/psr7 (1.6.1): Downloading (100%)
  - Installing karsonzhang/fastadmin-addons (1.1.10): Downloading (100%)
  - Installing mtdowling/cron-expression (v1.2.3): Loading from cache
  - Installing overtrue/pinyin (3.0.6): Downloading (100%)
```

Figure 14 : Composer Installation

```
 DONE  Compiled successfully in 6447ms

No type errors found
Version: typescript 3.0.1
Time: 4223ms

  App running at:
  - Local:   http://localhost:8080/
  - Network: http://172.16.110.227:8080/

  Note that the development build is not optimized.
  To create a production build, run npm run build.
```

Figure 15 : Building Vue.Js for production

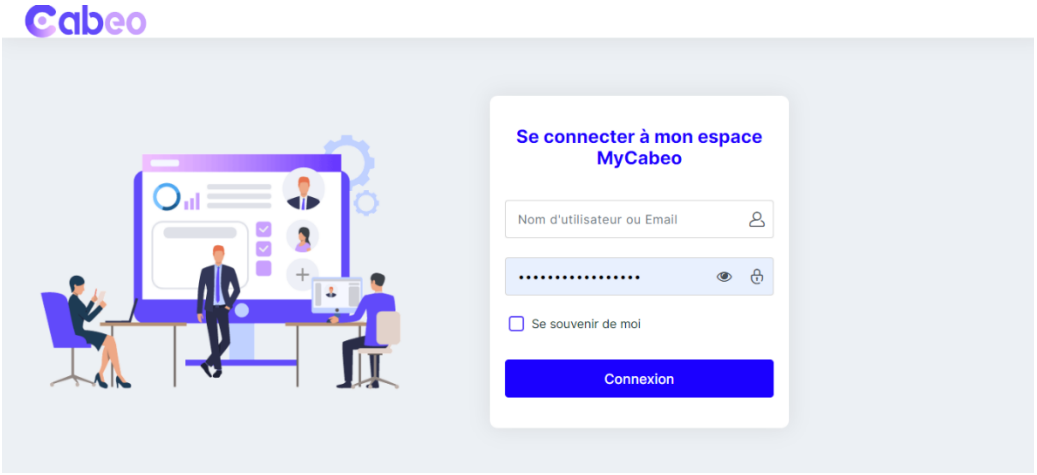


Figure 16 : Login page of the old build

3.2.5. Sprint Review

While these technologies and bundles provide powerful tools for enhancing Symfony applications, it's essential to carefully consider their trade-offs and potential disadvantages. Proper planning and consideration of project replacements were crucial to mitigate these challenges and ensure successful implementation.

3.2.6. Sprint Retrospective

The retrospective is a meeting during which the Scrum team uses its experience over the past sprint to improve its organization in order to be more efficient.

What worked well	What can be improved
The coherence of the notes we took concerning the old technologies	Time Management

Table 3 : Sprint 1 improvement plan

3.3. Sprint 2: « Migration & legacy System Preservation»

3.3.1. Objectif

The migration process is all about the choices of useful strategies such as data migration, code refactoring, and extracting scripts that will facilitate the transition while preserving the integrity of the legacy system.

This sprint describes the profits we gained by extracting and transforming data and methods from the legacy system into a new compatible and useful resources aiming to help with the development of the target system.

By preserving the legacy system's key aspects, we can leverage their previous investments, knowledge, and established business processes while benefiting from the advancements and efficiencies offered by the technologies we aim to switch.

3.3.2. Functional Needs

In this Sprint users can perform a specific number of tasks:

- The Administrator can log into his account.
- The Administrator can access informations concerning his account.
- The Administrator can modify profile's informations.
- The Administrator can log out.

3.3.3. Sprint Backlog

Story ID	Task ID	Task	Estimated Days	Priority
2	1	Execute Doctrine, a powerful object-relational mapper bundled with Symfony, to map objects into a MySQL database.	1	9
2	2	Migrate the database into Django models by inspecting the MySQL database and reconstructing the php classes into Python	3	6
2	3	Filtering Tables, relations and removal of irrelevant constructed tables	4	8
3	4	Creation of the Front and Back applications	2	7
3	5	Integrate Metronic template into the project's frontend for enhanced UI/UX.	4	10

Table 4 : Sprint 2 Backlog

3.3.4. Sprint Analysis

3.3.4.1. Task 2.1: Doctrine Execution

Symfony equips developers with a comprehensive suite of tools for efficient database integration in their applications, powered by Doctrine – a renowned collection of PHP libraries designed specifically for seamless database interaction. [7]

The process begins with installing "Doctrine" dependency then we go thorough creating the database by running the command "**php bin/console doctrine:database:create**", once the database is created, we can start managing schema changes and versioning.

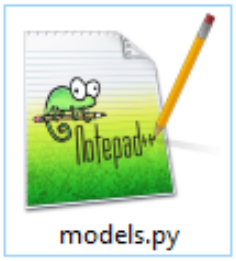
The "Doctrine Migrations Bundle" provides automatic generation of migration files by running the command "**php bin/console make:migration**", this command analyzes any changes the entities or annotations and generates a migration class that captures those changes.

Once the migration file is generated, we applied the changes to the database by running the command "**php bin/console doctrine:migrations:migrate**", This command executes the pending migrations and updates the database schema accordingly. This way we ensured that our database schema matches the defined structure in entity classes.

3.3.4.2. Task 2.2: Database Inspection

To generate Django model definitions based on our existing database, we used the "inspectdb" command of Django Framework.

By running "**python manage.py inspectdb > models.py**" in the terminal or command prompt within a shallow project, Django will analyze the database tables and automatically generate model definitions while the generated definitions will be saved in a file named models.py.



Once we have reviewed and customized the model definitions, we will be including progressively the classes from within "models.py" file by placing it within the new created app's directory.

```
PS C:\Users\MyComputer\Desktop\DjangoProjects\allcartImageAPI> python .\manage.py inspectdb
# This is an auto-generated Django model module.
# You'll have to do the following manually to clean this up:
#   * Rearrange models' order
#   * Make sure each model has one field with primary_key=True
#   * Make sure each ForeignKey and OneToOneField has `on_delete` set to the desired behavior
#   * Remove `managed = False` lines if you wish to allow Django to create, modify, and delete the tab
le
# Feel free to rename the models, but don't rename db_table values or field names.
from django.db import models
# Unable to inspect table 'sample_data'
# The error was: 'NoneType' object is not subscriptable
```

Figure 17 : inspectdb Command

This process saves time and effort, especially when working with an existing database schema in our case.

The "**inspectdb**" command in Django allows us to leverage an existing database by automatically generating model definitions.

This feature is particularly useful when you want to integrate Django with an established database schema.

3.3.4.3. Task 2.3: Entity Filtering

Some previous system entities will not be needed or used in the new environment, The entities that are no longer relevant or required for the operation of the new application have been carefully evaluated as part of the migration plan.

We wanted to simplify the migration procedure, optimize the database design, and improve the overall speed and maintainability of the new application by removing these superfluous items

By making this choice, we can concentrate on moving and integrating the crucial entities, assuring a smooth transition while removing extra complexity and any compatibility problems.

We refined the model definitions by removing the "**FOSUserBundle**", we will be incorporating the "models.py" file in a more scalable way by making the classes easy to update.

The "FOSUserBundle" [6] is a Symfony bundle in PHP that provides user management features for Symfony applications. It offers a set of ready-to-use functionalities to handle user registration, authentication, and other related tasks, however we will not be transitioning this bundle into the new project because of the existing user support available natively with the Django farmwork.

In our application, we have an entity called "Cabinet" that ensures that each administrator has a dedicated row. The technician who installs the application is responsible for building the Cabinet and mapping the first user as the administrator.

