République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane Mira de Béjaïa
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master Professionel en Informatique

Option : Génie Logiciel

Thème

Conception et réalisation d'une application web et mobile **Bignova Immo** pour la gestion et le suivi des projets immobiliers

Présenté par :

TAGHZOUIT Ahmed

Soutenu devant le jury composé de :

Présidente	Mme. EL BOUHISSI BRAHAMI Houda	M.C.A	U. A/Mira Béjaïa
Examinateur	Mr. MOHAMMEDI Mohamed	M.C.A	U. A/Mira Béjaïa
Encadrant	Mr. ALLEM Khaled	M.A.A	U. A/Mira Béjaïa
Co-encadrant	Mr. BEKKA Reda	INVITÉ	EURL Bignova

Promotion 2023 – 2024

Remerciements

Mes remerciements s'adressent à mon encadrant Monsieur **ALLEM KHALED**, pour avoir accepté de diriger ce travail. Son soutien, sa clairvoyance, ses compétences, ainsi que son infinie disponibilité nous ont été d'une aide inestimable.

Je tiens également à remercier sincèrement les membres du jury Madame **EL BOUHISSI BRAHAMI HOUDA** et Monsieur **MOHAMMEDI MOHAMED**, pour avoir accepté d'évaluer ce travail. Leur disponibilité, leur bienveillance ainsi que leurs remarques pertinentes ont grandement enrichi ce mémoire.

Je tiens également à exprimer ma profonde gratitude à toute l'équipe de l'entreprise **BIGNOVA** pour leur contribution essentielle à notre projet. Leur collaboration, leur expertise et leur soutien constant ont considérablement enrichi mon expérience pratique et ont été cruciaux pour la réalisation de ce mémoire.

Je remercie en particulier Monsieur **BEKKA REDA** dont l'engagement et les conseils avisés ont grandement facilité l'avancement de notre travail.

Enfin, je remercie également ma famille et mes amis pour leur soutien permanent, qui m'a été bien utile.

Dédicaces

À ma famille et à mes amis,

Je dédie ce travail à ma famille et à mes amis qui m'ont soutenu tout au long de mon parcours. Votre présence, vos encouragements et votre confiance ont été essentiels pour surmonter les moments difficiles. Grâce à vous, j'ai pu avancer avec sérénité et détermination.

À Monsieur **ALLEM Khaled**,

Je vous dédie ce mémoire en signe de reconnaissance pour votre accompagnement, vos conseils avisés et votre encadrement bienveillant tout au long de ce projet. Votre soutien a été d'une grande importance dans l'aboutissement de ce travail.

À l'équipe de **Bignova**,

Je tiens à vous dédier ce travail en témoignage de l'impact positif que votre accompagnement a eu sur mon parcours. Cette expérience professionnelle à vos côtés a été enrichissante tant sur le plan technique qu'humain.

Enfin, à toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce mémoire, je vous adresse toute ma gratitude.

Merci à tous.

Table des Matières

Table des figures

Liste des tableaux

Liste des Abréviations

API Application Programming Interface
HTTP Hypertext Transfer Protocol
IDE Integrated Development Environment Standardization
JS JavaScript
JSON JavaScript Object Notation
JWT JSON Web Token
NoSQL Not Only SQL
SQL Structured Query Language
UI User Interface
UML Unified Modeling Language
UP Unified Process
URL Uniform Resource Locator

Introduction Générale

Depuis quelques décennies, les avancées technologiques ont profondément transformé les méthodes de travail et les pratiques professionnelles dans de nombreux secteurs. Le secteur immobilier, pilier essentiel de l'économie, n'a pas échappé à cette révolution. Les entreprises immobilières, autrefois dépendantes de processus manuels et de gestion sur papier, adoptent désormais des solutions numériques pour améliorer leur efficacité, leur précision et leur capacité de gestion.

Face à cette complexité croissante et à la diversité des tâches liées à la gestion des projets immobiliers, les entreprises sont confrontées à plusieurs défis. Comment centraliser et optimiser efficacement la gestion de leurs opérations tout en assurant une coordination fluide entre les différents acteurs ? Comment offrir une interface utilisateur conviviale qui répond aux besoins spécifiques de chaque rôle au sein de l'organisation, tout en garantissant la sécurité des données et une gestion efficace des droits d'accès ?

L'objectif principal de notre étude est de développer une application web et mobile moderne et intuitive pour gérer efficacement les projets immobiliers. Cette application vise à centraliser les diverses tâches de gestion, à offrir une interface utilisateur simple et efficace, et à répondre aux besoins spécifiques des entreprises immobilières. En intégrant des fonctionnalités clés et une gestion avancée des droits d'accès, cette application se veut être un outil indispensable pour les entreprises immobilières modernes.

Le présent mémoire est structuré en quatre chapitres, chacun abordant une étape clé du développement de cette application :

Chapitre 1 : Contexte du projet et méthodologie de conception
Ce premier chapitre contextualise le projet en présentant l'état actuel du marché immobilier et les défis auxquels il est confronté. Il explore également la méthodologie de développement utilisée, en particulier la méthode Unified Process (UP), pour structurer et guider le développement de l'application.

Chapitre 2 : Spécification et analyse des besoins
Ce chapitre est dédié à l'identification des acteurs clés et à la description détaillée des besoins fonctionnels et non fonctionnels de l'application. Cette analyse permet de définir précisément les attentes des utilisateurs et les contraintes techniques à respecter.

Chapitre 3 : Conception de l'application
Dans ce chapitre, nous détaillons la conception technique de l'application, en décrivant son architecture, ses composants principaux et les technologies utilisées. Cette section vise à fournir une vue d'ensemble claire et précise du système, facilitant ainsi la compréhension et la mise en œuvre du projet.

Chapitre 4 : Réalisation

Le dernier chapitre se concentre sur la phase de réalisation de l'application. Il couvre le développement des différentes fonctionnalités, les outils et langages de programmation employés, ainsi que les défis rencontrés et les solutions apportées. La présentation de l'application se fait à travers des interfaces utilisateurs et des cas d'utilisation pratiques.

Enfin, une conclusion générale vient clore ce mémoire en résumant les apports du projet et en ouvrant sur des pistes d'évolution futures de l'application.

1

Contexte du projet et Méthodologie de conception

1.1 Introduction

Ce chapitre présente le contexte du projet en analysant les solutions existantes de gestion de projets immobiliers. L'objectif est d'identifier les forces et les faiblesses des outils actuels afin de mieux comprendre les besoins non satisfaits des utilisateurs. Par la suite, nous définirons la méthodologie de conception adoptée pour développer notre application. Cette approche structurée nous permettra de répondre efficacement aux exigences du marché et d'assurer une gestion optimale du projet.

1.2 Présentation de l'entreprise d'accueil

BigNova, une boîte de développement créative et réactive basée à Béjaïa, Algérie et à Paris, France. Spécialistes en création de sites internet, applications web et mobile, web design, graphic design, conseils et optimisation digitale,
BigNova est une boîte fondée et dirigée par des technologues du web , mobile et de design [1].

FIGURE 1.1 – Logo de Bignova.

1.3 Étude et Critique de l'Existant

Dans le domaine de la gestion des bâtiments et des chantiers, il existe une multitude d'applications offrant une variété de fonctionnalités adaptées aux besoins spécifiques des entreprises. Ces solutions logicielles peuvent aller de la simple gestion des devis et factures à des systèmes complets intégrant la gestion des stocks, la planification des ressources humaines, et le suivi en temps réel des projets. La diversité des offres permet aux entreprises de choisir des outils qui répondent précisément à leurs exigences opérationnelles.

Dans cette section, nous allons présenter des applications, tant au niveau national qu'international, afin d'illustrer la richesse et la diversité des options disponibles. Nous mettrons en lumière leurs principales fonctionnalités et avantages pour aider à comprendre comment elles peuvent répondre aux différents besoins. De plus, nous analyserons également les inconvénients de chaque solution pour fournir une vue d'ensemble équilibrée et aider les entreprises à faire des choix éclairés.

1.3.1 À l'international

— **EBP Bâtiment**

C'est un logiciel de gestion bâtiment de l'éditeur EBP. Dédié aux professionnels du bâtiment, ce logiciel s'adapte à vos spécificités métiers. EBP Bâtiment fait le lien entre l'activité commerciale, la planification des chantiers et le suivi de la trésorerie. Performant et adapté aux PME, ce logiciel s'impose comme une référence en matière de solution de gestion bâtiment [2].

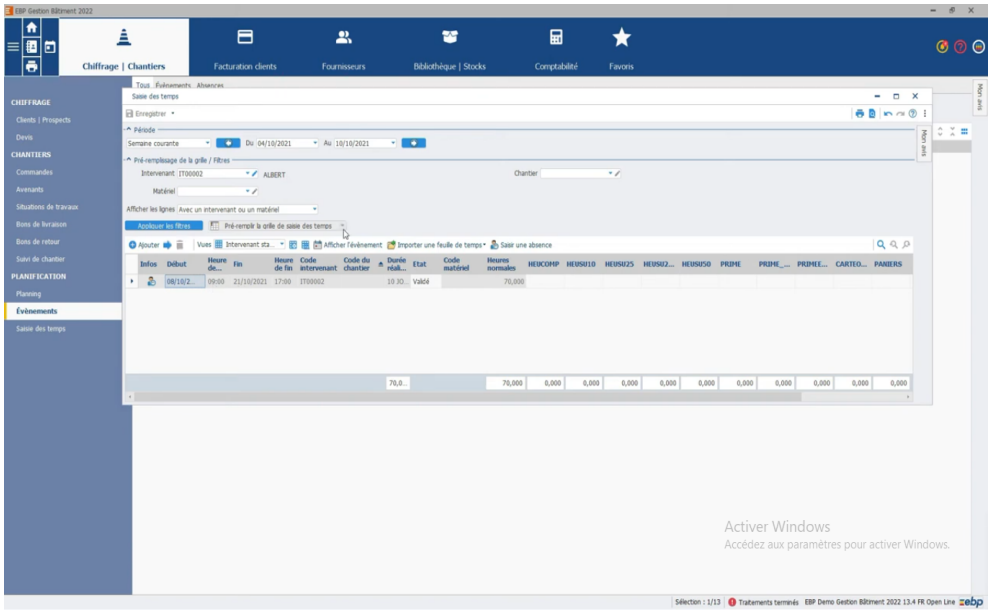

FIGURE 1.2 – Interface EBP Bâtiment.

Avantages

— **Gestion complète des chantiers** : EBP Bâtiment permet une gestion complète des chantiers, incluant le suivi des travaux, la gestion des achats et des stocks, ainsi que le suivi financier détaillé.

— **Gestion des contacts** : Vous pouvez stocker les informations de base des clients et des fournisseurs dans EBP Bâtiment, facilitant ainsi la gestion des relations clients et fournisseurs.

— **Avancement des ventes** : EBP Bâtiment suit principalement les aspects financiers et matériels des chantiers. Pour un suivi détaillé de l'avancement des ventes de lots, une solution complémentaire pourrait être nécessaire.

Inconvénients

— **Design** : L'interface utilisateur d'EBP Bâtiment peut paraître vieillissante et manquer d'intuitivité, ce qui peut entraîner des difficultés pour les utilisateurs lors de la navigation et de l'utilisation des fonctionnalités.

— **Suivi par les clients** : Les clients n'ont pas accès à un suivi détaillé de l'avancement des projets, ce qui limite la transparence et le contrôle qu'ils peuvent exercer.

— **Adaptation au marché algérien** : EBP Bâtiment n'est pas spécifiquement conçu pour le marché algérien, ce qui peut entraîner des limitations dans la gestion des particularités locales et des exigences spécifiques de ce marché.

1.3.2 Àu Niveau National

Au niveau national, plusieurs applications répondent à des besoins spécifiques dans la gestion immobilière. Par exemple, "Fatoura" se distingue dans la gestion des produits, de la facturation et des dépenses, offrant des outils pour suivre et gérer ces aspects financiers de manière efficace [3].

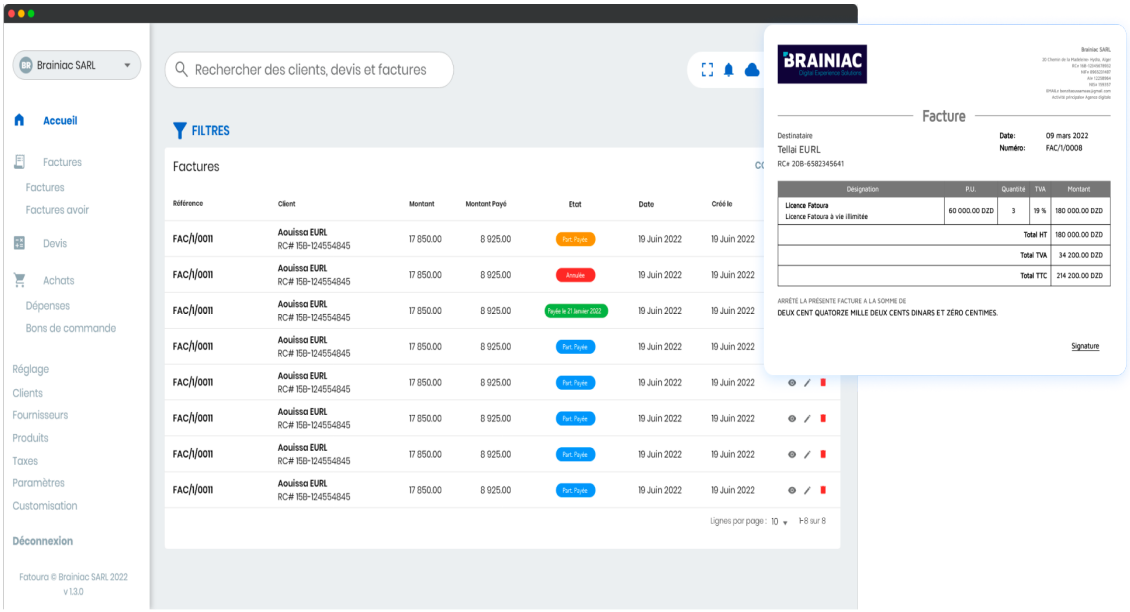

FIGURE 1.3 – Interface de Fatoura.