The technician uses a command-line interface to create a new Cabinet during the initial setup phase. This command-line interface enables them to specify the essential “Cabinet” information and configurations. Once the Cabinet is built, the technician assigns the first user, known as the "genesis user," as the Cabinet's administrator.

3.3.4.4. Task 2.4: Front & Back Applications

- ✓ *Front-End Application*

The moment has come to initiate the development of both the front-end and back-end applications.

We can rely on few straightforward procedures to construct the **React** app, therefore we used the create-react-app command to avoid configuring many build tools, The instant reloads will enable us to concentrate on the development tasks without interruptions and also when deployment phase arrives, the bundles will be instinctively optimized for efficient performance.

In our development parade, we will leverage the capabilities of several exceptional libraries, including Styled Components, Polished, Axios and React Forms to empower us with the creation of a robust and dynamic application.

We have been entrusted by our clients with something truly exceptional to harness in our project: The Metronic Template, which is a template based on Bootstrap.

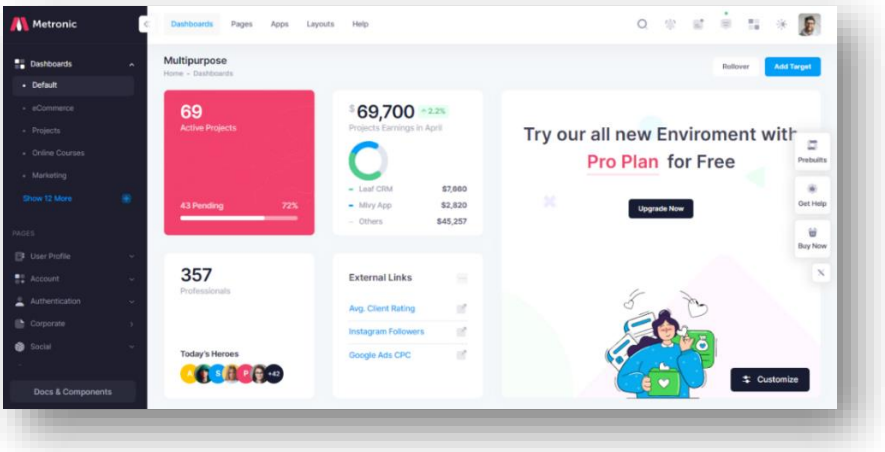


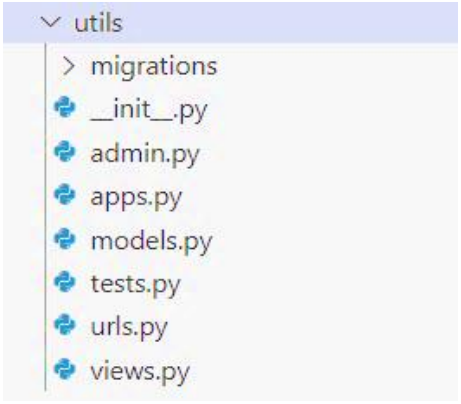
Figure 18 : Metronic Dashboard

✓ *Back-End Application*

Our initial step in developing the Django backend app was to set up a virtual environment in order to secure the project's isolation and properly manage its dependencies.

For our Django backend app, this strategy ensures a clear and independent configuration, minimizing conflicts and supplying a more dependable development environment.

After installing all the necessary packages and modules, we created a new Django app using the "**django-admin startapp**" command.



This will generate the necessary files and directories for your app, including models, views, and templates.

In a Django app, several files are orchestrated to play specific roles in defining the structure and operation of a typical Django program.

The following is a brief explanation of each file typically present in a Django app:

- **models.py**: This file uses Django's Object-Relational Mapping (ORM) framework to define the data models for the application. Database tables are represented by models, and each model class corresponds to a different table.

- **views.py**: The functions or classes that handle HTTP requests and produce suitable replies are found in the file. The logic involved with templates, obtaining data from models, and communicating with the app's business logic is determined by views.

- **urls.py**: The URL configuration of the app is specified in this file. It specifies how URLs and views are mapped, enabling Django to direct incoming requests to the proper view function or class.

- **admin.py**: Models have to be registered in the Django admin interface using the file. Data Management and manipulation is specified the built-in Django admin site by the model's representation and personalizing the admin interface.

3.3.4.5. Task 2.5: Metronic Template

This template brings forth a wealth of possibilities that will allow us to create a visually impressive and humble functional app.

With the solid foundation provided by **Bootstrap**, we can harness its pre-designed layouts, rich UI elements, and intuitive customization options to create a remarkable user experience.

One of the challenges we encountered with the Metronic template was the inability to fully utilize its pre-built React components due to limitations imposed by **TypeScript** and the new **NEXT.JS** app creation method, as a result, we had to resort to using the vanilla version of the template, relying on HTML, CSS, and JavaScript. However, this approach came with significant drawbacks, especially in terms of integrating JavaScript interactivity with React.

We attempted various methods to bridge this gap, including using a "ScriptTag" library and creating a custom hook that appends a script tag to the final DOM. However, these workarounds proved to be less than ideal. Modifying the public folder by adding scripts also presented its own set of challenges and was not considered a good practice. Consequently, we were only able to consume the CSS styles provided by the Metronic template.

While this limitation prevented us from fully leveraging the interactivity features of JavaScript within the React components, we strived to find rewrite the JavaScript to ensure a satisfactory user experience.

3.3.5. Sprint Conceptions

3.3.5.1. Story 1

The following is the sequence diagram describing “Doctrine” Migration

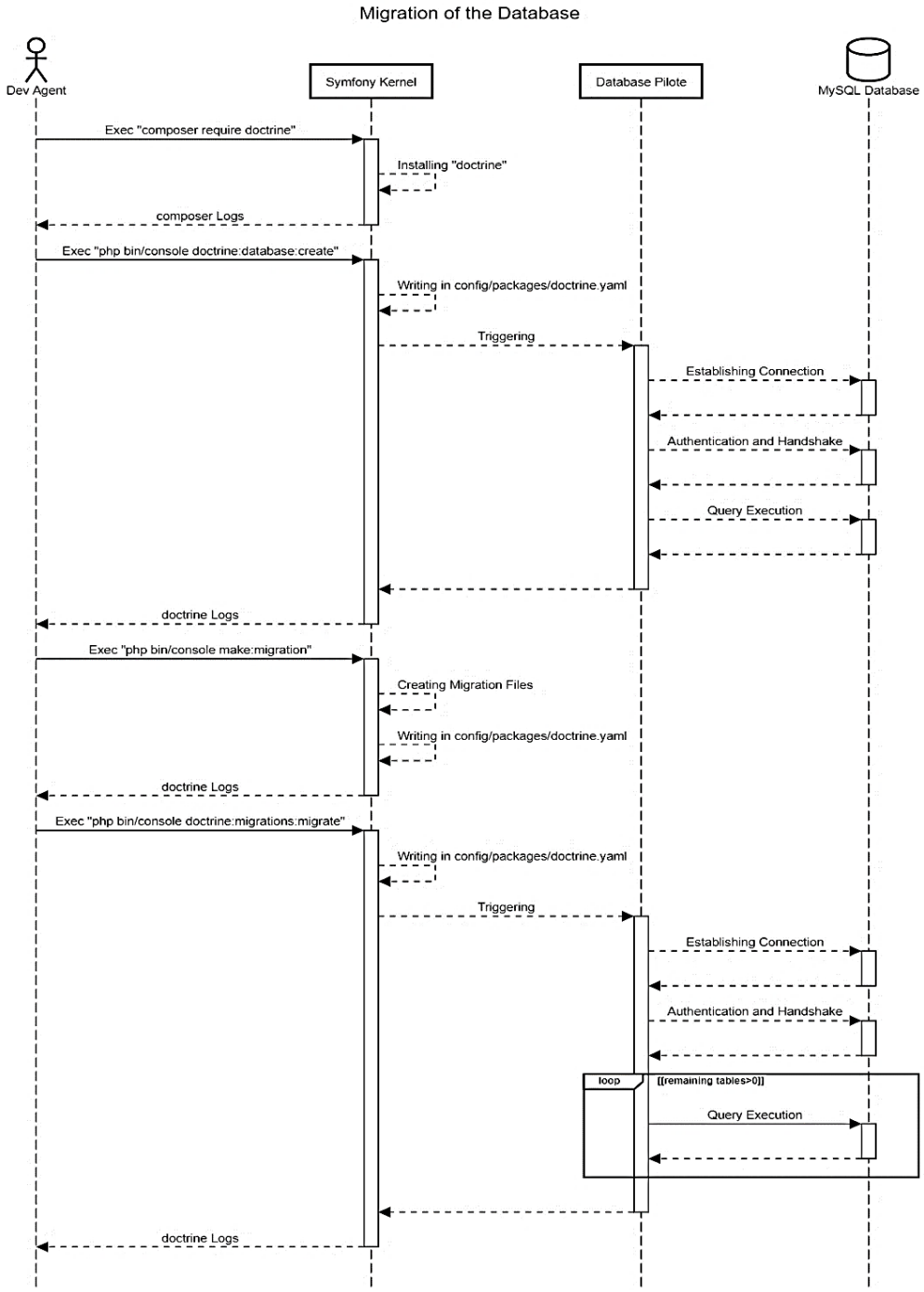


Figure 19 : Migration Sequence Diagram

The following is the sequence diagram describing the database inspection

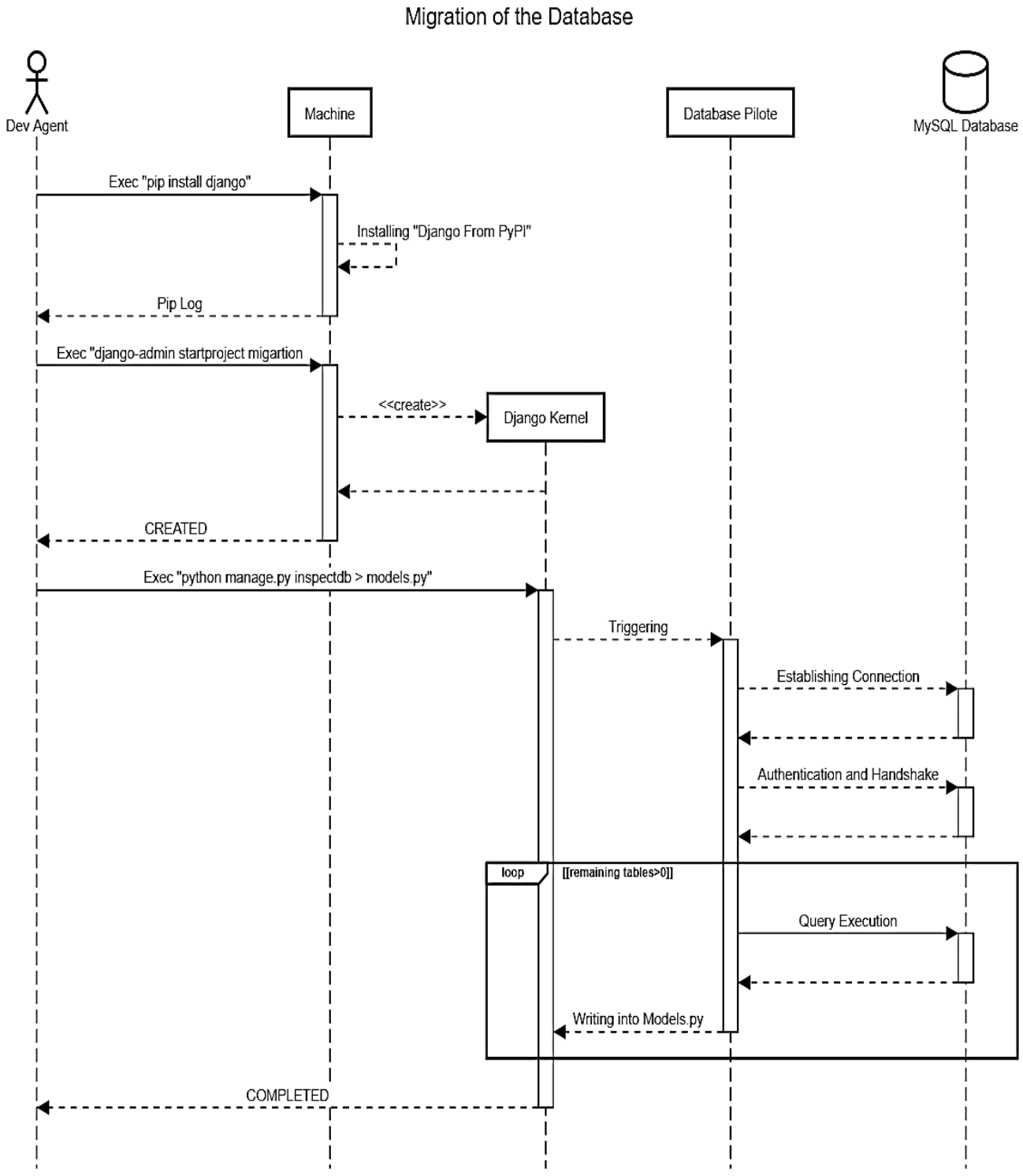


Figure 20 : Migration Sequence Diagram

3.3.5.2. Story 2

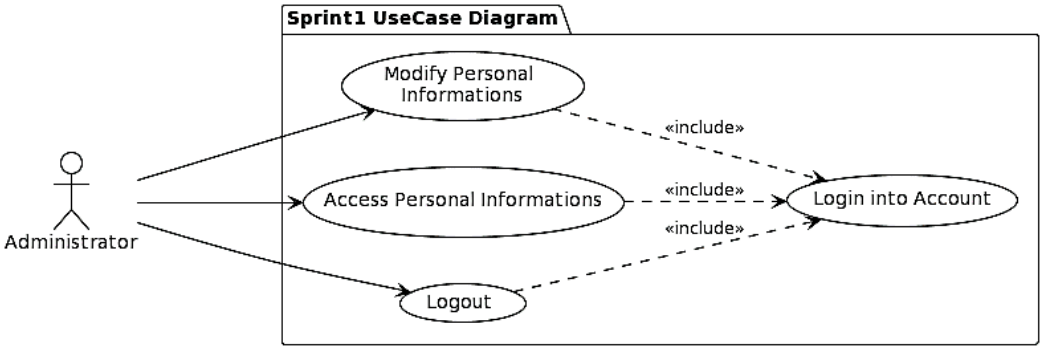


Figure 21 : Sprint 2 Use Case Diagram

Actor(s)	Administrator
Objectif	Log Into Account
Nominal Scenario	• Open the login page • Insert mail • Insert password • Press login button
Alternative Scenario	• If the mail is not found an error will occur • If the password doesn't match an error will occur

Table 5:Use Case Description "Login into Account"

Actor(s)	Administrator
Objectif	Modify Personal Informations
Precondition	User has to be authenticated
Nominal Scenario	• Open The User Menu • Press on "Paramètre de profile" • Press on "Modifier" • Insert informations to modify • Confirm by Pressing "Enregistrer"
Alternative Scenario	• If the form's entries are not correct an error will occur • If the password doesn't match an error will occur • The session is expired and will automatically logout

Table 6 : Use Case Description" Modify Personal Informations"

Actor(s)	Administrator
Objectif	Access Personal Informations
Precondition	User has to be authenticated
Nominal Scenario	• Open The User Menu • Press on "Profile"
Alternative Scenario	• The session is expired and will automatically logout

Table 7 : Use Case Description "Access Personal Informations"

Actor(s)	Administrator
Objectif	Logout
Precondition	User has to be authenticated
Nominal Scenario	• Open The User Menu • Press on “Deconnexion”
Alternative Scenario	• The session is expired and will automatically logout

Table 8: Use Case Description “Logout”

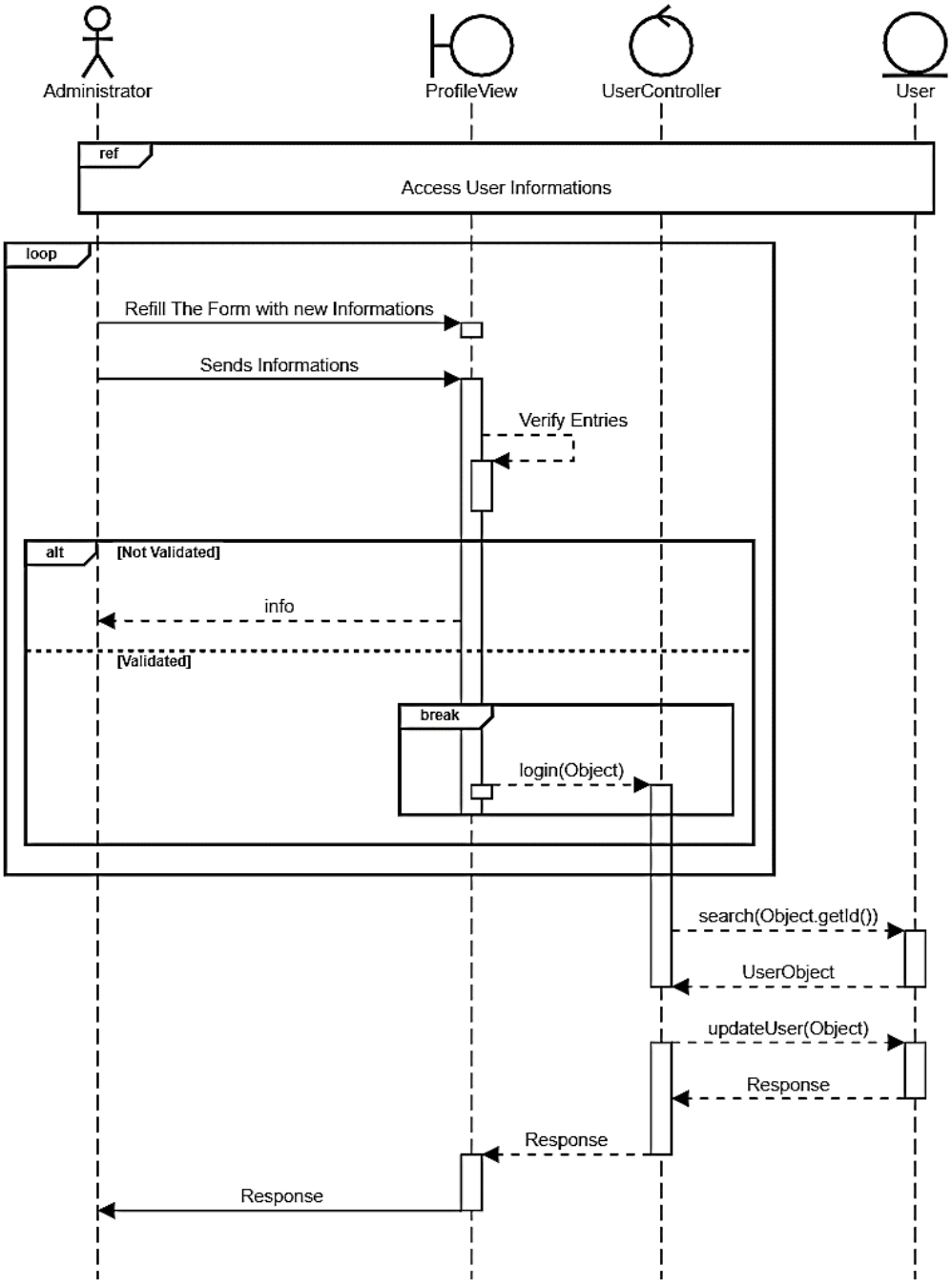


Figure 22 : “Modify Personal Informations” Sequence Diagram

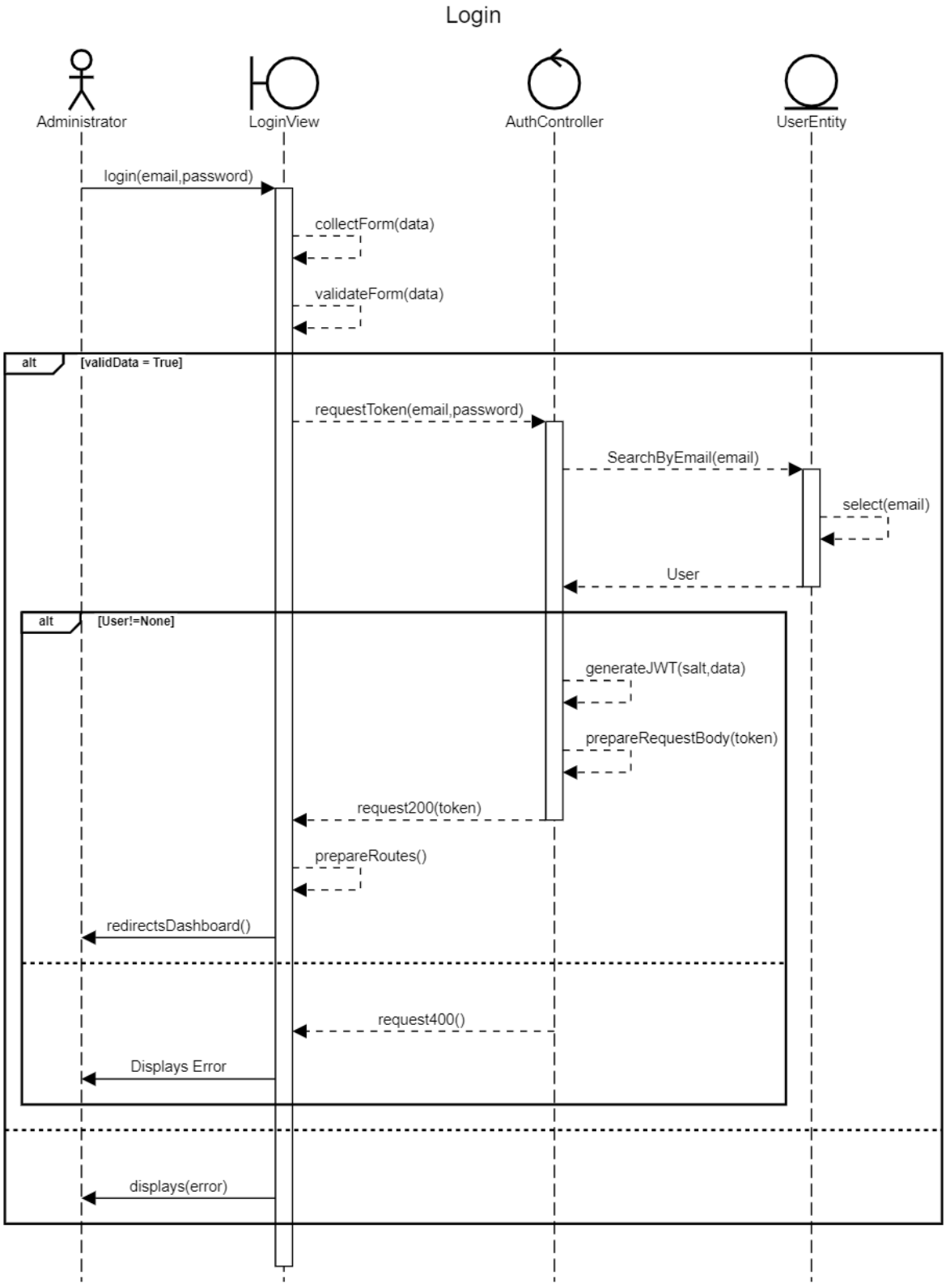


Figure 23: "Login into account" Sequance Diagram

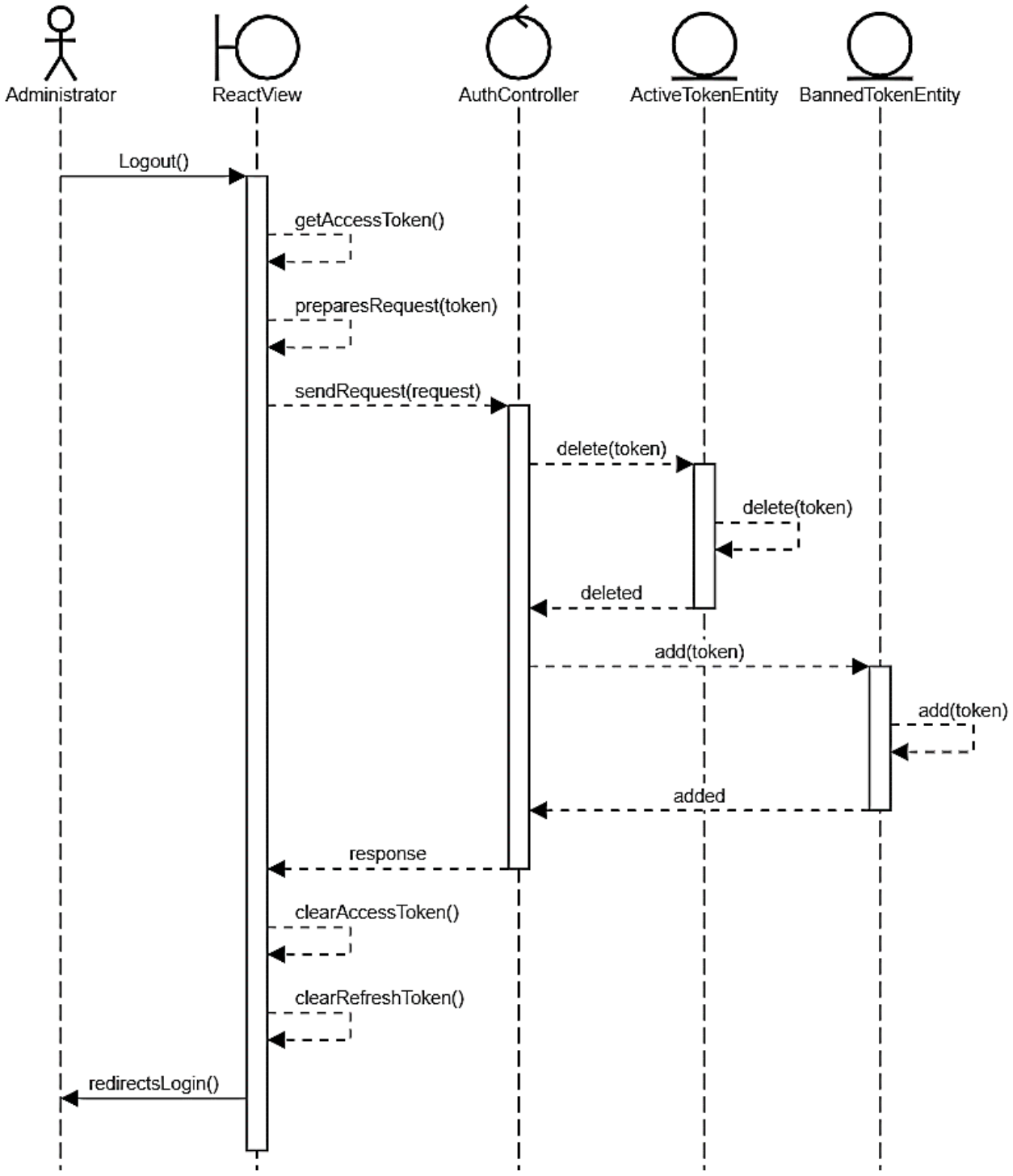


Figure 24: "Logout" Sequence diagram

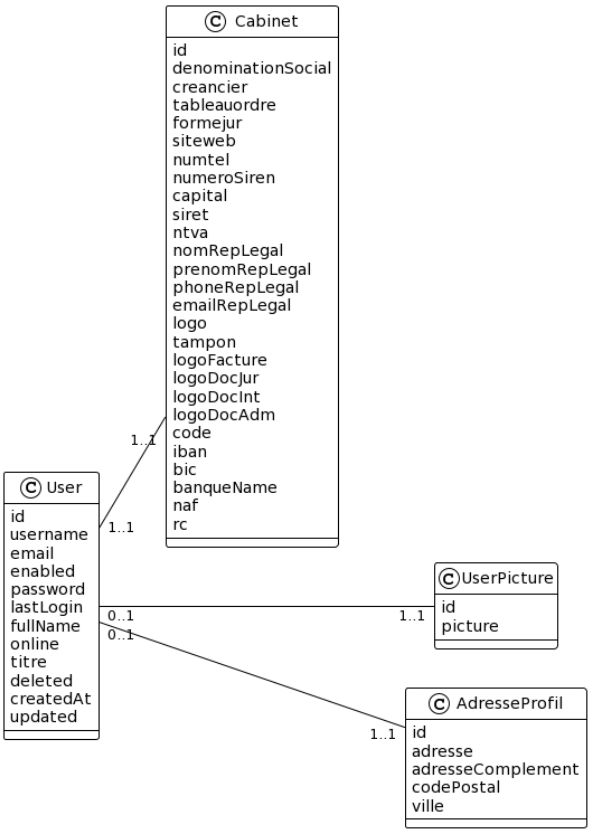


Figure 25 : Sprint 2 Class Diagram

3.3.6. Sprint Implementation

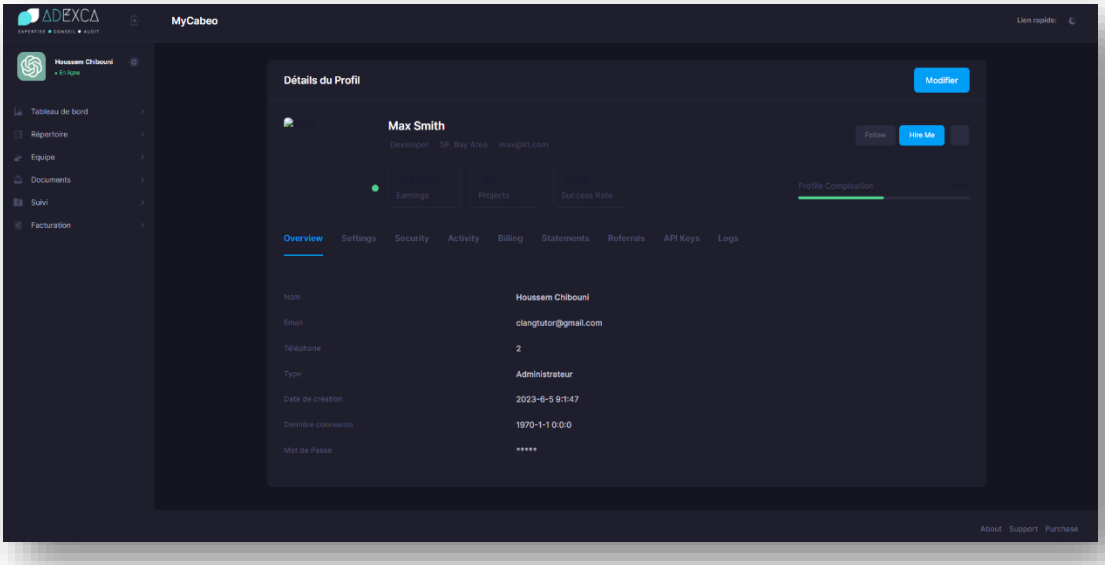


Figure 26 : Accessing User Informations

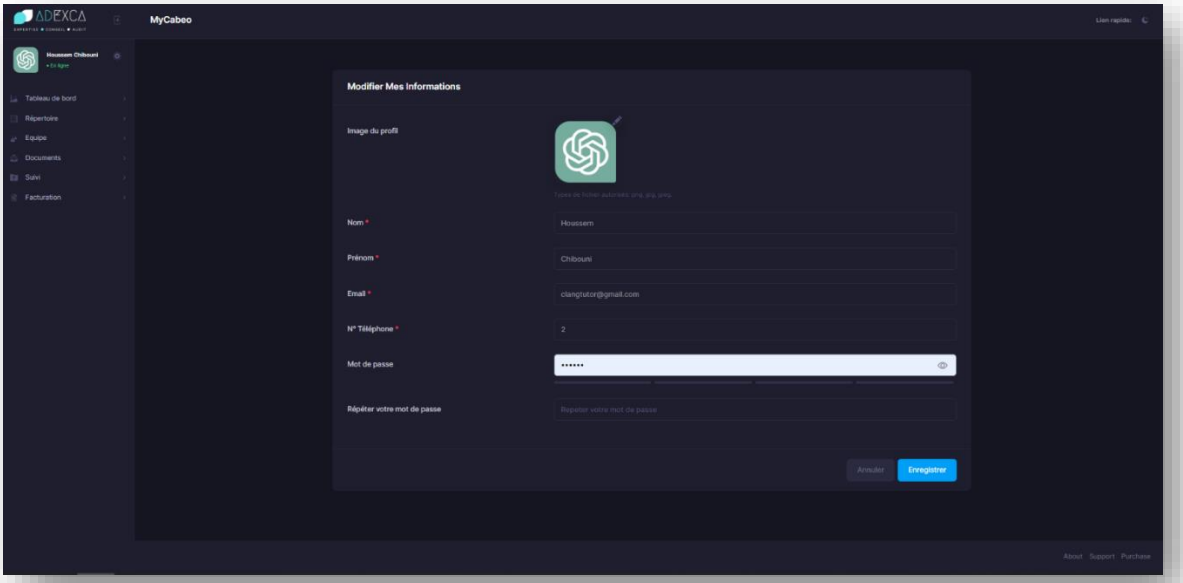


Figure 27 : Modifying User Informations

3.3.7. Sprint Review

During this sprint, Our Team completed the migration process of models from Symfony to Django, as well as encountered several challenges related to the integration of the Metronic Template into React.