D'autre part, "GP Immo" de YlarSoft est dédié à la gestion des ventes, des clients, des lots et des réservations, facilitant le suivi des opérations immobilières et des interactions avec les clients [4].

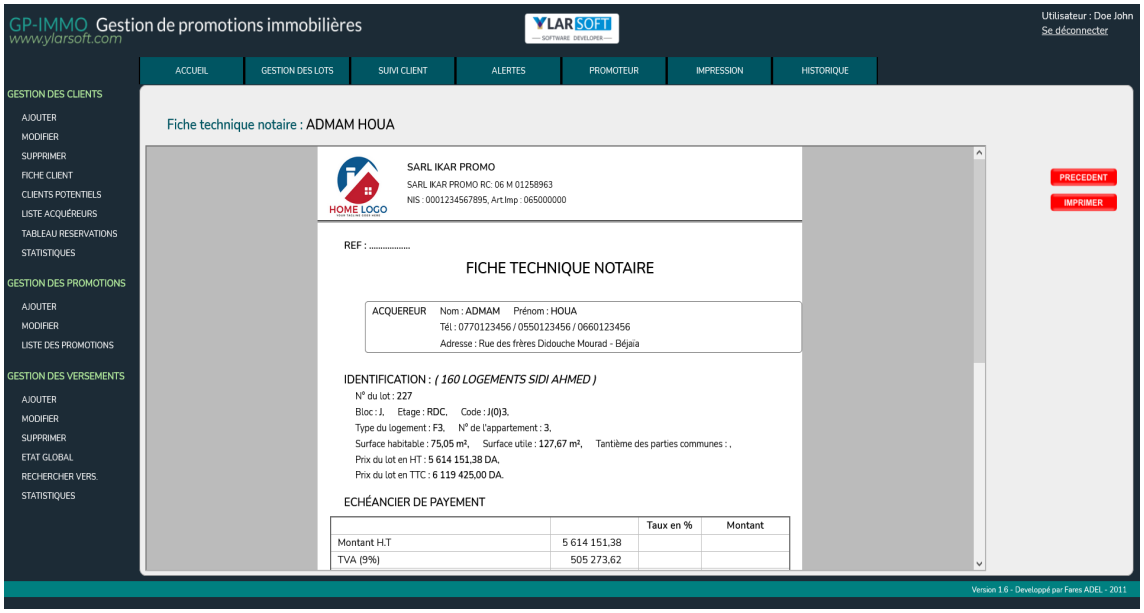

FIGURE 1.4 – Interface de GP-IMMO.

1.4 Problématique

Au niveau national, bien que plusieurs applications web répondent à des besoins spécifiques dans la gestion immobilière, les utilisateurs se trouvent souvent obligés d'utiliser plusieurs solutions pour satisfaire l'ensemble de leurs exigences. En effet, certaines applications se concentrent sur la gestion des produits, de la facturation et des dépenses, tandis que d'autres se spécialisent dans la gestion des ventes, des clients, des lots et des réservations.
Cette fragmentation oblige les entreprises à adopter plusieurs outils pour couvrir tous leurs besoins, ce qui peut entraîner une perte d'efficacité, des difficultés de coordination et une augmentation des coûts de gestion. Ainsi, l'absence d'une solution intégrée qui combine efficacement ces divers aspects de la gestion immobilière souligne la nécessité de développer des applications plus complètes et adaptées aux besoins globaux du secteur.

1.5 Solution proposée

Après cette étude nous proposons le développement de notre propre application, "Bignova Immo". Cette solution vise à intégrer plusieurs fonctionnalités essentielles pour répondre aux besoins diversifiés des entreprises immobilières.

"Bignova Immo" offrira une gestion complète des clients, des réservations et des paiements, centralisant ainsi toutes les informations nécessaires pour une gestion efficace des relations clients. En outre, elle permettra la gestion des produits et des dépenses, offrant des outils pour suivre et optimiser les aspects financiers de l'entreprise. De plus, notre application proposera une gestion intégrée des projets et des employés, facilitant la coordination et le suivi des chantiers en cours. Elle permettra également une utilisation multi-utilisateurs, avec une gestion avancée des accès pour différents administrateurs, assurant une répartition efficace des rôles et des responsabilités au sein de l'organisation.

Pour améliorer encore l'expérience utilisateur, "Bignova Immo" inclura également une application mobile. Cette application permettra aux clients d'accéder facilement à des informations sur l'avancement de leurs projets et de gérer leurs paiements en toute simplicité. Grâce à cette solution complète, nous visons à combler les lacunes des applications existantes et à offrir une gestion immobilière plus intégrée et efficiente.

Voici l'identité visuelle de notre application "Bignova Immo" :

FIGURE 1.5 – Logo de Bignova Immo.

1.5.1 Pourquoi une application web ?

Le choix de développer une application web pour "Bignova Immo" repose sur plusieurs avantages clés :

Accessibilité : Elles sont accessibles depuis n'importe quel appareil connecté à Internet, ce qui permet une grande flexibilité et une mobilité accrue pour les utilisateurs.

Mises à jour instantanées : Les mises à jour et les améliorations peuvent être déployées instantanément, assurant que tous les utilisateurs disposent toujours de la version la plus récente de l'application sans avoir à effectuer des mises à jour manuelles.

Compatibilité multiplateforme : Une application web fonctionne de manière transparente sur différents systèmes d'exploitation (Windows, macOS, Linux) et navigateurs web, offrant une expérience utilisateur cohérente quel que soit l'environnement technique.

Sécurité et sauvegarde des données : Les applications web centralisent les données sur des serveurs sécurisés, garantissant ainsi que les données peuvent être facilement récupérées en cas de problème.

1.5.2 Pourquoi une application mobile ?

Nous avons choisi de développer une application mobile afin de garantir une accessibilité facile et immédiate pour les utilisateurs. Les applications mobiles permettent un accès direct depuis les smartphones et les tablettes, offrant ainsi une grande flexibilité aux clients pour consulter l'avancement de leurs projets et gérer leurs paiements en tout lieu et à tout moment.

De plus, les notifications push intégrées aux applications mobiles permettent d'envoyer des mises à jour instantanées et des alertes importantes directement sur les appareils des utilisateurs, assurant ainsi qu'ils restent informés des évolutions en temps réel sans avoir à vérifier constamment leur compte.

1.6 Processus de Développement

Un processus de développement logiciel est un ensemble d'activités nécessaires pour transformer les exigences d'un utilisateur en un système logiciel [5]. Il définit les étapes, les rôles, les outils et les techniques à suivre pour transformer une idée en un produit logiciel fonctionnel.

1.6.1 Processus unifié (UP)

Définition :

Le Unified Process (UP) est un cadre méthodologique pour le développement logiciel, structuré en phases itératives et incrémentales.Le Unified Process est plus qu'un simple processus, c'est un cadre méthodologique générique qui peut être spécialisé pour une très large gamme de systèmes logiciels, couvrant différents domaines d'application, types d'organisations, niveaux de compétence et tailles de projets. [5].

Caractéristiques :

— **Phases** : Le UP est divisé en quatre phases principales : Inception, Élaboration, Construction, et Transition. Chaque phase se concentre sur différents aspects du développement logiciel.

— **Itératif et incrémental** : Le développement se fait par itérations, permettant d'améliorer et d'affiner le produit au fil du temps avec des versions successives.

— **Gestion des risques** : Une des priorités du UP est l'identification précoce et la gestion proactive des risques associés au projet.

— **Modularité** : Le processus est flexible et peut être adapté en fonction des besoins spécifiques du projet et des retours obtenus au cours du développement.

Phases du cycle de vie :

— **Phase d'initialisation** : Cette phase permet de définir la vision du projet, sa portée, sa faisabilité, et son business case, afin de pouvoir décider au mieux de sa poursuite ou de son arrêt.

— **Phase d'élaboration** : Elle poursuit trois objectifs principaux en parallèle : identifier et décrire la majeure partie des besoins des utilisateurs, construire l'architecture de base du système, et lever les risques majeurs du projet.

— **Phase de construction** : Cette phase consiste principalement à concevoir et implémenter l'ensemble des éléments opérationnels (autres que ceux de l'architecture de base). C'est la phase la plus consommatrice en ressources et en effort.

— **Phase de transition** : Elle permet de faire passer le système informatique des mains des développeurs à celles des utilisateurs finaux.

Enchaînement des activités :

— **Exigences** : Présente le système du point de vue de l'utilisateur et recense les besoins fonctionnels et non fonctionnels.

— **Analyse** : Définit une spécification complète des besoins issus des cas d'utilisation et une compréhension des exigences du client.

— **Conception** : Décrit les différentes vues (fonctionnelles, dynamiques et statiques) d'une architecture, à travers un diagramme de classe et d'interaction, etc.

— **Implémentation** : Réalise l'implémentation des résultats de conception, l'intégration du système et des classes.

— **Test** : Planifie et met en œuvre les tests pour assurer la qualité du système développé.

Le Processus Unifié propose cet enchaînement d'activités, représenté dans la figure ci-dessous :

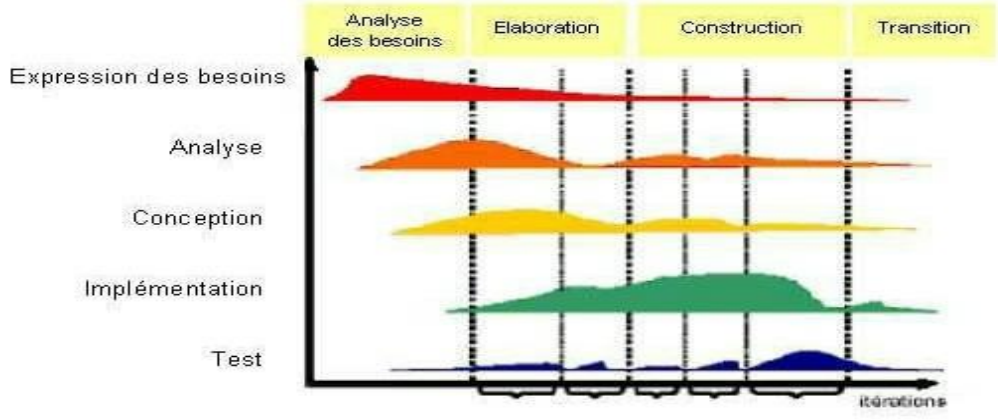

FIGURE 1.6 – Enchaînements d'activités au cours du cycle de vie.

1.6.2 Unified Modeling Language (UML)

Définition :

Le Unified Modeling Language (UML) est un langage de modélisation graphique utilisé pour représenter, visualiser, spécifier et documenter les systèmes logiciels de manière structurée. UML est largement adopté pour la modélisation orientée objet et la conception de systèmes complexes.

Caractéristiques :

- **Diagrammes variés** : UML inclut différents types de diagrammes, tels que les diagrammes de cas d'utilisation, de séquence, de classe, d'activité, etc., chacun servant à représenter des aspects spécifiques du système.
- **Modélisation orientée objet** : UML facilite la modélisation des objets et leurs interactions au sein d'un système, aidant ainsi à clarifier et à organiser les structures complexes.
- **Standard international** : UML est reconnu comme un standard international, assurant une compréhension et une communication uniformes entre les différents acteurs du développement logiciel.
- **Documentation visuelle** : UML offre des représentations graphiques qui aident à documenter et à communiquer les conceptions et les architectures logicielles de manière claire et concise.

Dans notre projet, nous allons utiliser les diagrammes suivants, définis comme tel :

Diagramme de cas d'utilisation : Ce diagramme représente les différentes fonctionnalités que le système doit offrir aux utilisateurs. Il est utilisé pour capturer les besoins fonctionnels et techniques nécessaires à la réalisation du projet.

Diagramme de séquence : Ce type de diagramme d'interaction illustre les échanges de messages entre les objets, dans le cadre d'un scénario particulier du système. Il aide à développer et analyser les scénarios d'utilisation, en détaillant comment les objets interagissent pour accomplir une tâche spécifique.

Diagramme de classe : Ce diagramme, crucial dans toutes les méthodes orientées objet, montre les classes du système, leurs attributs, leurs méthodes et les relations entre elles. Il offre la plus grande variété de notations pour représenter les structures statiques du système.

1.7 Conclusion

Dans ce chapitre, nous avons présenté le contexte du projet, en soulignant la problématique rencontrée et en proposant une solution innovante avec "Bignova Immo". Nous avons également mis en avant l'intérêt de développer une application web et mobile pour répondre aux besoins des utilisateurs de manière efficace et accessible. Pour conclure, nous avons décrit le processus de développement adapté à notre projet. Dans le prochain chapitre, nous aborderons la spécification et l'analyse des besoins.

2

Spécification et Analyse des besoins

2.1 Introduction

Ce chapitre introduit la phase d'analyse des besoins. Nous commencerons par identifier les différents acteurs du système ainsi que les besoins fonctionnels qu'ils expriment.
Ensuite, nous décrirons la spécification des besoins non fonctionnels et leurs contraintes. Enfin, les cas d'utilisation seront accompagnés d'une description détaillée et des diagrammes de séquence.