However, we also encountered some difficulties when integrating the Metronic Template. The main issue was due to TypeScript's constraints, it was impossible to fully utilize the prebuilt React components provided by the Metronic Template. As a result, we were forced to use the template's default configuration, which relies on HTML, CSS, and JavaScript.

While this method efficiently enabled us to consume CSS styles, it presented difficulties when attempting to integrate JavaScript interactivity with our React components.

3.3.8. Sprint Retrospective

The retrospective is a meeting during which the Scrum team uses its experience over the past sprint to improve its organization in order to be more efficient.

What worked well	What can be improved
The precision of the adapted solutions in terms of time and implementation.	More usage of prebuilt packages to optimize the code

Table 9 : Sprint 2 improvement plan

3.4. Conclusion

In conclusion, Release 1 encompassed two sprints: Sprint 1 focused on familiarizing ourselves with the old technology stack, while Sprint 2 marked the beginning of our application development journey.

4. Chapter: Release 2

4.1. Introduction:

This chapter will include the backlog, the different diagrams and the description for the realization of the first release with its graphical interfaces.

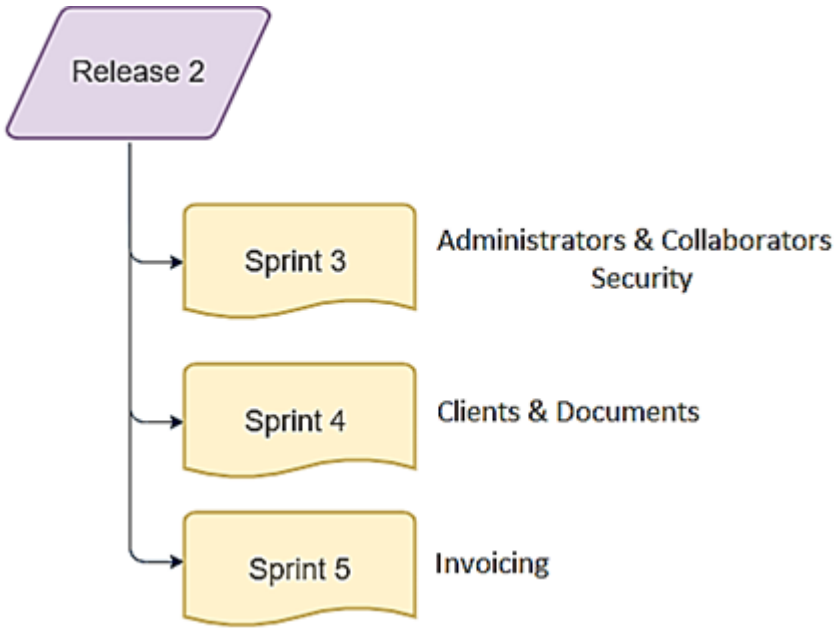


Figure 28 : Release 2

4.1. Sprint 3: «Administrators & Collaborators Security»

4.1.1. Objectif

The objective of this sprint is to define and implement the collaborator entity within our application. This will involve importing the necessary database schema and creating the establishing the Django models, repositories, and services.

The collaborator entity will serve as a key component in our application's user management and authentication system, also we are aiming to enhance the security

of our APIs by implementing the “bcrypt” hashing algorithm, we can securely store and compare passwords.

We will be also defining a more robust code separation in our React application to enhance the modularity and the evolving criteria of our project.

4.1.2. Functional Needs

In this Sprint users can perform a specific number of tasks:

- The administrator can access a collaborator’s informations.
- The administrator can add a collaborator to the cabinet.
- The administrator can modify a collaborator within his cabinet.
- The administrator can delete a collaborator within his cabinet.
- The collaborator can log into his account.
- The collaborator can logout.
- The collaborator can access his personal informations.
- The collaborator can modify his personal informations.

4.1.2. Sprint Backlog

Story ID	Task ID	Task	Estimated Days	Priority
4	1	Separate React folders and initialize main components	5	7
4	2	Systematically organize React components	2	6
5	1	Implement JWT technique	1	5
5	2	Implement Bcrypt hashing algorithm	1	6
6	1	Create reusable components for forms and authentication	1	6
7	1	Create the collaborator's components	2	8
7	2	Building the API service of Collaborators	2	8

Table 10 : Sprint 3 Backlog

4.1.3. Sprint Analysis

4.1.3.1. Task 4.1: React Folders

To manage the codebase of the front application, it's usual practice to separate the code folders though we will be using a variety of strategies, but one well-liked and suggested folder layout is the "Feature-based" or "Domain-based" structure.

public: that contains static assets and files that are not processed or transformed by the build tools

src: contains all the front business logic and definitions

api: this folder contains Axios API calls used to grasp data from the backend

components: this folder contains components that are specific to the feature. These components are typically not shared outside of the feature.

hooks: It contains unique React hooks we wrote write for our application.

interceptors: this folder contains patterns and mechanisms used to intercept and modify requests or responses in a network communication layer.

media: contains static media files such as pictures and figures.

var: contains application constants mapped from “dotenv” file.

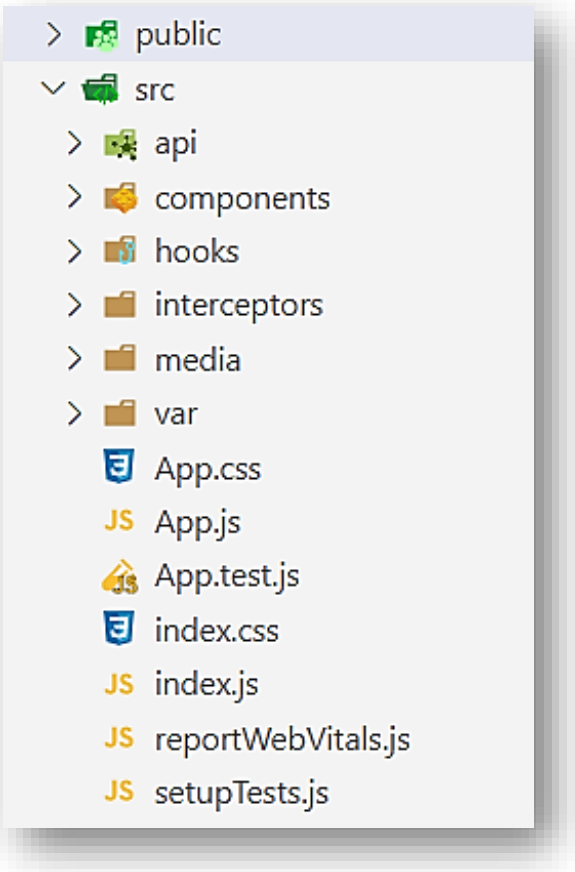


Figure 29 : Hierarchy of the React Application

4.1.3.2. Task 4.2: React Components

In our React application, we have organized our components in a structured manner. At the top level, we have a main component that serves as the container for our application. This main component consists of a header, a sidebar, and a footer.

- The header component contains the branding theme button at the top of our application. It provides a consistent look and feel across different pages or views.

- The sidebar component is responsible for displaying the navigation menu. It contains a list of menu items, each accompanied by a dropdown component. The dropdown component enhances the user experience by providing additional options or submenus when a menu item is clicked or hovered over, additionally, we have implemented a user dropdown menu.

  This user dropdown menu serves as a convenient and accessible way for users to access their profile information, profile settings, and logout button.

- When the user clicks on the user dropdown menu, it expands to reveal the available options. The profile information section displays relevant details about the user, such as their name, profile picture, or any other personalized information.

- The profile settings option within the user dropdown menu allows users to access and modify their account settings. This can include actions such as

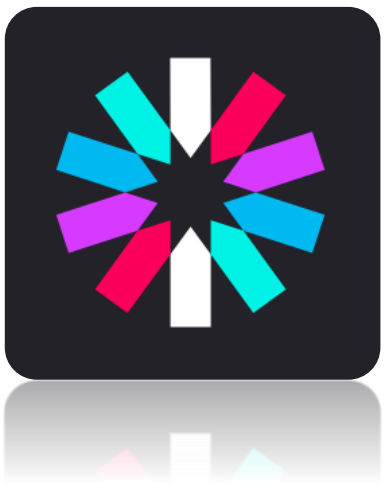
updating their profile information, changing passwords, or managing privacy settings.

- Finally, the logout button provides a quick way for users to securely log out of the application. When clicked, it initiates the logout process, clearing any active session data and redirecting the user to the appropriate landing page, such as a login or home page.

- The footer component is placed at the bottom of our application and typically includes relevant links, copyright information.

Within the main component, we have a central content area where the changing content of our application is rendered. This allowed us to dynamically update and display different components or views based on user interactions and routing.

4.1.3.3. Task 5.1: Secure Authentication



JWT (JSON Web Token) is a small and self-contained system for securely sending information between layers as a JSON object and we will be implementing it in our applications for authentication and permission. JWT tokens are digitally signed with a secret or public/private key combination, confirming the token's integrity and authenticity.

We will be setting expiration timestamps for our tokens, which can be used to manage sessions. To receive a fresh token after an expired token, the user must re-authenticate using a refresh token.

Django provides a prebuilt for working with JWT in back-end applications. This library integrates JWT authentication into the Django REST framework, making it easier to implement secure token-based authentication in our API endpoints.

You can configure the token session's timestamp and refresh timestamp

4.1.3.4. Task 5.2: Hashing using Bcrypt

"bcrypt" is a popular password hashing technique that is used to securely store passwords.

It is well-known for its resistance to brute-force attacks and its ability to stymie password cracking attempts. It is based on the Blowfish symmetric encryption cipher [10].

In a typical scenario, the front-end app should not hash the user's password overwise it has to be sent securely over an encrypted connection to the server.

The server-side application would then handle the password hashing using "bcrypt" algorithm then it's stored in the server's database, ensuring that the user's password is securely protected.

The hashing should be done at the back-end. The back-end is under your control, so you can enforce that the hashing is taking place as it should. Additionally you can have client-side hashing. [11]

4.1.3.5. Task 6.1: Reusable Components

In my React application, I designed reusable components for forms and authentication. These components are intended to improve the efficiency and scalability of form handling and user authentication processes.

- ✓ *Native Select Component*

I developed a "**Native Select Component**" in Metronic Style: I designed a native select component that matches the Metronic design. This component allows users to select items from a visually pleasing and consistent dropdown menu and perform search operations.

- ✓ *Component based on utilities from "React Forms"*

I also integrated "React Forms" to automate form input handling; this library includes utilities and hooks, such as react-hook-form, that help to ease form validation, data submission, and error handling. I can easily control form using this library.

- ✓ *Action Button*

I have developed an interactive "**Action Button**" component that can be included in all my forms and views. This component not only serves as a reusable button element but it also provides additional functionality through its submenus.

The "**ActionButton**" component has submenus that allow the user to choose between two operations:

- Redirecting to a link or invoking an API call: This versatility enables the button to handle various scenarios and cater to different user requirements.

- Option to invoke an API call: the "ActionButton" component triggers a method that initiates the API request. This can be used to perform actions such as saving form data, updating user information, deleting entities or interacting with backend services.

✓ *Not Found Component*

I have implemented a responsive "**Not Found**" page in my application to handle cases where users navigate to invalid links or routes.

This page is intended to complement the overall concept of the application, resulting in a consistent and visually pleasing experience.

✓ *Popup Alert Component*

I have implemented a "**Popup Alert component**" that can be called when the user attempts to perform an invalid action. This popup serves as a user-friendly way to deliver informative messages and guide their interactions.

When the user performs an action that is not possible or violates certain conditions, the popup component is invoked to display a message explaining the issue currently occurring.

This can include error messages, warnings, or any other relevant information to help the user understand why the action cannot be performed.

The popup component is visually beautiful and intuitive, drawing the user's attention to the crucial message being given. We adjusted to match the application's overall theme and layout, ensuring a consistent appearance and feel.

4.1.3.6. Task 7.1: Collaborator's Components

I have implemented two components that are related to collaborators: the form component and the list component with action buttons for delete and modify functionality.

The form component serves as an interface for creating or editing collaborator data. It includes input fields and validation to ensure accurate and complete information is provided. Users can enter details such as name, email, role, and other relevant data.

The list component displays a list of existing collaborators working in the "Cabinet" which we distributed the application. Each row in the list is accompanied by "Action button" s for delete and modify functionality.

These buttons enable the user to remove collaborators from the list or to update their informations, clicking on the delete button triggers a confirmation popup, ensuring that users confirm their intent before proceeding with the deletion.

4.1.3.7. Task 7.2: Collaborator's API

I have implemented API methods that encapsulate the necessary logic for collaborating with the backend services. These API methods handle tasks such as retrieving the list of collaborators, submitting form data, updating existing rows, and deleting collaborators.

This API call will trigger the creation of a new user account that will be assigned to the added collaborator.

4.1.4. Sprint Conceptions

4.1.4.1. Story 7

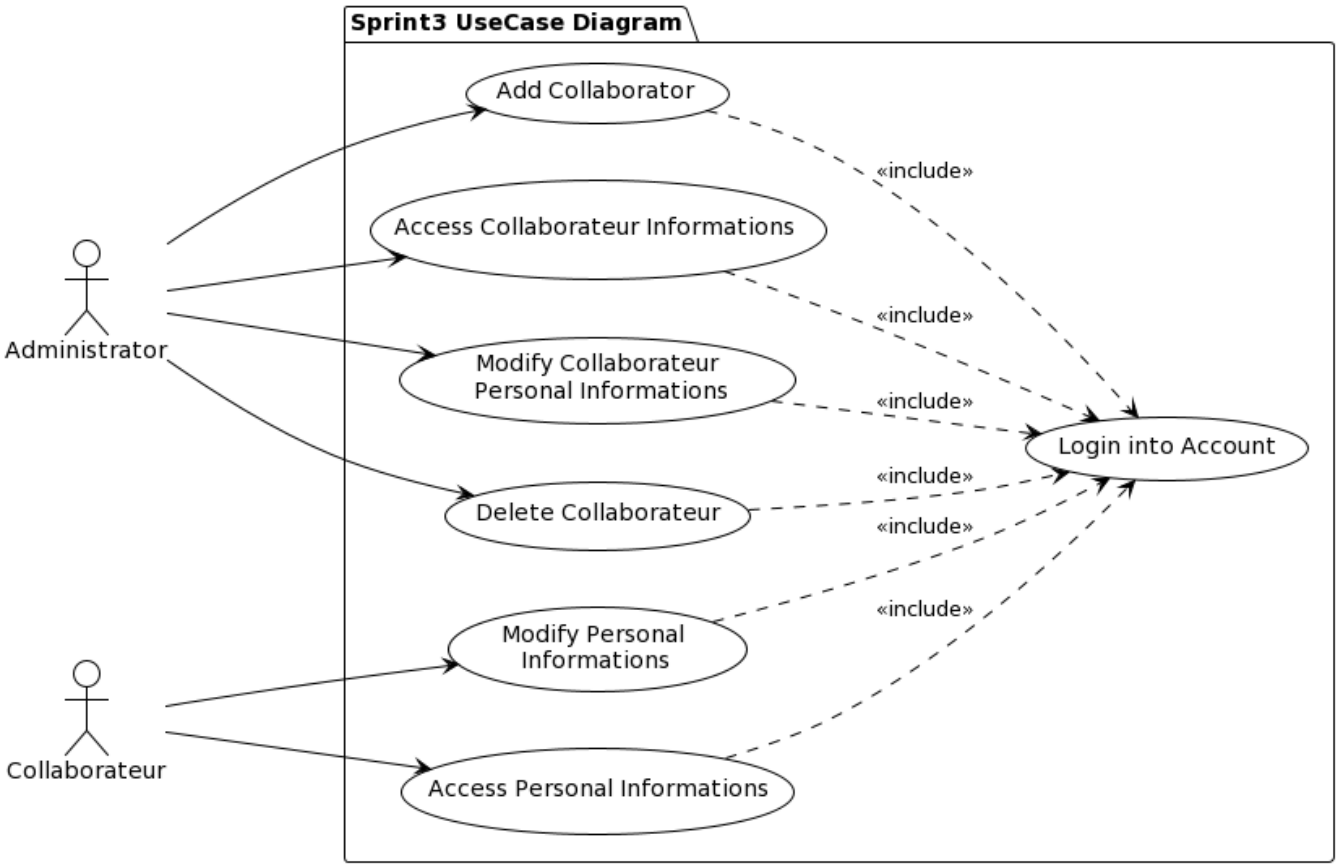


Figure 30 : Sprint 3 Use Case Diagram

Actor(s)	Administrator
Objectif	Add Collaborator
Precondition	Administrator has to be authenticated
Nominal Scenario	• Open the sidebar • Press on “Equipe” menu • Choose “Liste des Collaborateur” sub-menu • Press on “Ajouter” • Fill the form • Submit the form
Alternative Scenario	• If the form’s entries are not correct an error will occur • The session is expired and will automatically logout

Table 11 : Use Case Description “Add Collaborator”

Actor(s)	Administrator
Objectif	Access Collaborator Informations
Precondition	Administrator has to be authenticated
Nominal Scenario	• Open the sidebar • Press on “Equipe” menu • Choose “Liste des Collaborateur” sub-menu
Alternative Scenario	• If the form’s entries are not correct an error will occur • If the password doesn’t match an error will occur • The session is expired and will automatically logout

Table 12 : Use Case Description “Access Collaborator Informations”

Actor(s)	Administrator
Objectif	Modify Collaborator’s Personal Informations
Precondition	Administrator has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Equipe” Menu • Choose “Liste des Collaborateurs” sub-menu • Use the action button • Press on Modify • Fill the form • Submit the form
Alternative Scenario	• If the form’s entries are not correct an error will occur • The session is expired and will automatically logout

Table 13 : Use Case Description “Modify Collaborator’s Personal Informations”

Actor(s)	Administrator
Objectif	Delete Collaborator
Precondition	Administrator has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Equipe” Menu • Choose “Liste des Collaborateurs” sub-menu • Use the action button • Press on “Effacer” • Confirm the deletion of the collaborator
Alternative Scenario	• The session is expired and will automatically logout

Table 14 : Use Case Description “Delete Collaborator”

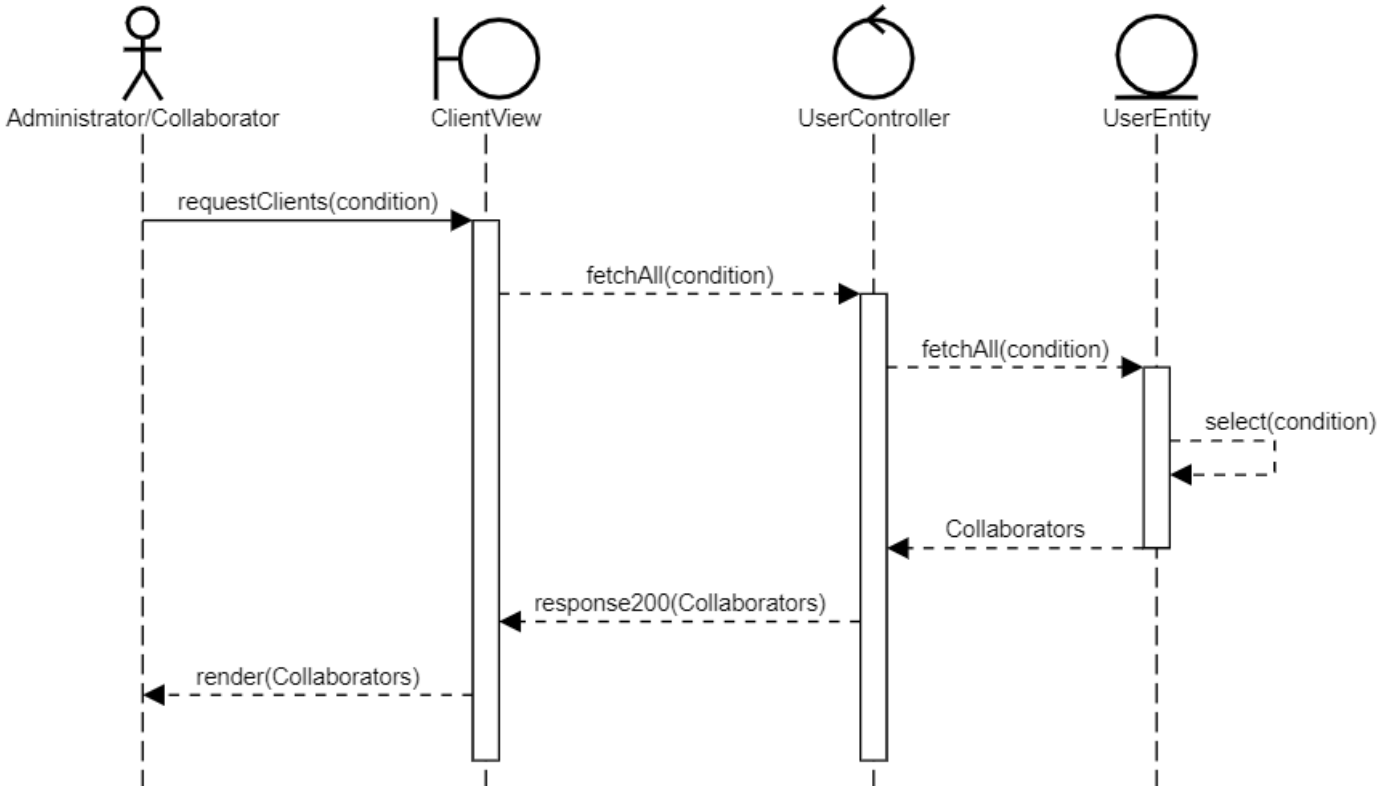


Figure 31: “Access Collaborator” Sequance diagram

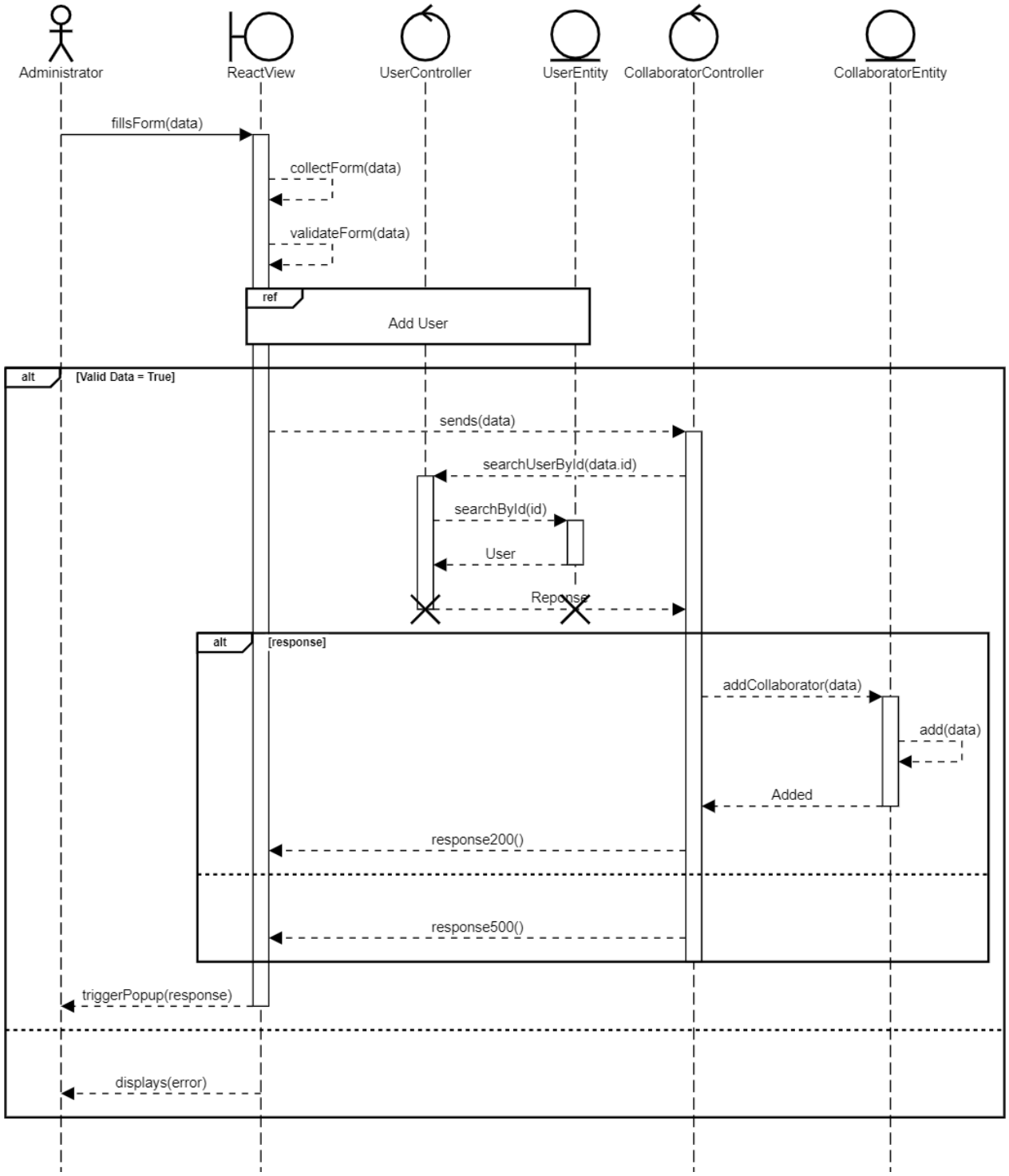


Figure 32 : "Add Collaborator" Sequence Diagram

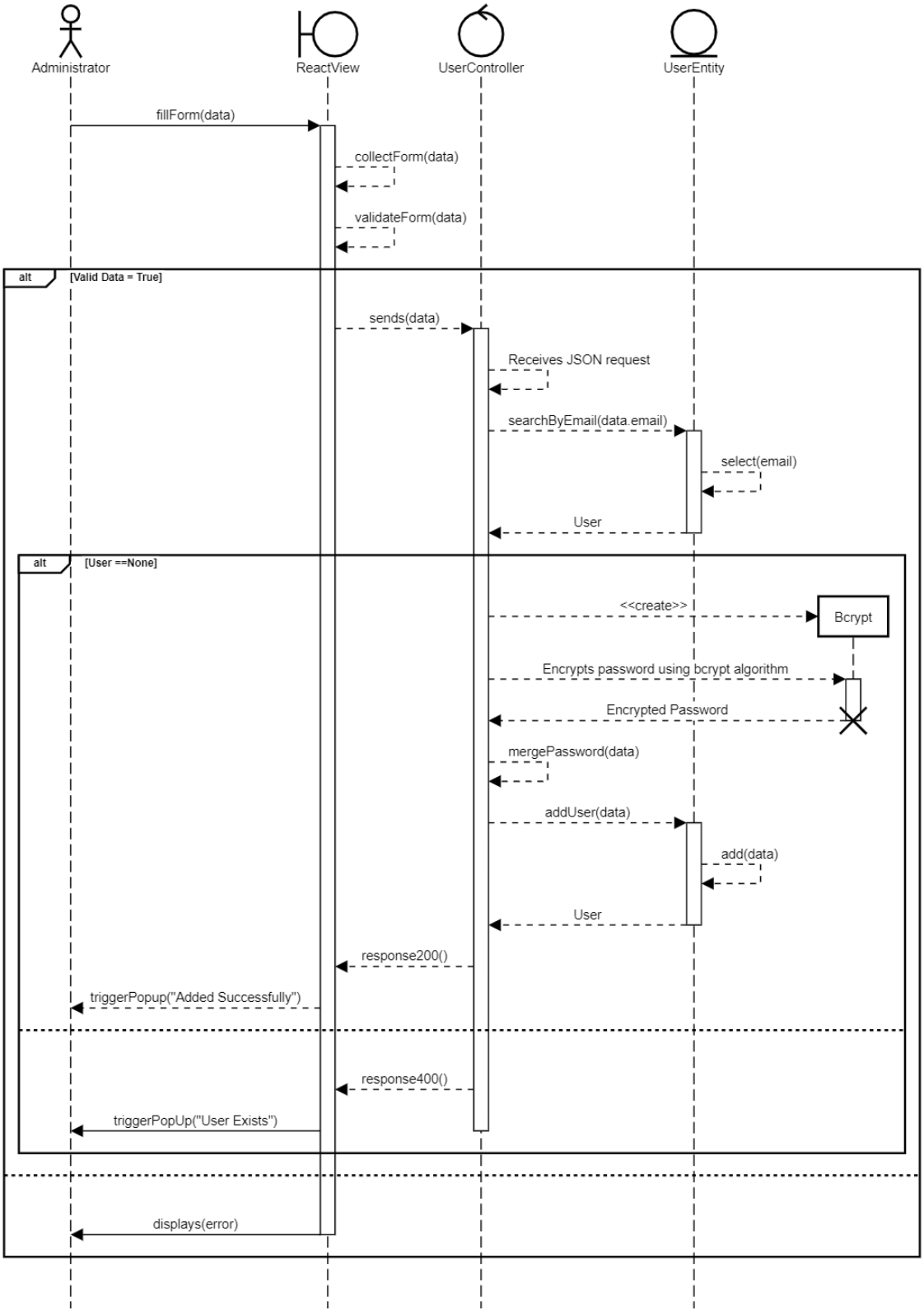


Figure 33 : "Add User" Sequence Diagram

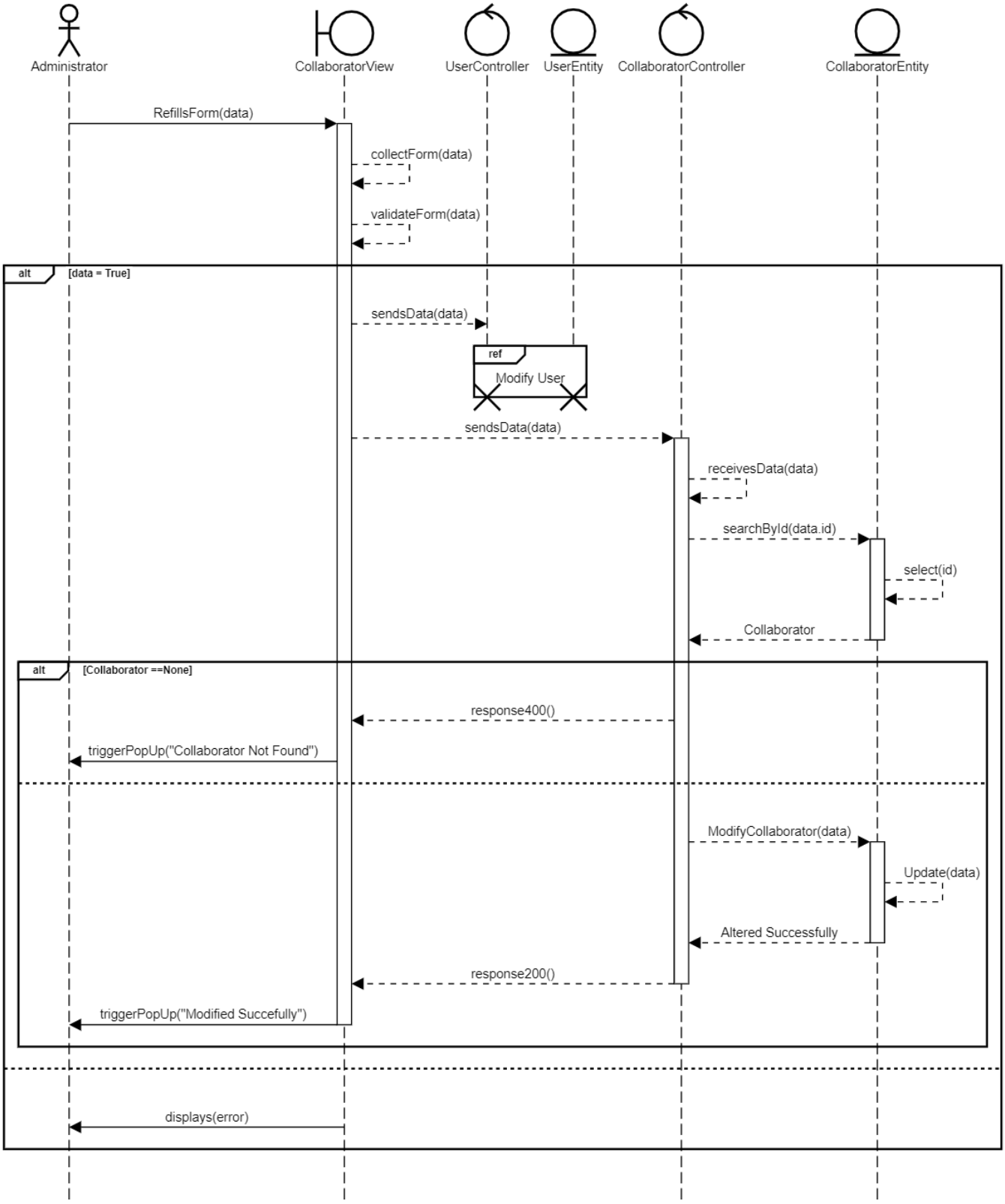


Figure 34 : "Modify Collaborator" Sequance Diagram

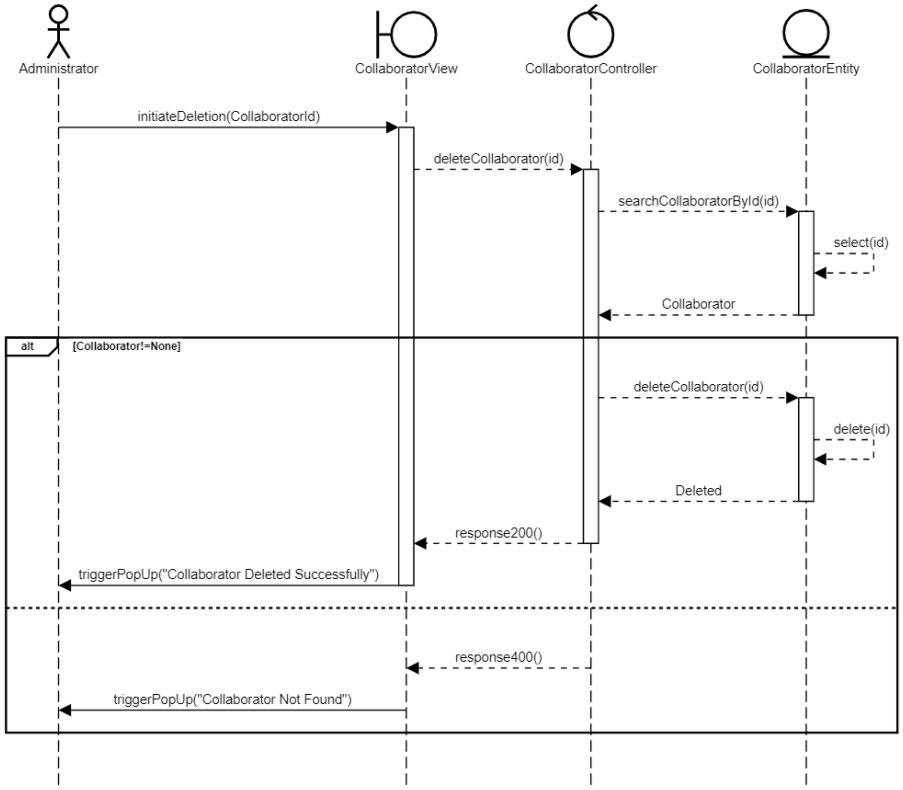


Figure 35 : Delete Collaborator Sequence Diagram

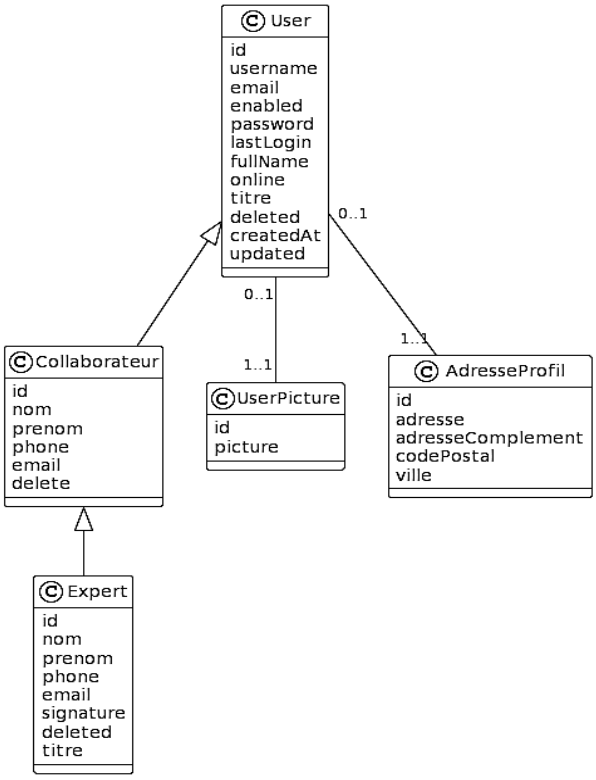


Figure 36 : Sprint 3 Class Diagram

4.1.5. Sprint Implementation

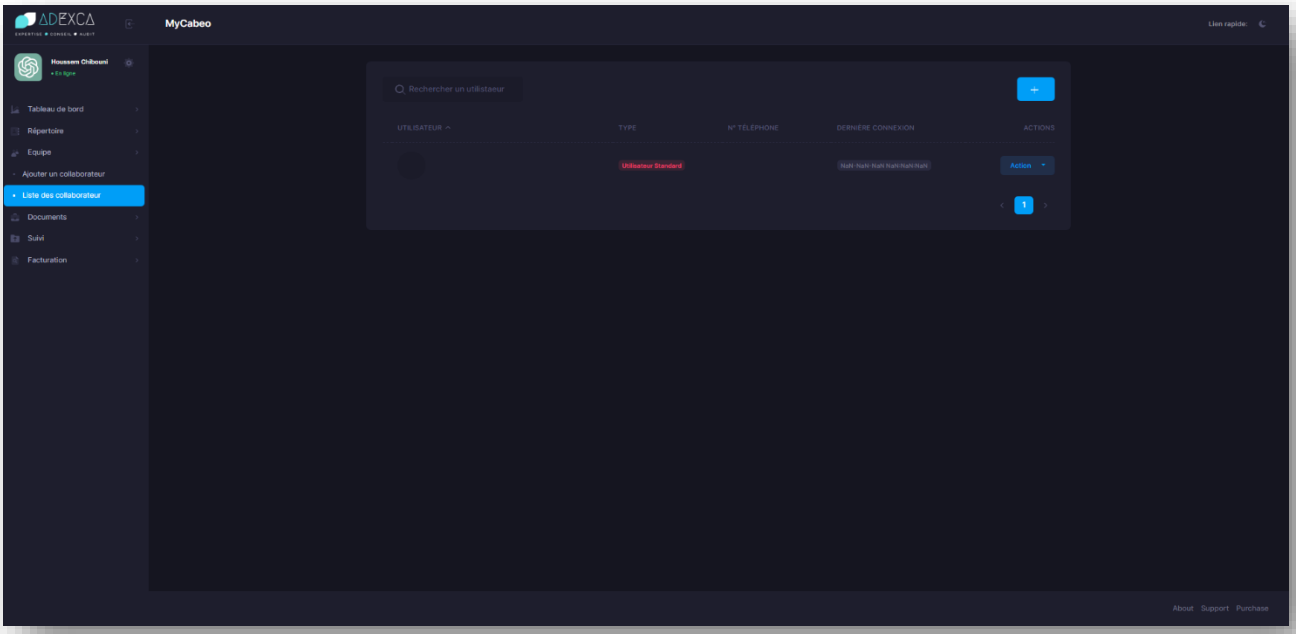


Figure 37 : Listing All the Collaborators

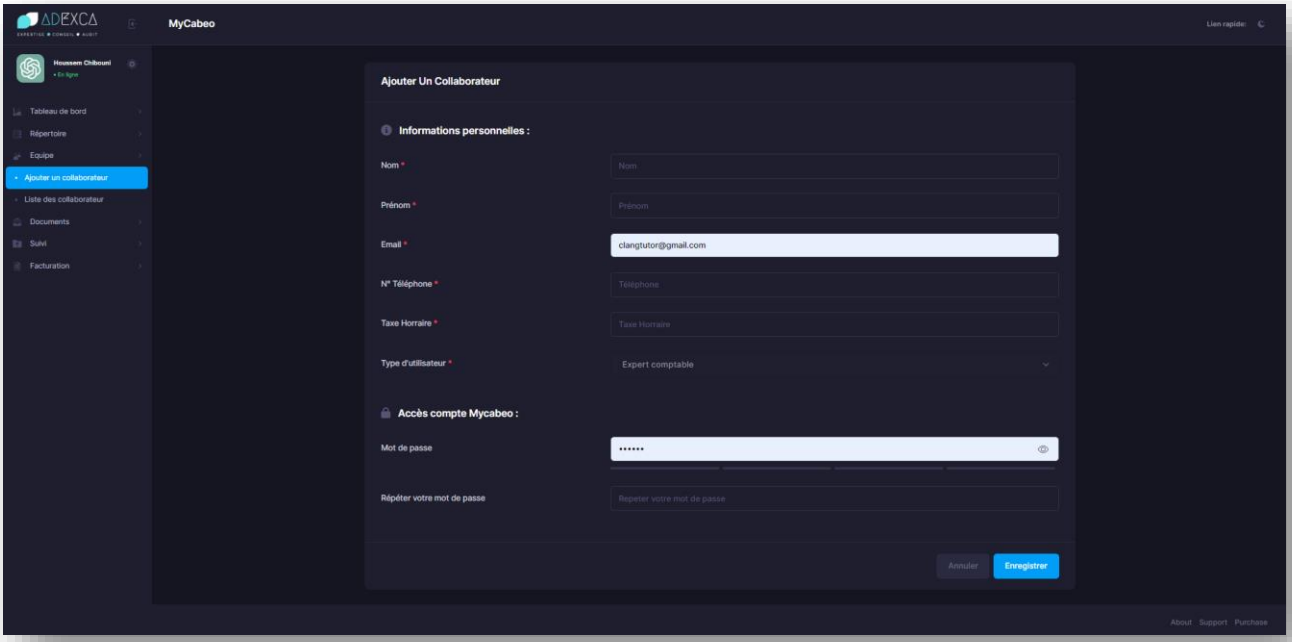


Figure 38 : Adding Collaborators

4.1.6. Sprint Review

In this sprint, we focused on two key stories.