2.2 Spécification et analyse des besoins

Cette section vise à clarifier les objectifs du projet en identifiant les fonctionnalités essentielles à mettre en œuvre. Elle est divisée en deux parties : d'une part, les besoins fonctionnels, qui décrivent les fonctionnalités spécifiques requises par le système ; d'autre part, les besoins non fonctionnels, qui définissent les contraintes, ainsi que les exigences de performance et de qualité que le système doit respecter.

2.2.1 Besoins fonctionnels

Les besoins fonctionnels décrivent les fonctionnalités spécifiques que le système doit offrir pour répondre aux attentes des utilisateurs.
Elles se résument en les fonctionnalités suivantes :

Gestion des Produits : Le système doit permettre aux administrateurs de consulter la liste des produits disponibles. Ils auront la possibilité d'ajouter de nouveaux produits à la liste ainsi que de supprimer des produits existants. Ils pourront également rechercher et trier les produits, ainsi que voir les détails tels que l'administrateur qui a ajouté le produit et la date d'achat.

Gestion des Projets Immobiliers : Les administrateurs devront pouvoir consulter la liste des projets immobiliers en cours, terminés, ou planifiés pour l'avenir. Les administrateurs auront la capacité d'ajouter de nouveaux projets, de supprimer des projets existants et de gérer leur état, que ce soit pour les projets en cours, terminés ou futurs.

Gestion des Employés et Pointages : Le système doit permettre aux administrateurs de consulter la liste des employés. Ils auront la possibilité d'ajouter de nouveaux employés, de supprimer des employés existants et de gérer le pointage des employés.

Gestion des Lots et des Acquéreurs : Les administrateurs devront pouvoir consulter les lots réservés et non réservés. Les administrateurs auront la possibilité de gérer les réservations de lots, en permettant d'ajouter, de modifier ou de supprimer des réservations.

Gestion des Paiements : Le système doit permettre aux administrateurs de consulter l'historique des paiements effectués par les clients. Les administrateurs pourront ajouter de nouveaux enregistrements de paiements, gérer les bons de paiement et suivre les transactions.

Gestion des Administrateurs : Le super administrateur(propriétaire) devra pouvoir consulter la liste des administrateurs, ajouter de nouveaux administrateurs, supprimer des administrateurs existants, et définir ou modifier les droits d'accès des administrateurs.

Dashboard : Un tableau de bord personnalisable doit être fourni pour permettre aux utilisateurs de visualiser les informations clés sur les projets, les paiements, les réservations, et autres aspects importants de l'application.

Langue et Authentification : Le système doit offrir un support multilingue, permettant aux utilisateurs de choisir la langue de l'application. De plus, une authentification sécurisée doit être assurée pour tous les utilisateurs de l'application afin de protéger l'accès aux fonctionnalités et aux données sensibles.

Application Mobile : Une application mobile doit permettre aux clients de suivre l'avancement de leurs projets et de gérer leurs paiements directement depuis leur appareil mobile.

2.2.2 Besoins non fonctionnels

Les besoins non fonctionnels, bien qu'ils n'affectent pas directement les fonctionnalités du système, ont un impact significatif sur l'expérience utilisateur. Ils définissent les critères de qualité, les contraintes techniques et les performances attendues du système. Ces besoins incluent les exigences en matière de performance, de matériel, et de contraintes d'implémentation.

Performance :L'application doit offrir des temps de réponse rapides et une haute disponibilité pour garantir une expérience utilisateur fluide. Pour assurer la performance, il est essentiel de :

Optimiser les requêtes de la base de données : Écrire des requêtes efficaces en évitant les sous-requêtes lourdes en limitant le nombre de colonnes sélectionnées peut réduire le temps d'exécution des requêtes.

Pagination : consiste à diviser les résultats d'une requête en plusieurs pages ou ensembles de données plus petits. Cela permet de récupérer et d'afficher une portion limitée de données à la fois, plutôt que de charger tout le jeu de données en une seule fois.

Mise en cache : est une technique de gestion de la performance qui consiste à stocker temporairement des données ou des résultats intermédiaires pour réduire le temps de réponse et la charge sur les systèmes backend

Sécurité : Les données collectées par l'application doivent être accessibles uniquement par le personnel autorisé. Cela peut être assuré en utilisant des techniques telles que les JSON Web Tokens (JWT) [6] pour la gestion des sessions utilisateur et le hachage des mots de passe [7] pour garantir leur sécurité.

Facilité d'utilisation : est cruciale pour garantir une expérience utilisateur agréable et intuitive. L'application doit être conçue de manière à ce que les utilisateurs puissent naviguer facilement et accomplir leurs tâches sans confusion. Cela implique une interface claire et cohérente, des fonctionnalités bien organisées, et des éléments interactifs accessibles et réactifs. L'objectif est de rendre l'application aussi intuitive que possible, même pour les utilisateurs novices

Maintenabilité : est la capacité des composants ou des applications à être maintenus.Pour assurer une maintenabilité optimale, il est important de structurer le code de manière modulaire et de le documenter soigneusement. Cela facilite les mises à jour, la correction des bugs et l'ajout de nouvelles fonctionnalités. L'application des bonnes pratiques de développement, telles que les tests unitaires réguliers et les revues de code, contribue également à maintenir un code propre et facilement gérable. En fin de compte, l'objectif est de rendre le système facile à comprendre, à modifier et à étendre pour les développeurs.

La disponibilité : désigne la capacité de l'application à être accessible et opérationnelle en permanence. Elle est cruciale pour garantir que les utilisateurs peuvent accéder au service à tout moment, sans interruption.

2.2.3 Identification des acteurs

Pour bien comprendre les interactions au sein de notre système, nous devons identifier les différents acteurs qui interagiront avec l'application. Les principaux acteurs de notre application sont :

Super Administrateur : Responsable de la gestion globale de l'application, y compris la gestion des administrateurs.

Administrateur : Responsable de la gestion des employés, des projets, des produits, des lots, des acquéreurs, des réservations et des paiements, selon les droits d'accès définis.

Acquéreur : Utilisateur de l'application mobile pour suivre l'avancement de son lot et gérer ses paiements.

2.2.4 Identification des cas d'utilisation

Un cas d'utilisation (use case) représente un scénario précis dans lequel un acteur interagit avec le système pour accomplir un objectif spécifique. Il décrit la séquence d'actions ou d'événements qui permettent à l'acteur d'atteindre cet objectif, en montrant comment le système réagit aux actions de l'acteur. Les cas d'utilisation sont utilisés pour capturer les exigences fonctionnelles d'un système et pour s'assurer que toutes les interactions nécessaires entre les utilisateurs et le système sont correctement définies et prises en compte dans le développement du logiciel.

Nº	Cas d'utilisation	Acteur
1	**S'authentifier**	Acquéreur, Administrateur
2	**Gérer les produits** (ajouter, consulter, modifier, supprimer, filtrer, ajouter une quantité, prendre une quantité)	Administrateur
3	**Gérer les employés** (ajouter, consulter, modifier, supprimer, filtrer)	Administrateur
4	**Gérer les pointages** (créer, filtrer par date, filtrer par employé)	Administrateur
5	**Gérer les administrateurs** (consulter, ajouter, supprimer, activer, désactiver)	Super Administrateur
6	**Gérer les projets et les lots** (ajouter, consulter, modifier, supprimer, filtrer, changer l'état)	Administrateur
7	**Gérer les types de lots** (ajouter, consulter, modifier, supprimer, filtrer)	Administrateur
8	**Gérer les acquéreurs** (ajouter, consulter, modifier, supprimer, filtrer, changer l'état)	Administrateur
9	**Gérer les réservations** (ajouter, consulter, modifier, supprimer, filtrer)	Administrateur
10	**Gérer les paiements et les bons**	Administrateur, Acquéreur
11	**Voir l'avancement de projet**	Administrateur, Acquéreur

TABLE 2.1 – Cas d'utilisations associés

Remarque : Il est à noter que le Super Administrateur dispose de tous les droits et privilèges d'un Administrateur, en plus de capacités supplémentaires.

2.2.5 Diagramme de cas d'utilisation global

Dans cette section, le diagramme de cas d'utilisation a été répartis en trois diagrammes distincts pour une meilleure clarté et organisation. Le premier diagramme se concentre sur les interactions de l'acquéreur, le second sur la gestion des ressources, et le troisième sur la gestion des ventes. Ces diagrammes illustrent les scénarios spécifiques pour chaque domaine, offrant une vue d'ensemble structurée des fonctionnalités du système.

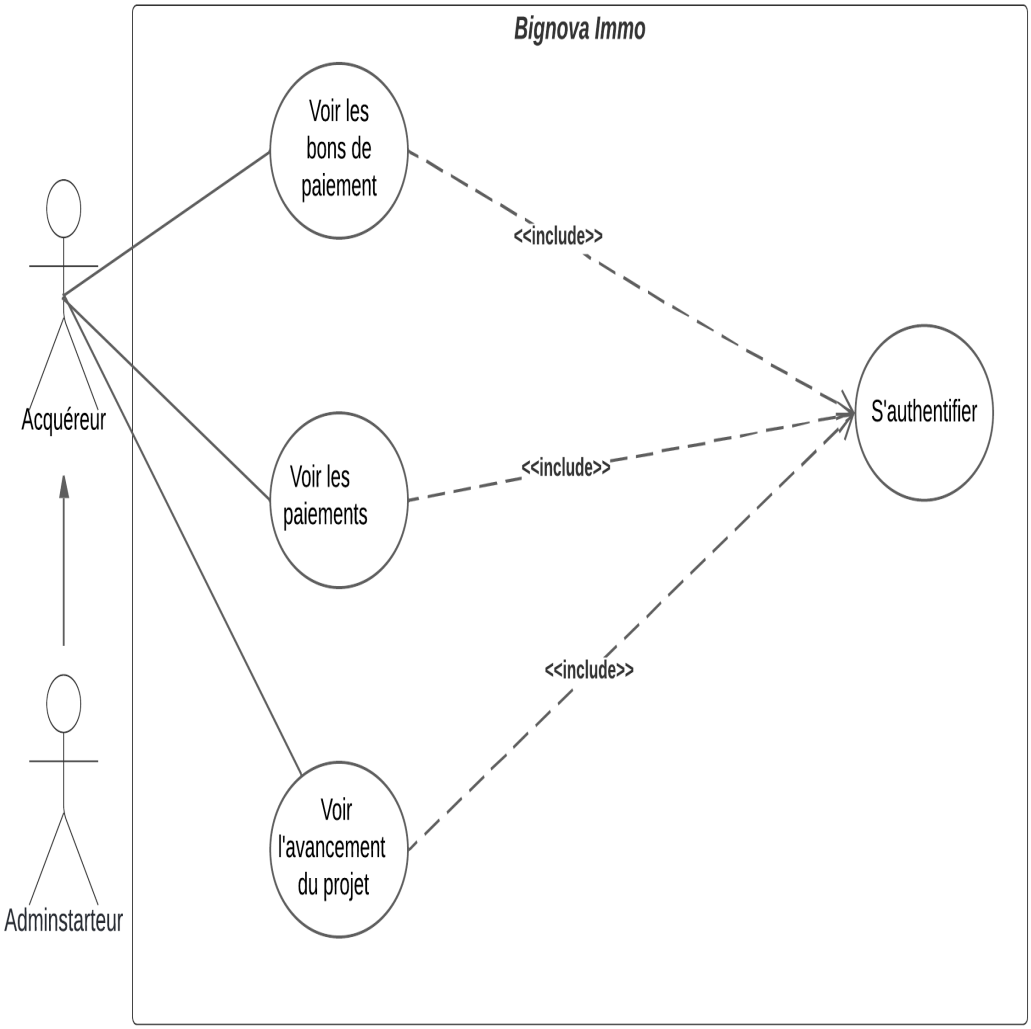

FIGURE 2.1 – Diagramme de cas d'utilisation "Acquéreur".