Firstly, we tackled a technical story to organize our React components, improving the structure and maintainability. This enhanced the implementation of a systematic approach, establishing a coherent folder structure, and promoting component reusability.

Secondly, we successfully completed a user story that involved implementing the collaborator entity.

4.1.7. Sprint Retrospective

The retrospective is a meeting during which the Scrum team uses its experience over the past sprint to improve its organization in order to be more efficient.

What worked well	What can be improved
• The efficiency reusable components we developed earlier	• Business Logic Implementation

Table 15: Sprint 3 improvement plan

4.3. Sprint 4: «Clients & Documents »

4.3.1. Objectif

The objective of this sprint is to define and implement the collaborator entity within our application. This will involve importing the necessary database schema and creating the establishing the Django models, repositories, and services.

4.3.2. Functional Needs:

In this Sprint users can perform a specific number of tasks:

- The administrator and collaborators can add an enterprise.
- The administrator and collaborators can access an enterprise.
- The administrator and collaborators can modify an enterprise.
- The administrator and collaborators can delete an enterprise.
- The administrator and collaborators can add a client.
- The administrator and collaborators can access a client.
- The administrator and collaborators can modify a client.
- The administrator and collaborators can delete a client.
- The Administrator and collaborator can handle internal documents
- The Administrator and collaborator can handle couriers
- The Administrator and collaborator can handle legal documents

4.3.3. Sprint Backlog

Story ID	Task ID	Task	Estimated Days	Priority
8	1	Build the API services of “Repertoire”	4	7
8	2	Create the “Repertoire” components	3	8
9	1	Build the API services of “Documents”	4	8
9	2	Create the “Documents” components	3	7

Table 16 : Sprint 4 Backlog

4.3.4. Sprint Analysis

4.3.4.1. Task 8.1 & Task 9.1

The Repertoire API is created to facilitate the management of clients, prospects, and contacts within the enterprise. This API enables efficient tracking and organization of relevant information related to these entities.

The "clients" category represents the current stakeholders of the enterprise.

These are the individuals or organizations with whom the business has an ongoing relationship. The Repertoire API allows for the storage and retrieval of client data, including contact details, transaction history, and any other pertinent information.

The "prospects" category encompasses potential business opportunities. These are individuals or organizations that the enterprise intends to engage with to establish new partnerships or secure future projects.

The Repertoire API enables the storage and management of prospect information, including leads, interactions, and progress tracking towards converting them into clients.

Lastly, the "contacts" category consists of individuals known to the enterprise but who have not yet become clients. These contacts may have expressed interest in the business, attended events, or engaged in preliminary discussions. The Repertoire API enables the tracking of contact details, communication history, and any relevant notes or updates.

By implementing the Repertoire API, the enterprise can effectively organize and manage its relationships with clients, prospects, and contacts.

4.3.4.2. Task 8.2 & Task 9.2

The Repertoires components contains several key components of the previous releases that facilitate the management of clients, prospects, and contacts within the enterprise. It operates like every CRUD operation we did in the previous releases.

When using the Documents API service, uploading a document or accessing existing documents requires the Storage handler service to be seamlessly integrated.

The Storage handler service connects the API to the underlying storage infrastructure.

The API gets document data from the client application in order to add a document. The API then sends this information to the Storage handler service, which

is in charge of the file upload procedure. The Storage handler service ensures that the document is uploaded to the storage infrastructure in an ordered structure.

The Storage handler service generates a filename for the document after it is uploaded and returns it to the API. This identifier can then be associated with the corresponding metadata by the API and saved in the database for future reference.

When accessing documents, the API gets a request from the client application that includes the unique identifier of the document as well as any other relevant arguments. To retrieve the requested document from the storage infrastructure, the API talks with the Storage handler service.

4.3.5. Sprint Conceptions

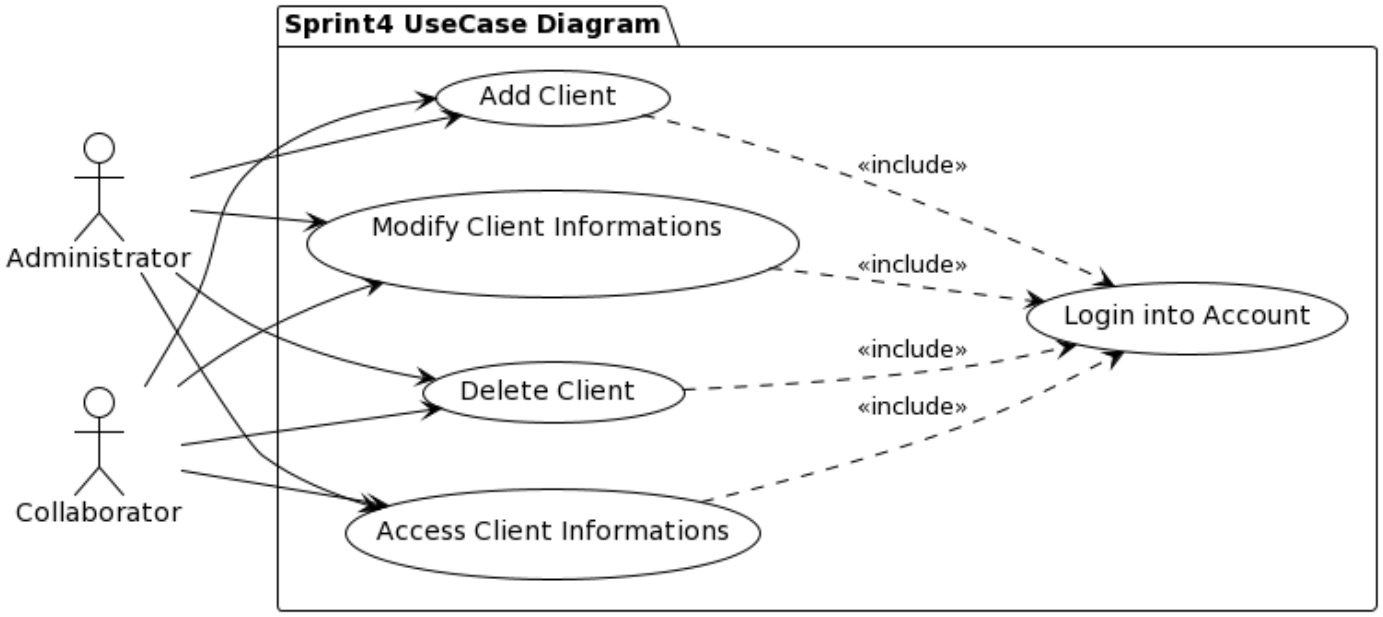


Figure 39 : Story 8 Use Case Diagram

Actor(s)	Administrator / Collaborator
Objectif	Add Client
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Repertoire” Menu • Choose the sub-menu of the client’s type • Press on “Ajouter *” • Fill the form • Submit the form
Alternative Scenario	• The session is expired and will automatically logout • If the form’s entries are not correct an error will occur

Table 17 : Use Case Description “Add Client”

Actor(s)	Administrator / Collaborator
Objectif	Modify Client Informations
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Repertoire” Menu • Choose the sub-menu of the client’s type • Press on “Modifier” • Rewrite the form • Submit the form
Alternative Scenario	• The session is expired and will automatically logout • If the form’s entries are not correct an error will occur

Table 18 : Use Case Description “Access Client Informations”

Actor(s)	Administrator / Collaborator
Objectif	Modify Client Informations
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Repertoire” Menu • Choose the sub-menu of the client’s type • Press on “Modifier” • Rewrite the form • Submit the form
Alternative Scenario	• The session is expired and will automatically logout • If the form’s entries are not correct an error will occur

Table 19 : Use Case Description” Modify Client Informations”

Actor(s)	Administrator / Collaborator
Objectif	Delete Client
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Repertoire” Menu • Choose the sub-menu of the client’s type • Press on “Action Button” • Press on “Effacer” • Confirm the deletion of the client
Alternative Scenario	• The session is expired and will automatically logout • If the form’s entries are not correct an error will occur

Table 20 : Use Case Description” Delete Client”

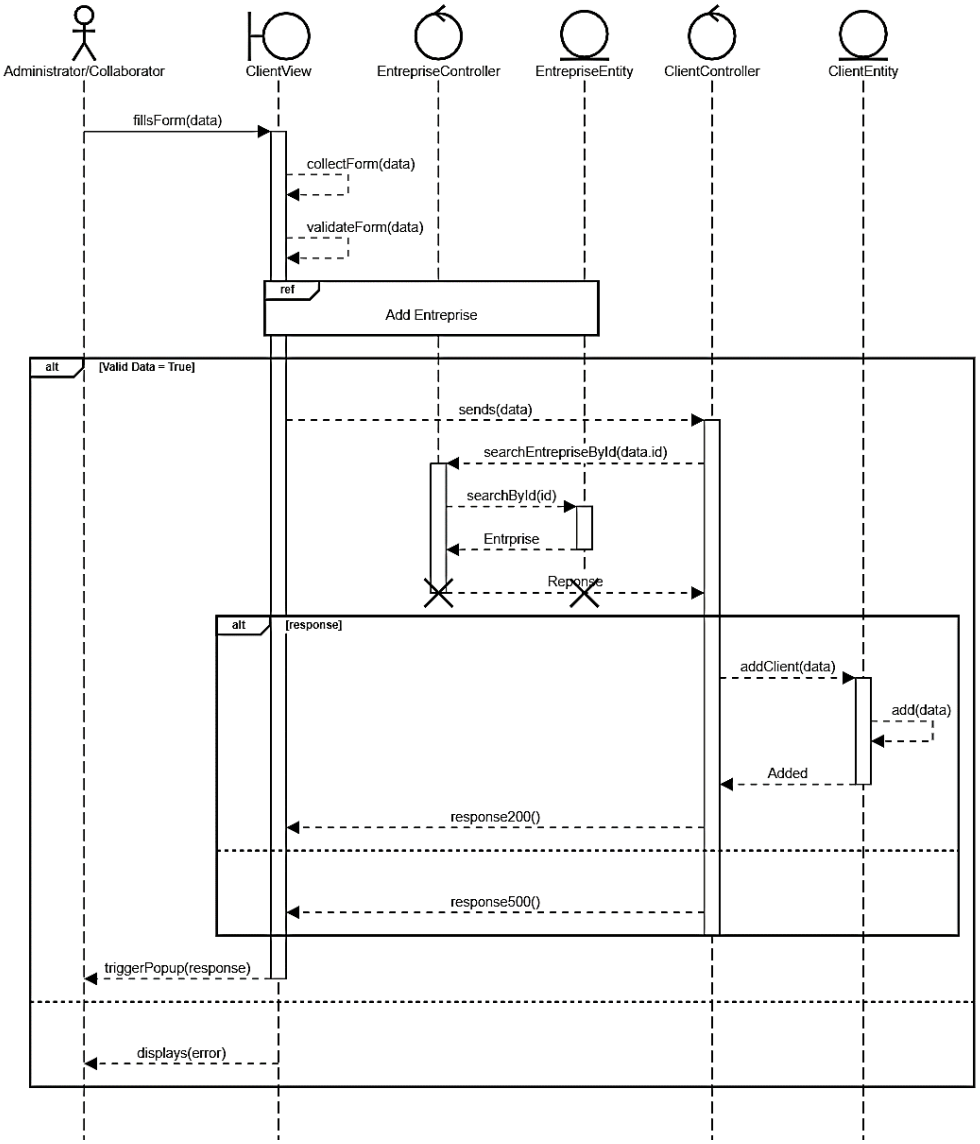


Figure 40 :”Add Client” Sequance Diagram

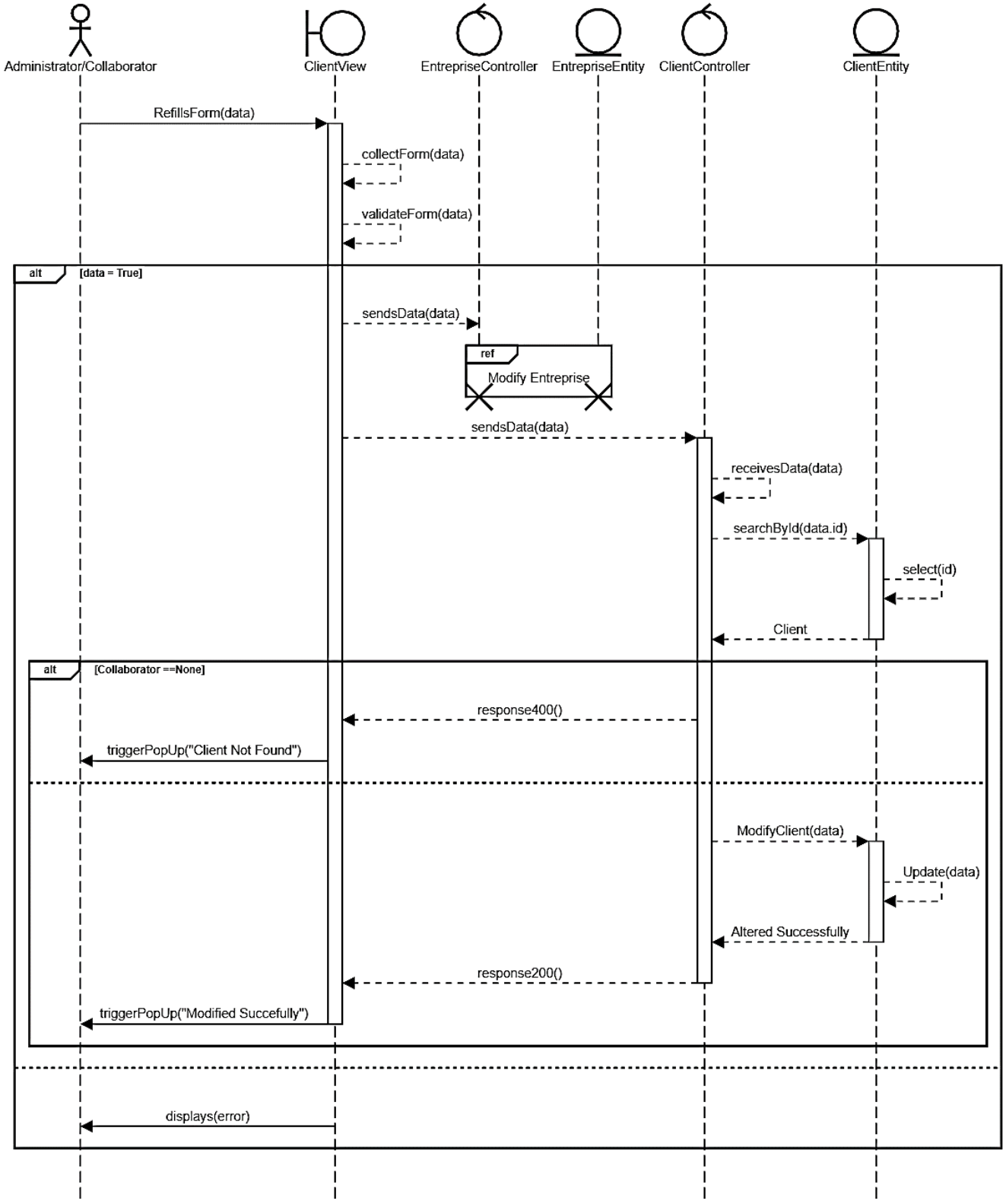


Figure 41:" Modify Client Informations" Sequance Diagram

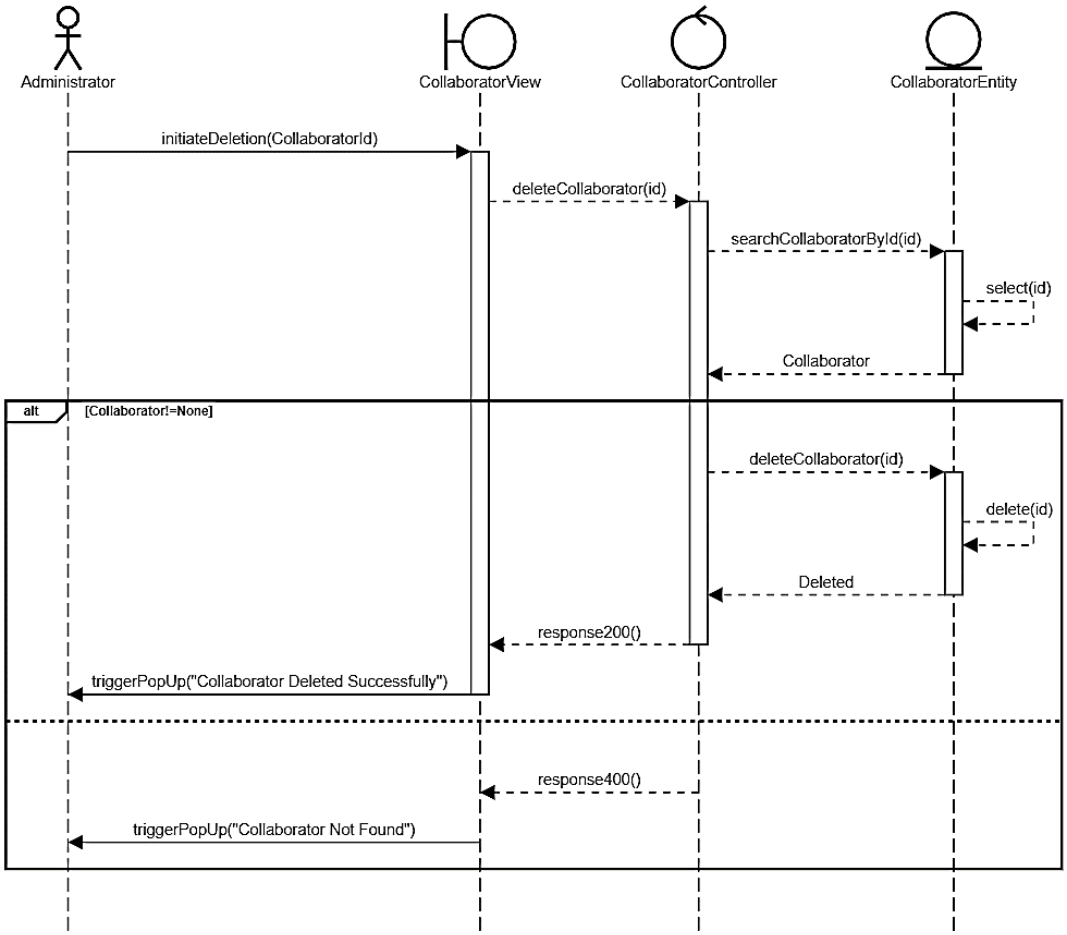


Figure 42 : "Delete Client" Sequance Diagram

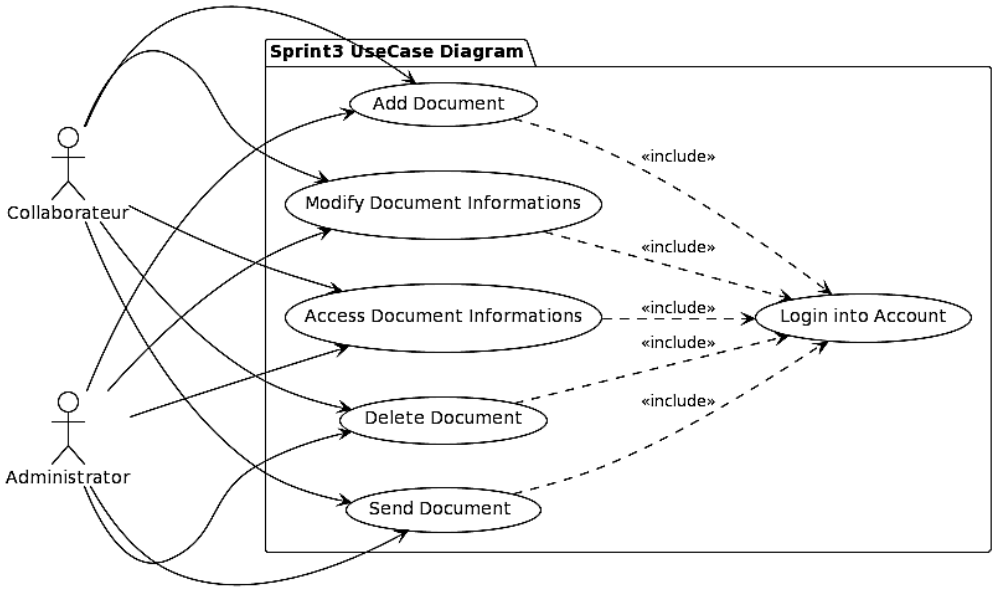


Figure 43 : Story 9 Use Case Diagram

Actor(s)	Administrator / Collaborator
Objectif	Access Document's Information
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on "Documents" Menu • Choose the sub-menu of the document's type
Alternative Scenario	• The session is expired and will automatically logout

Table 21: Use Case Description" Access Document Information"

Actor(s)	Administrator / Collaborator
Objectif	Add Documents
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on "Documents" Menu • Choose the sub-menu of the document's type • Press on "Ajouter *" • Fill the form & upload the document • Submit the form
Alternative Scenario	• The session is expired and will automatically logout • If the form's entries are not correct an error will occur

Table 22 : Use Case Description" Add Document"

Actor(s)	Administrator / Collaborator
Objectif	Modify Document Informations
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on "Documents" Menu • Choose the sub-menu of the document's type • Press on "Modifier" • Rewrite the form & upload the documents • Submit the form
Alternative Scenario	• The session is expired and will automatically logout • If the form's entries are not correct an error will occur

Table 23 : Use Case description" Modify Document Informations"

Actor(s)	Administrator / Collaborator
Objectif	Delete Document
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Documents” Menu • Choose the sub-menu of the Document’s type • Press on “Action Button” • Press on “Effacer” • Confirm the deletion of the document
Alternative Scenario	• The session is expired and will automatically logout • The document is privileged

Table 24 : Use Case Description” Delete Document”

Actor(s)	Administrator / Collaborator
Objectif	Send Document
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Documents” Menu • Choose the sub-menu of the document’s type • Press on “Action Button” • Press on “Envoyer” • Choose the target collaborators • Confirm sending
Alternative Scenario	• The session is expired and will automatically logout • The document is privileged

Table 25: Use Case Description “Send Document”

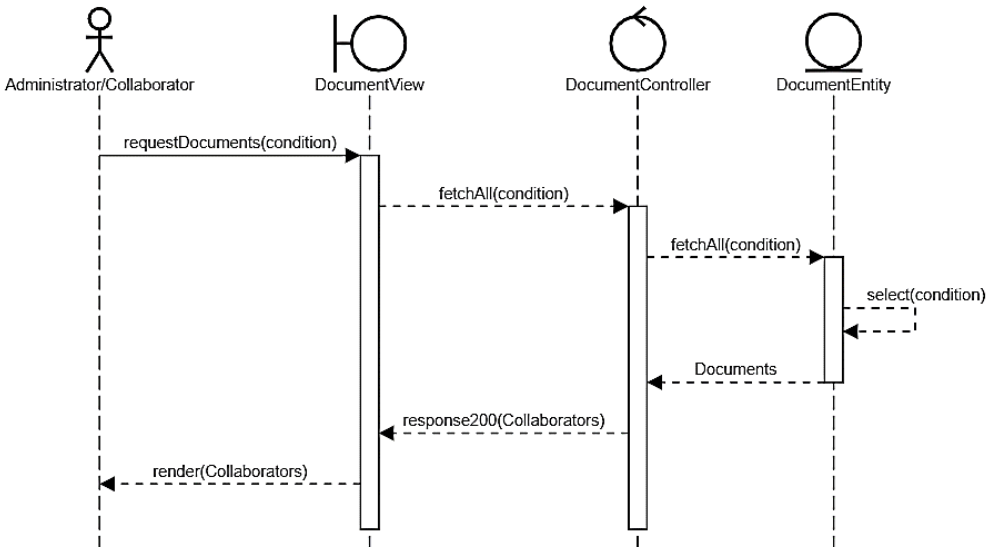


Figure 44 : “Access Document” Sequance Diagram

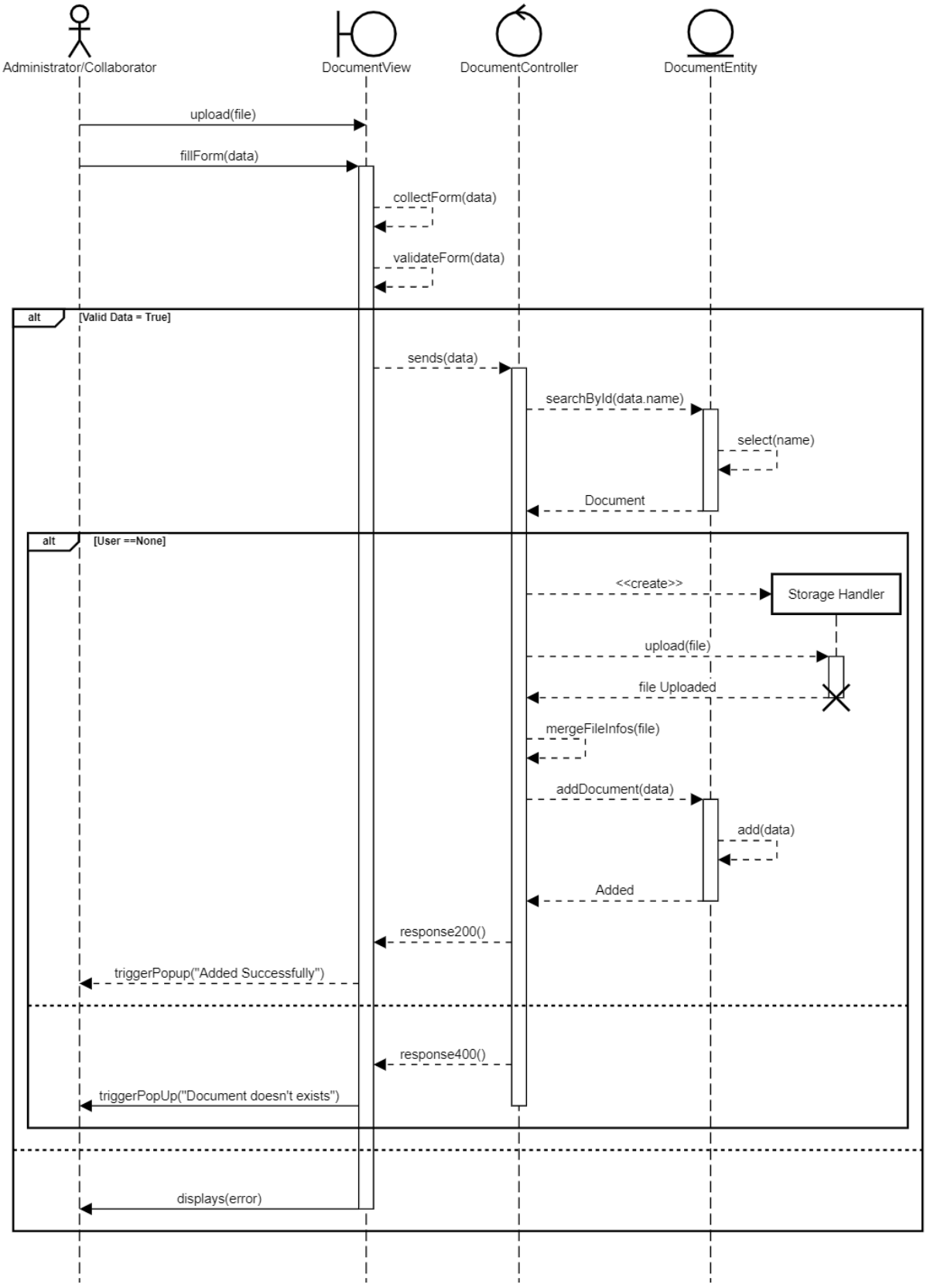


Figure 45:" Add Document" Sequance Diagram

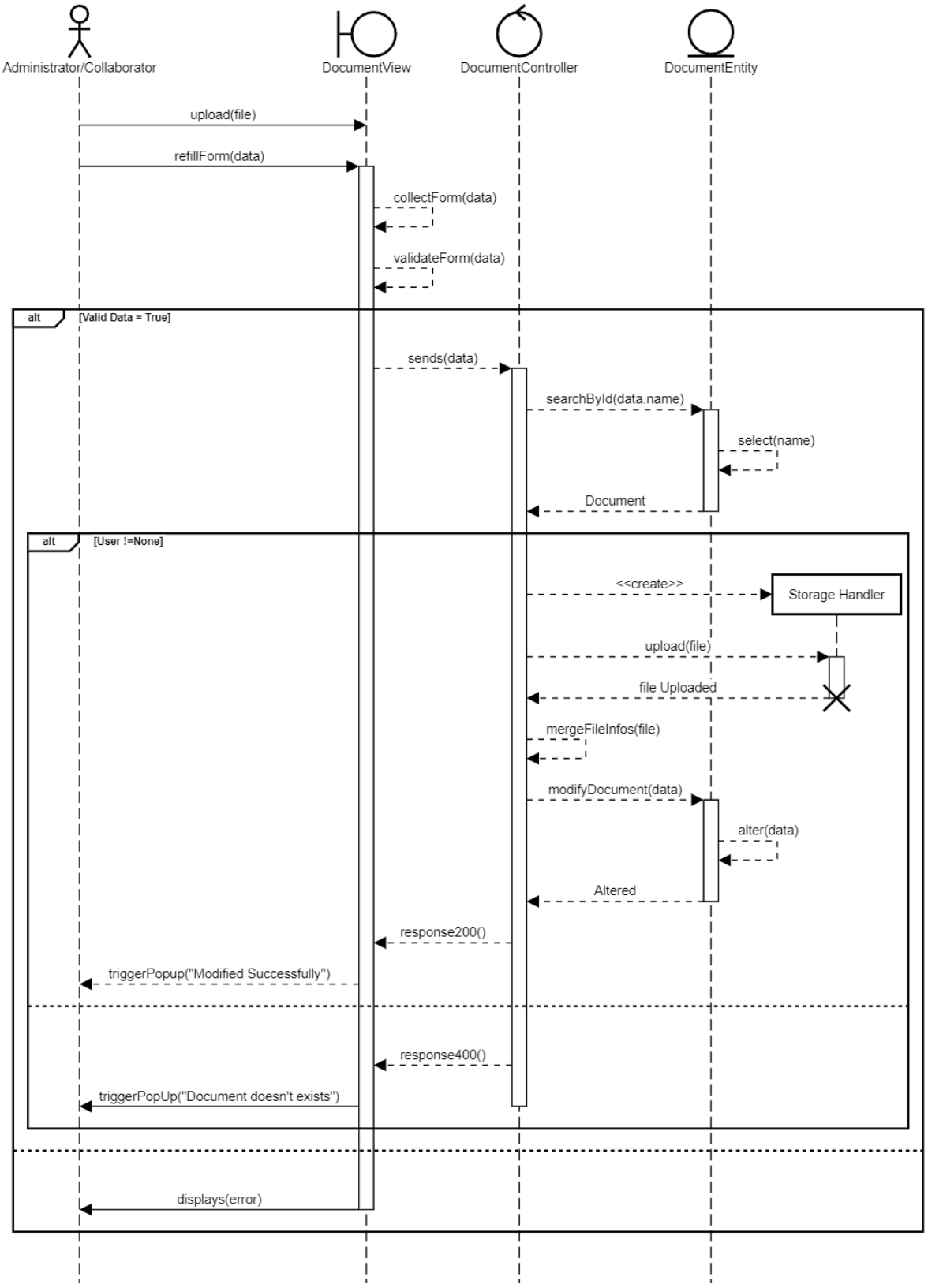


Figure 46: "Modify Document" Sequance Diagram

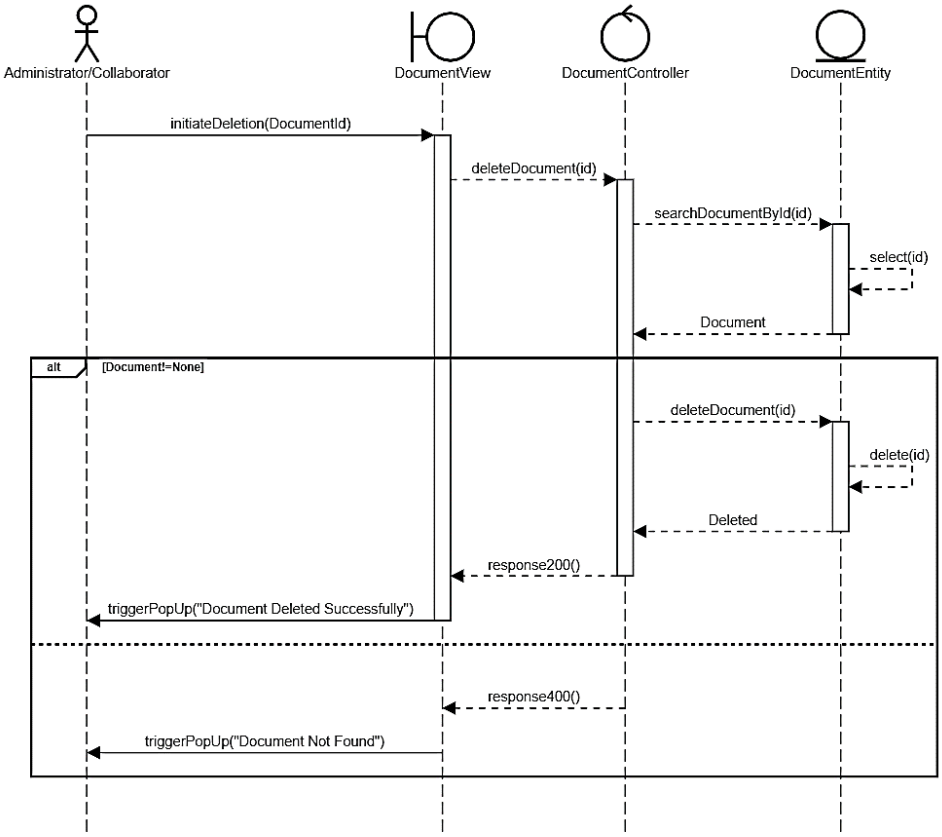


Figure 47 : « Delete Document « Sequance Diagram

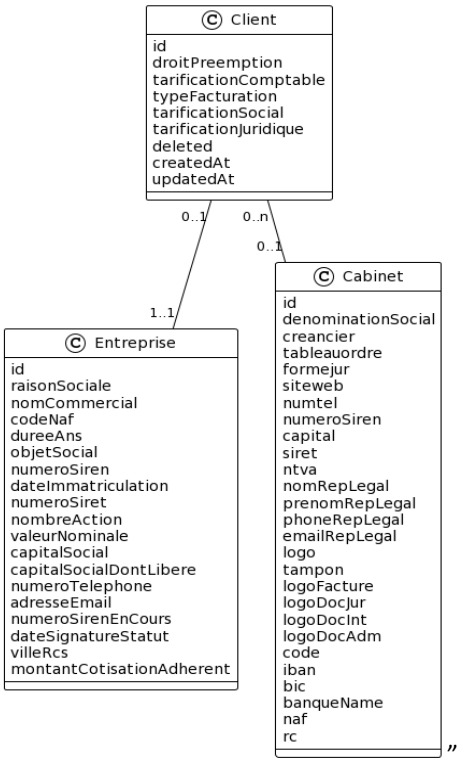


Figure 48 : Story 8 Class Diagram

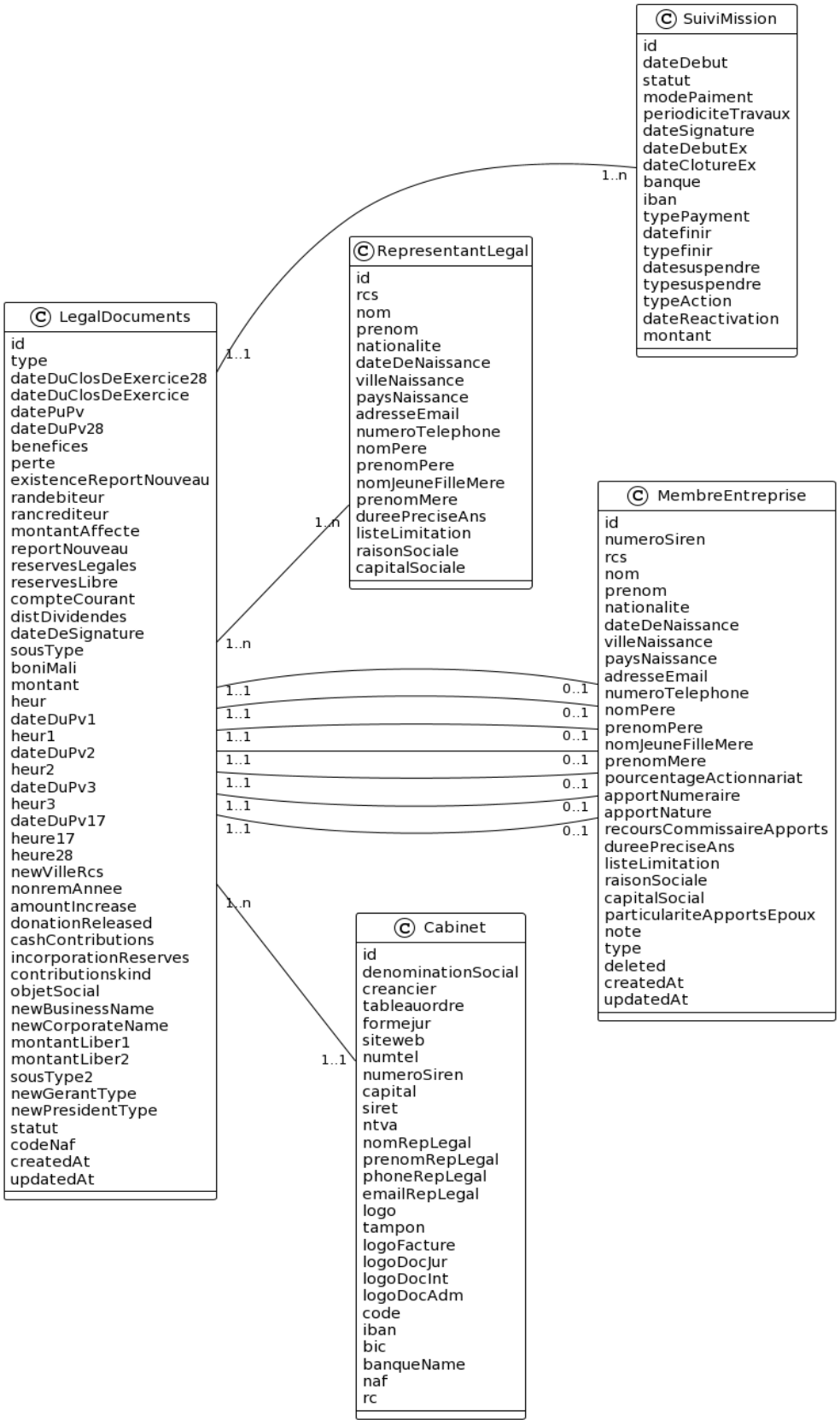


Figure 49 : Story 9 Class Diagram

4.3.6. Sprint Implementation

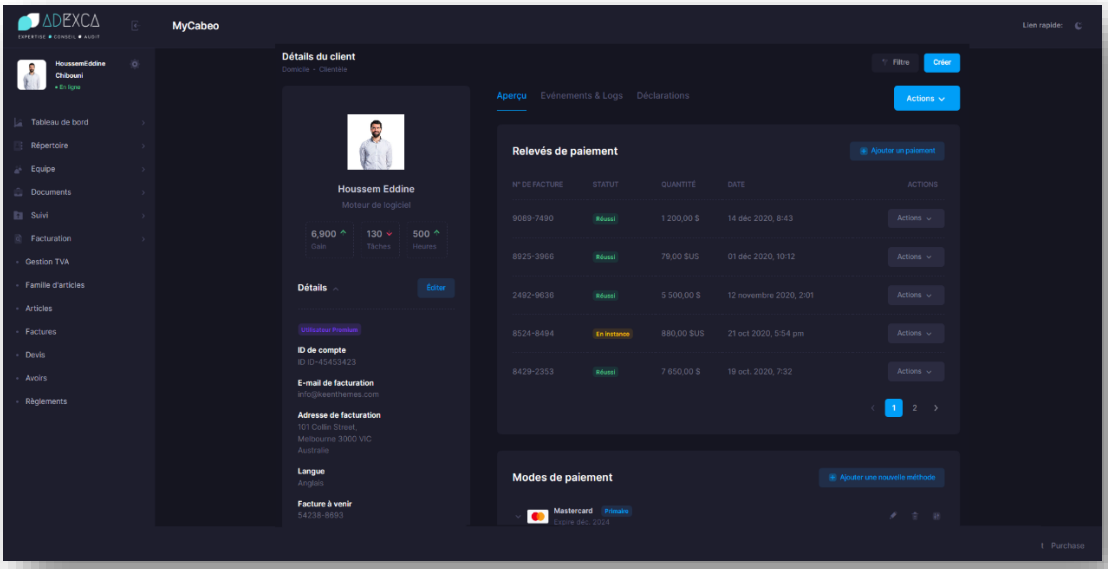


Figure 50: Adding client

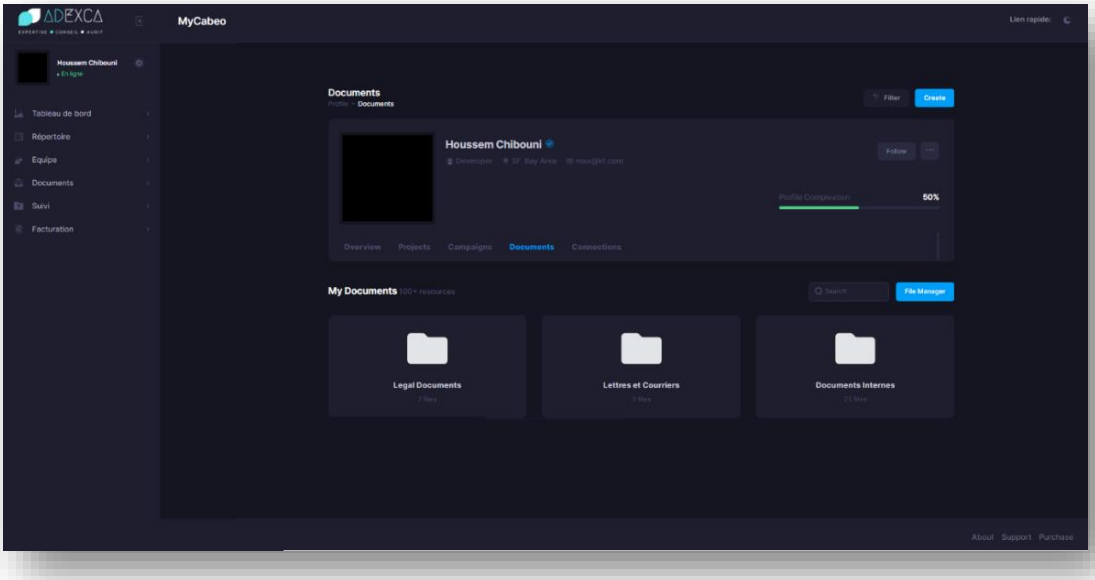


Figure 51 : Accessing Documents

4.3.7. Sprint Review

In this sprint, we focused on the HR (Human Resources) and legal side of the application, we tackled two user stories concerning the clients and their documents.