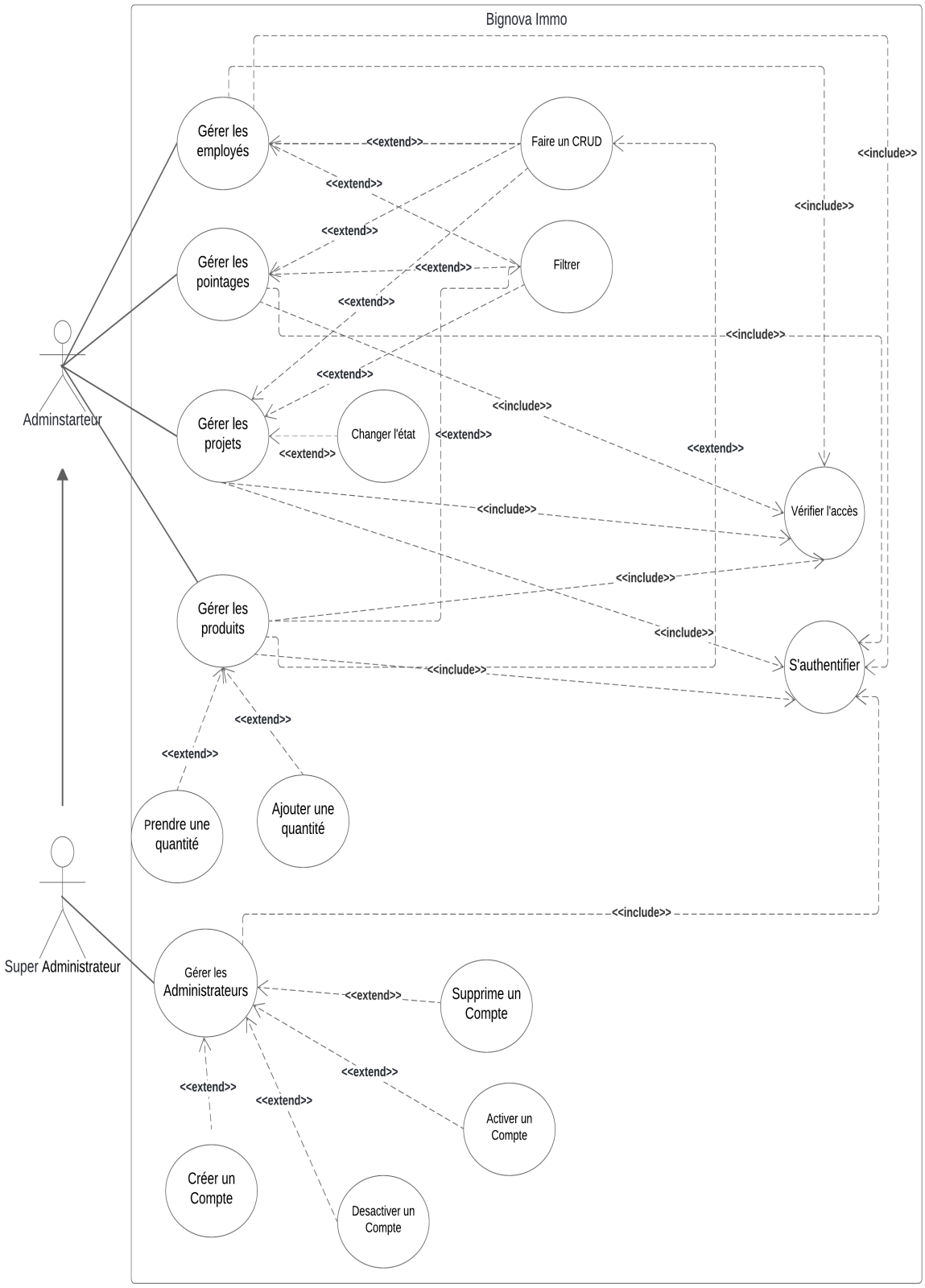

FIGURE 2.2 – Diagramme de cas d'utilisation "Gestion des ressources".

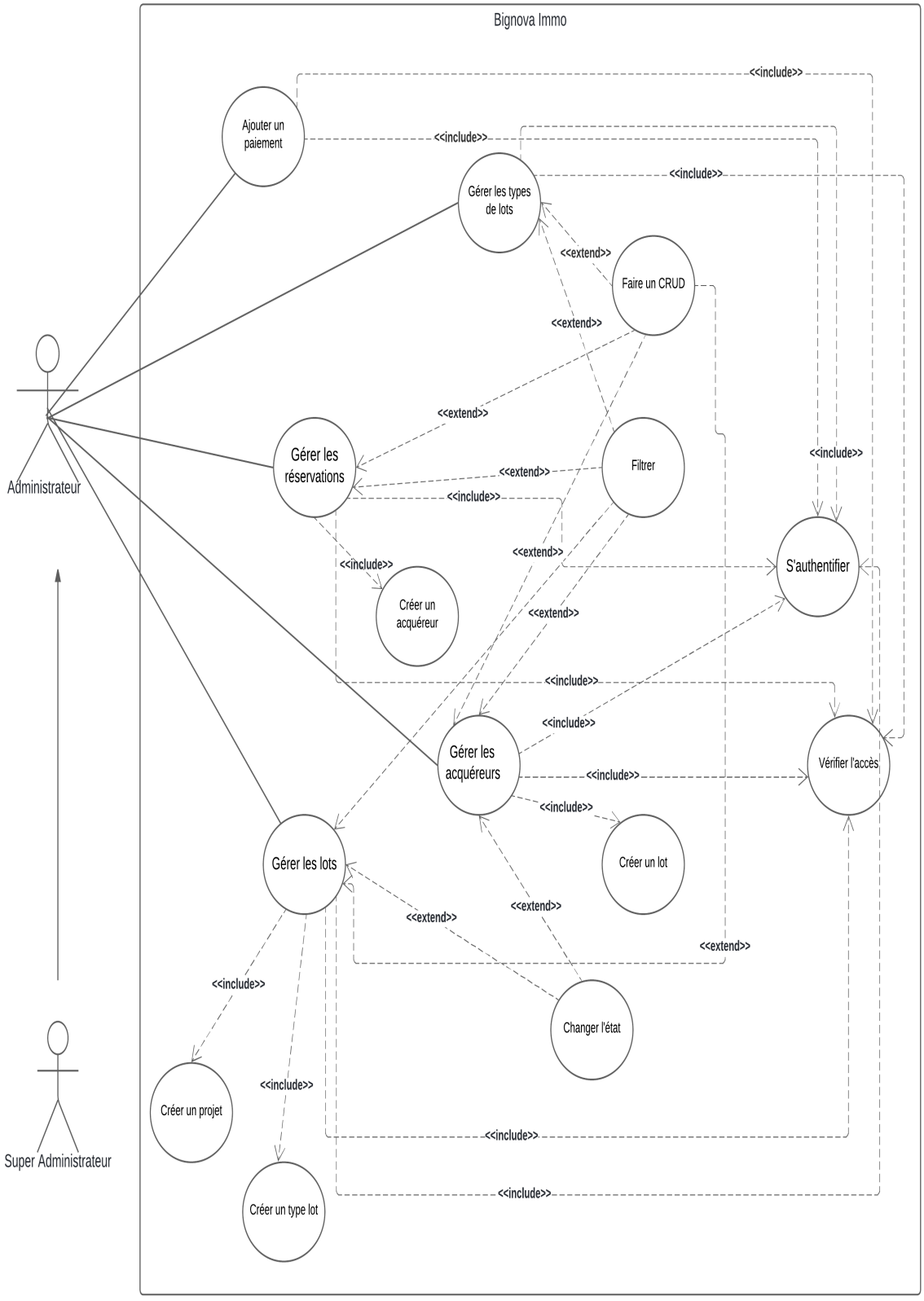

FIGURE 2.3 – Diagramme de cas d'utilisation "Gestion des ventes".

2.2.6 Description des cas d'utilisation

Nous allons présenter une description textuelle des cas d'utilisation les plus importants.