4.3.8. Sprint Retrospective

The retrospective is a meeting during which the Scrum team uses its experience over the past sprint to improve its organization in order to be more efficient.

What worked well	What can be improved
• CRUD automations • The reusable components	• Variety of forms controls • Services used

Table 26 : Sprint 4 improvement plan

4.4. Sprint 5: «Invoices»

4.4.1. Objectif

The goal of integrating invoices into our system is to streamline the enterprise's billing and invoicing process. We aimed in long terms to automate the generation, management, and tracking of invoices by integrating them, assuring accurate and timely invoicing for products or services given.

4.4.2. Functional Needs

In this Sprint users can perform a specific number of tasks:

- The administrator and collaborators can add an invoice.
- The administrator and collaborators can access an invoice.

- The administrator and collaborators can modify an invoice.
- The administrator and collaborators can delete an invoice.
- The administrator and collaborators can pay an invoice.
- The administrator and collaborators can cancel an invoice.

4.4.3. Sprint Backlog

Story ID	Task ID	Task	Estimated Days	Priority
10	1	Build the API services of “Invoice”	3	7
10	2	Create the “Facturation” components	8	8
10	3	Build the API services of “Taxe”	1	4
10	4	Create the “Taxe” components	2	5

Table 27 : Sprint 5 Backlog

4.4.4. Sprint Analysis

The process of integrating the final component, invoices, into our React and Django application involves several key steps. API endpoints in Django to handle invoice-related operations such as creating, retrieving, updating, and deleting invoices and other additional features like paying and canceling

Next, in the React frontend, we'll develop the user interface components to display and interact with invoices. This includes designing the invoice form for creating new invoices, rendering a list of existing invoices, and providing options for editing or deleting invoices.

The inclusion of the taxes to the application adds a new component that helps with invoices within the system. The tax entity represents a particular tax or tax rate that applies to certain transactions or goods.

The tax entity often includes properties such as the tax name, tax rate, effective date, and any other pertinent tax information. These properties give the information required to appropriately calculate and apply taxes in the application.

Through the relevant API endpoints or user interfaces, the tax entity can be created, changed, or destroyed. The program validates the provided information when creating or modifying a tax entity to verify the accuracy and integrity of tax data. This validation process may include duplicate checks.

The provision of CRUD capability for taxes has been considered to be redundant and unneeded within the scope of this project. As a result, the tax entity and its related CRUD actions will be excluded from the conception.

4.4.5. Sprint Conception

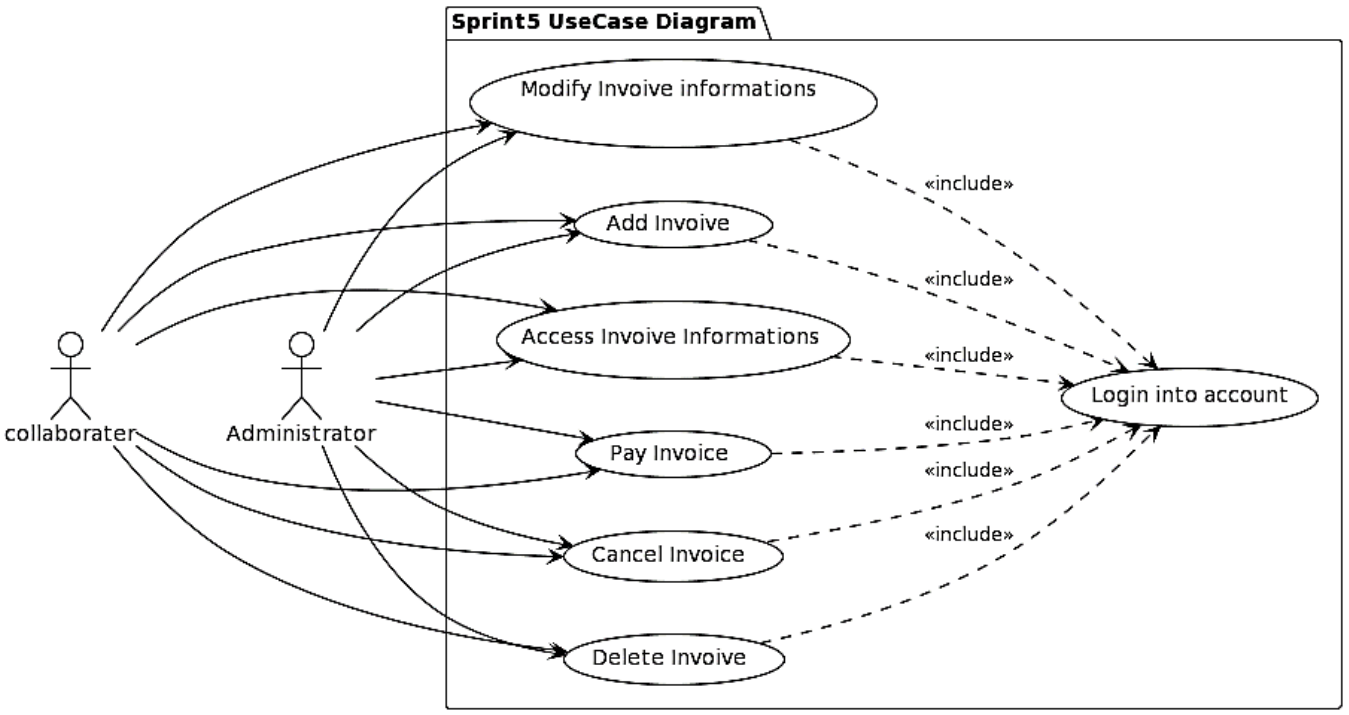


Figure 52 : Sprint 5 Use Case Diagram

Actor(s)	Administrator / Collaborator
Objectif	Add Invoice
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Facturation” Menu • Press on “Ajouter *” • Fill the form & choose “Taxe” • Submit the form
Alternative Scenario	• The session is expired and will automatically logout • If the form’s entries are not correct an error will occur

Table 28: Use Case Description “Add Invoice”

Actor(s)	Administrator / Collaborator
Objectif	Access Invoice Informations
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Facturation” Menu
Alternative Scenario	• The session is expired and will automatically logout

Table 29: Use Case Description “Access Invoice Informations”

Actor(s)	Administrator / Collaborator
Objectif	Modify Invoice Informations
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Facturation” Menu • Press on “Action Button” on specific row • Press on “Modifier” • Fill the form & choose “Taxe” • Submit the form
Alternative Scenario	• The session is expired and will automatically logout • If the form’s entries are not correct an error will occur

Table 30:Use Case Description “Modify Invoice Informations”

Actor(s)	Administrator / Collaborator
Objectif	Delete Invoice
Precondition	User has to be authenticated
Nominal Scenario	• Press on “Facturation” Menu • Press on “Action Button” on specific row • Press on “Supprimer” • Confirm Deletion
Alternative Scenario	• The session is expired and will automatically logout

Table 31 : Use Case Description “Delete Invoice”

Actor(s)	Administrator / Collaborator
Objectif	Pay Invoice
Precondition	User has to be authenticated
Nominal Scenario	• Press on “Facturation” Menu • Press on “Action Button” on specific row • Press on “Pay” • Confirm Payment
Alternative Scenario	• The session is expired and will automatically logout • Credit Card Problem

Table 32 : Use Case Description “Pay Invoice”

Actor(s)	Administrator / Collaborator
Objectif	Cancel Invoice
Precondition	User has to be authenticated
Nominal Scenario	• Open The Sidebar • Press on “Facturation” Menu • Press on “Action Button” on specific row • Press on “Annuler” • Confirm cancellation.
Alternative Scenario	• The session is expired and will automatically logout

Table 33 : Use Case Description “Cancel Invoice”

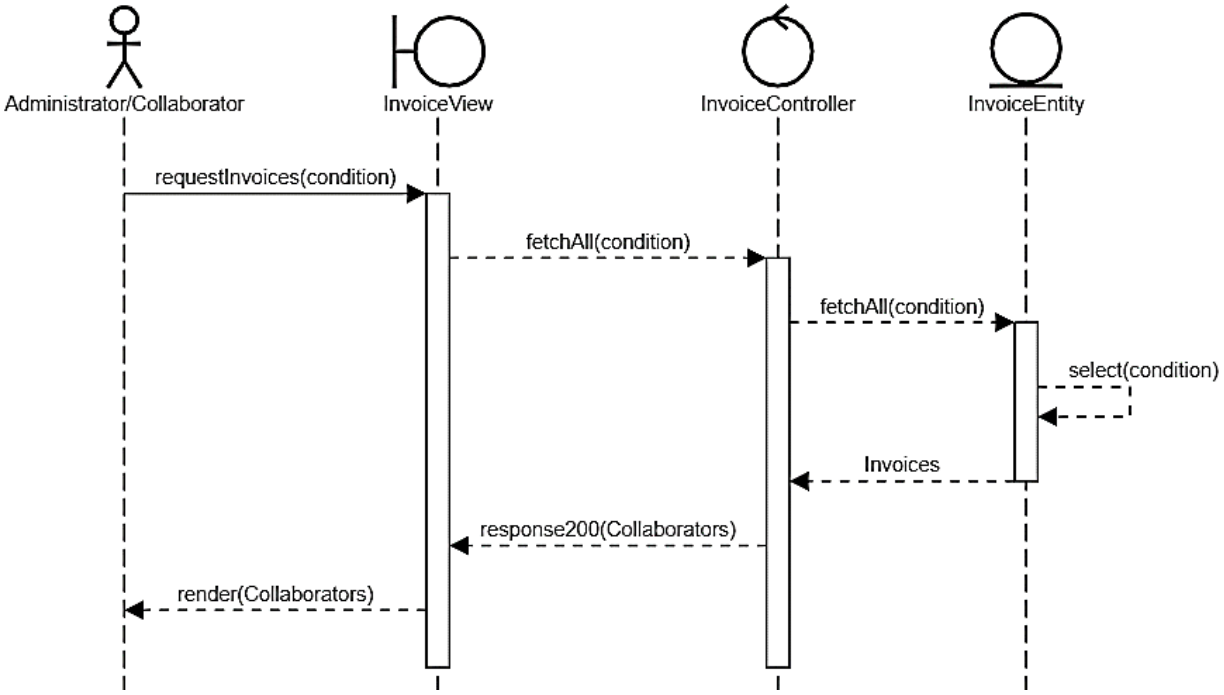


Figure 53 : “Access Invoice Informations” Sequance Diagram

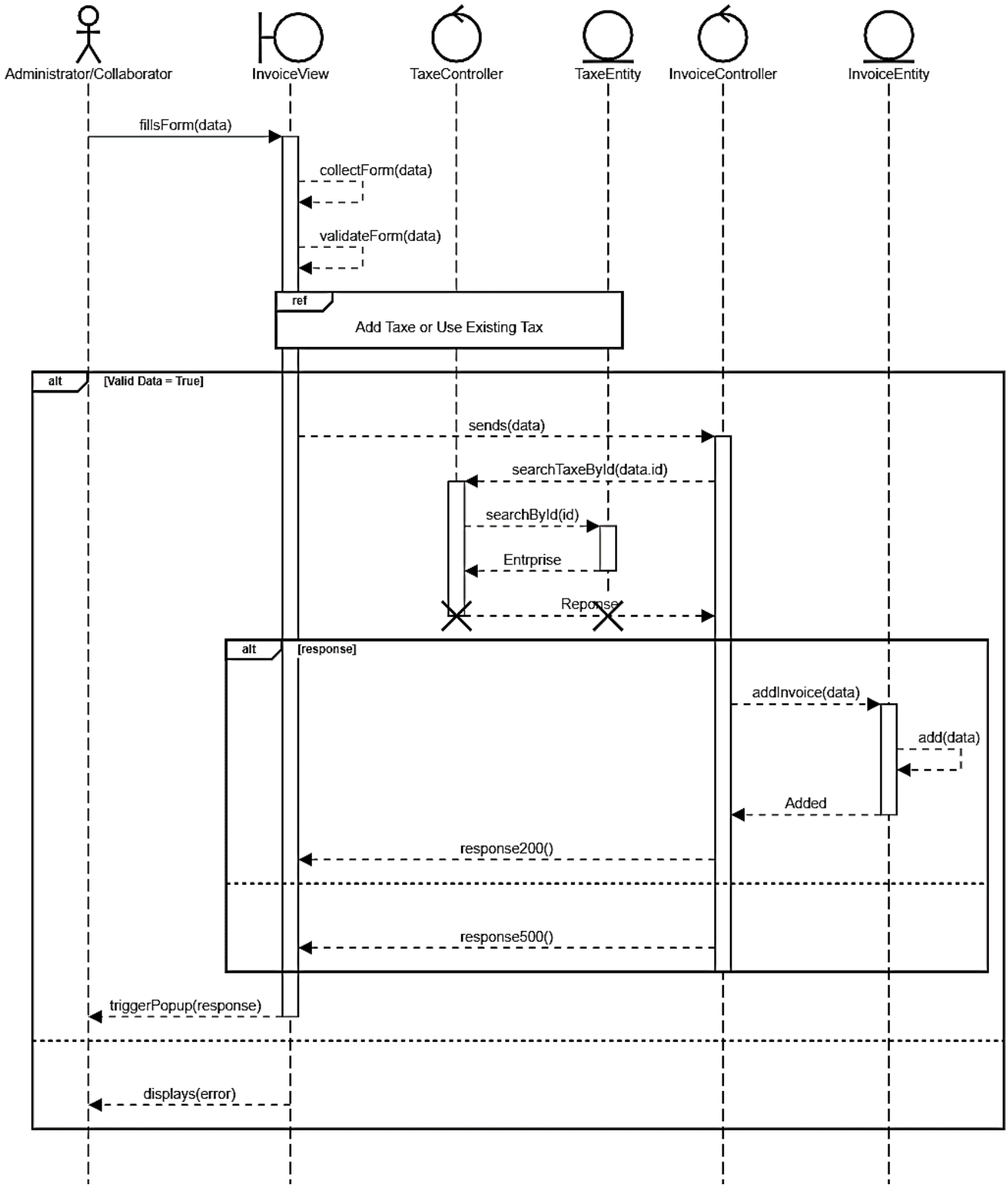


Figure 54 : "Add invoice" Sequance Diagram

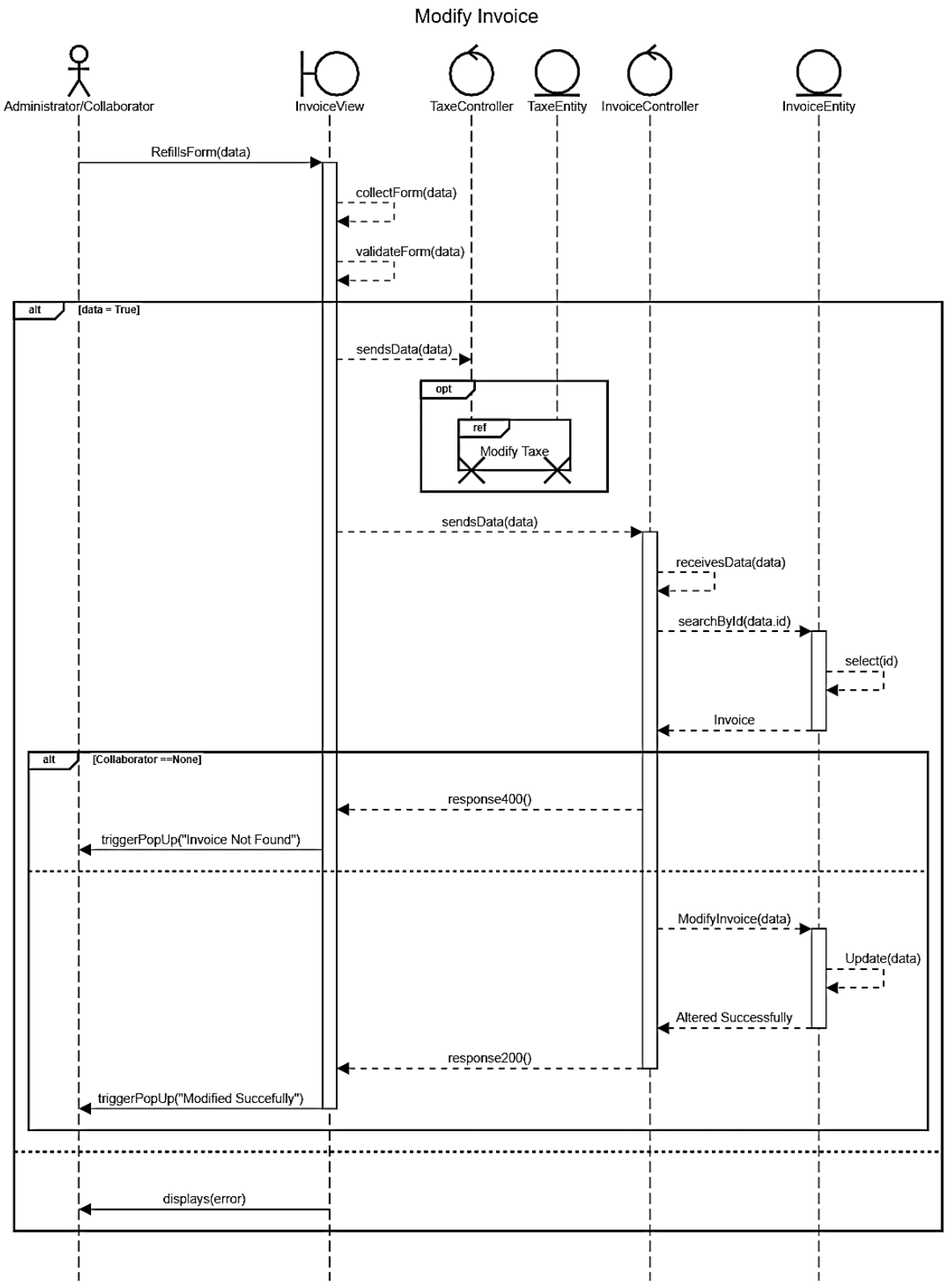


Figure 55 : "Modify Invoice" Sequance Diagram

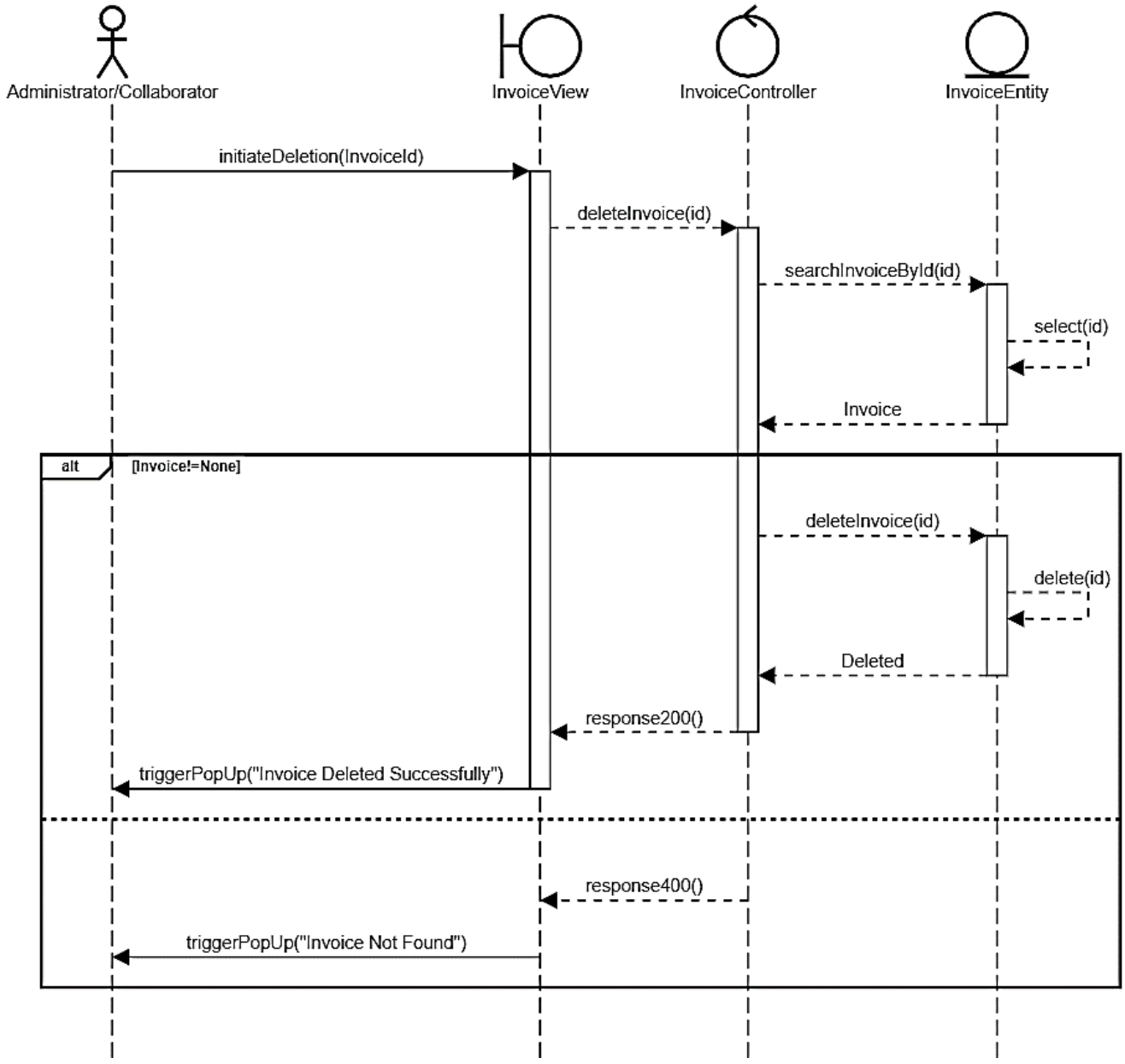


Figure 56 : "Delete Invoice" Sequance Diagram

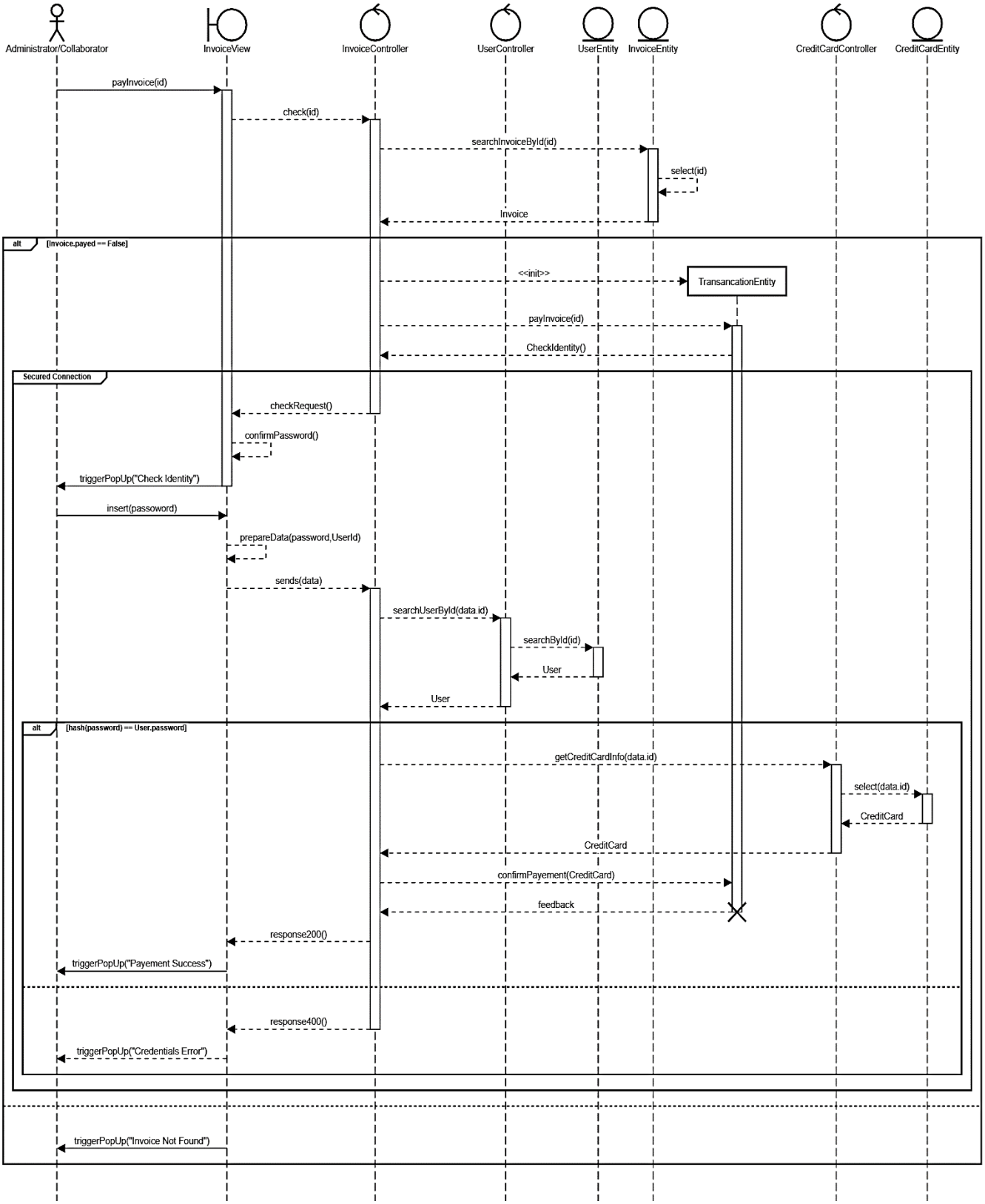


Figure 57 : "Pay Invoice" Sequance Diagram"

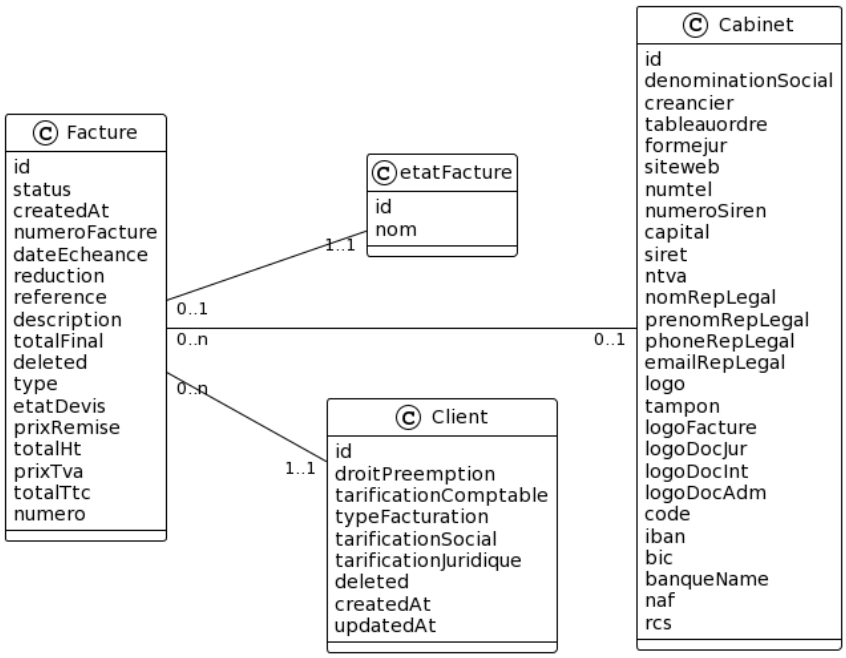


Figure 58 : Sprint 5 Class Diagram

4.4.6. Sprint Implementation

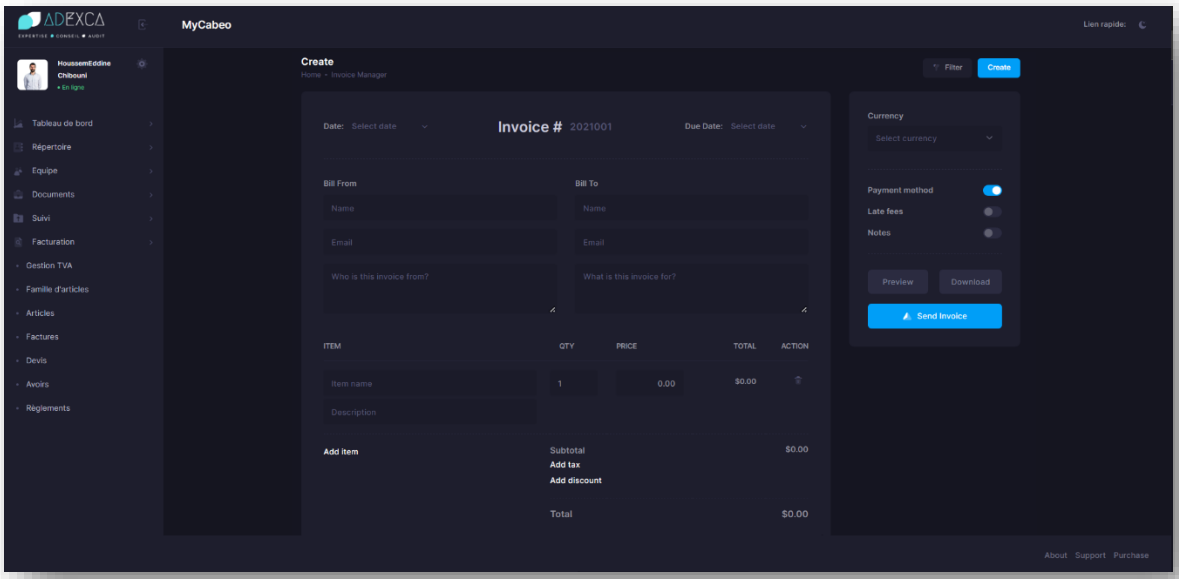


Figure 59: Add Invoice

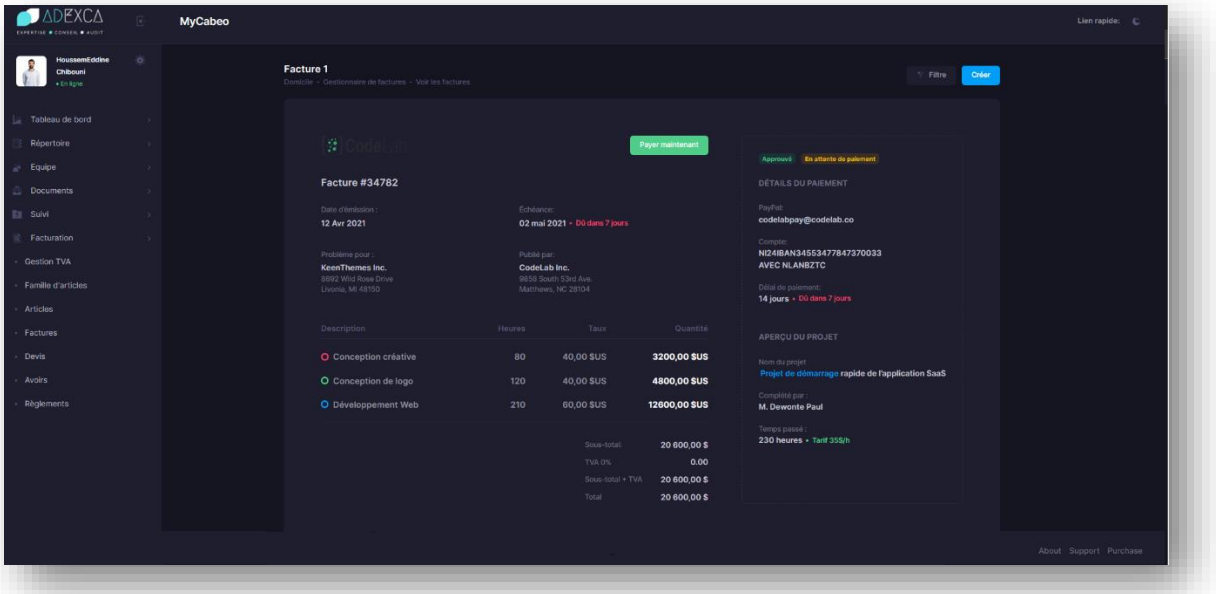


Figure 60: Access Invoice

4.4.7. Sprint Review

During this sprint, our primary focus was on integrating the invoices component into our React and Django application. This integration was a significant milestone, as it aimed to initiate the billing and invoicing process for our enterprise.

4.4.1. Sprint Retrospective

What worked well	What can be improved
• The invoice integration with the system • Taxe entity integration was so smooth	• Time Management and conceptions

Table 34 : Sprint 5 Improvement Plan

4.5. Conclusion

Release 2 settled down 3 sprints: Sprint 3 organized the front-end, security and API communication enhancement, and Sprint 4 focused on the collaborator entity while Sprint 5 focused on the invoicing feature.

5. General Conclusion

Finally, our path through this project has been marked with significant milestones and accomplishments. We made tremendous progress from the first sprint, where we were acquainted with the old system and technology, to the subsequent sprints, where we built and integrated numerous components.

While this project has met the very first initial specifications, it is important to acknowledge that there is room for improvement. As a large-scale application, it is expected that there will be ongoing enhancements and refinements to meet the evolving needs of users.

We successfully gathered legacy system resources from the old application, ensuring a smooth data transfer and maintaining the integrity of our database. Furthermore, we overcame difficulties in integrating the Metronic template, adjusting it to our needs while maximizing its potential.

Our codebase's maintainability and scalability have substantially improved thanks to the organization and we established a well-organized and efficient development environment by using reusable components and executing a methodical strategy.

The commitment and dedication of our team have played a pivotal role in achieving these positive outcomes.

Bibliography

[1] "The application of ISO 9001 to agile software," in *DBLP*, June 2008.

[2] "Scrum Guides," [Online]. Available: https://scrumguides.org/.

[3] "Wikipedia - Unified Modeling Language," [Online]. Available: https://en.wikipedia.org/wiki/Unified_Modeling_Language.

[4] "Wikipedia - Version Control," [Online]. Available: https://en.wikipedia.org/wiki/Version_control.

[5] "Composer," [Online]. Available: https://getcomposer.org/doc/00-intro.md.

[6] "GitHub - FosUserBundle," [Online]. Available: https://github.com/FriendsOfSymfony/FOSUserBundle.

[7] "Symfony - Doctrine," [Online]. Available: https://symfony.com/doc/current/doctrine.html.

[8] "Github - GOS WebSocketBundle," [Online]. Available: https://github.com/GeniusesOfSymfony/WebSocketBundle.

[9] "DomPDF," [Online]. Available: https://github.com/dompdf/dompdf.

[10] "Blowfish (Cipher)," [Online]. Available: https://en.wikipedia.org/wiki/Blowfish_(cipher).

[11] "Hashing Layer," [Online]. Available: https://security.stackexchange.com/questions/110948/password-hashing-on-frontend-or-backend#:~:text=The%20hashing%20should%20be%20done,can%20have%20client%2Dside%20hashing..

[12] "Symfony - Doctrine," [Online]. Available: https://symfony.com/doc/current/doctrine.html.