Description du cas d'utilisation Ajouter un produit :

Sommaire	
Titre	Ajouter un produit.
Résumé	Ajout d'un nouveau produit au catalogue.
Acteur	Administrateur.
Description des scénarios	
Pré-condition	L'administrateur doit être authentifié.
Scénario nominal	1. L'administrateur accède à l'interface de gestion des produits. 2. L'administrateur sélectionne l'option "Ajouter un produit". 3. Le système affiche un formulaire pour l'ajout du produit. 4. L'administrateur remplit les champs requis dans le formulaire. 5. L'administrateur soumet le formulaire. 6. Le système vérifie les informations et les enregistre. 7. Le système ajoute le produit au catalogue et affiche un message de confirmation. 8. Le système renvoie l'administrateur à l'interface de gestion des produits.
Scénario alternatif	**A1 : Champs manquants ou informations incorrectes.** Le système affiche un message d'erreur avec les champs à corriger. Le scénario nominal reprend au point 4. **A2 : Produit existe déjà.** Le scénario démarre après le point 3 du scénario nominal : 4. Le système affiche que le produit existe déjà et indique qu'une quantité peut être ajoutée. 5. L'administrateur accède à l'interface pour ajouter une quantité et remplit les champs nécessaires. 6. Le système vérifie les informations et ajoute la quantité au produit. 7. Le système affiche que la quantité a été ajoutée avec succès.
Scénario d'exception	**E1 : Erreur système lors de l'ajout du produit.** Le scénario démarre après le point 6 du scénario nominal : 7. Le système rencontre une erreur technique. 8. Le système affiche un message d'erreur indiquant que l'ajout du produit a échoué. 9. L'administrateur peut essayer à nouveau plus tard ou contacter le support technique.
Post-conditions	Le produit est ajouté au catalogue, sauf en cas d'erreur ou si le produit existe déjà. Si le produit existe déjà, une quantité est ajoutée au lieu de créer un nouveau produit.

TABLE 2.2 – Description du cas d'utilisation Ajouter un produit

Description du cas d'utilisation Ajouter un Administrateur :

Sommaire	
Titre	Ajouter un Administrateur.
Résumé	Ajout d'un nouvel administrateur avec définition des rôles d'accès.
Acteur	Super Administrateur.
Description des scénarios	
Pré-condition	Le super administrateur doit être authentifié .
Scénario nominal	1. Le super administrateur accède à l'interface de gestion des administrateurs. 2. Le super administrateur sélectionne l'option "Ajouter un administrateur". 3. Le système affiche un formulaire pour l'ajout de l'administrateur. 4. Le super administrateur remplit les champs obligatoires dans le formulaire, y compris la définition des rôles d'accès. 5. Le super administrateur soumet le formulaire. 6. Le système vérifie les informations, enregistre le nouvel administrateur et ses rôles. 7. Le système affiche un message de confirmation et renvoie le super administrateur à l'interface de gestion des administrateurs.
Scénario alternatif	**A1 : Champs obligatoires manquants.** Le système affiche un message d'erreur demandant de remplir tous les champs obligatoires. Le scénario nominal reprend au point 4.
Scénario d'exception	**E1 : Erreur dans la définition des rôles.** Le scénario démarre après le point 6 du scénario nominal : 7. Le super administrateur remarque une erreur dans la définition des rôles. 8. Il accède à l'interface de modification des administrateurs. 9. Le super administrateur modifie les rôles et soumet les changements. 10. Le système enregistre les nouvelles informations et affiche un message de confirmation de modification.
Post-conditions	Le nouvel administrateur est ajouté avec les rôles définis, et les erreurs éventuelles dans la définition des rôles peuvent être corrigées via l'interface de modification.

TABLE 2.3 – Description du cas d'utilisation Ajouter un Administrateur

Description du cas d'utilisation Ajouter ou modifier un acquéreur :

Sommaire	
Titre	Ajouter ou modifier un acquéreur.
Résumé	Ajout d'un nouvel acquéreur ou modification d'un acquéreur existant dans le système, avec possibilité de définir le type (prospect ou client).
Acteur	Administrateur.
Description des scénarios	
Pré-condition	L'utilisateur doit être authentifié et avoir accès à l'interface de gestion des acquéreurs.
Scénario nominal	**Option 1 : Ajouter un acquéreur :** 1. L'utilisateur accède à l'interface de gestion des acquéreurs. 2. L'utilisateur sélectionne l'option "Ajouter un acquéreur". 3. Le système affiche un formulaire pour l'ajout de l'acquéreur. 4. L'utilisateur remplit les champs requis dans le formulaire, incluant le type d'acquéreur (prospect ou client). 5. L'utilisateur soumet le formulaire. 6. Le système vérifie les informations et les enregistre. 7. Le système ajoute l'acquéreur et affiche un message de confirmation. 8. Le système renvoie l'utilisateur à l'interface de gestion des acquéreurs. **Option 2 : Modifier un acquéreur :** 1. L'utilisateur accède à l'interface de gestion des acquéreurs. 2. L'utilisateur sélectionne un acquéreur existant pour le modifier. 3. Le système affiche les informations actuelles de l'acquéreur. 4. L'utilisateur modifie les champs nécessaires. 5. L'utilisateur soumet les modifications. 6. Le système vérifie les informations et met à jour l'acquéreur. 7. Le système affiche un message de confirmation de la mise à jour.
Scénario alternatif	**A1 : Champs manquants ou informations incorrectes.** Le système affiche un message d'erreur avec les champs à corriger. Le scénario nominal reprend au point 4. **A2 : Aucune modification effectuée lors de la modification.** Le système affiche un message invitant à modifier au moins un attribut.
Scénario d'exception	**E1 : Email non valide.** Le scénario démarre après le point 6 du scénario nominal : 7. Le système affiche un message d'erreur indiquant que l'email ou le numéro de téléphone est invalide. 8. L'utilisateur peut corriger les informations et soumettre à nouveau.
Post-conditions	L'acquéreur est ajouté ou modifié dans le système, sauf en cas d'erreur ou de champs manquants.

TABLE 2.4 – Description du cas d'utilisation Ajouter ou modifier un acquéreur

Description du cas d'utilisation Supprimer un acquéreur :

Sommaire	
Titre	Supprimer un acquéreur.
Résumé	Suppression d'un acquéreur .
Acteur	Administrateur, Acquéreur.
Description des scénarios	
Pré-condition	L'utilisateur doit être authentifié et avoir accès à l'interface de gestion des acquéreurs.
Scénario nominal	**1. Supprimer un acquéreur** 1. L'utilisateur accède à l'interface de gestion des acquéreurs. 2. L'utilisateur sélectionne l'acquéreur à supprimer. 3. Le système affiche une demande de confirmation pour la suppression. 4. L'utilisateur confirme la suppression. 5. Le système supprime l'acquéreur du système. 6. Le système affiche un message de confirmation de la suppression. 7. Le système renvoie l'utilisateur à l'interface de gestion des acquéreurs.
Scénario alternatif	**A1 : Acquéreur est un client avec réservations et paiements.** 1. Le scénario démarre après le point 3 du scénario nominal. 2. Le système détecte que l'acquéreur est un client avec des réservations et des paiements. 3. Le système demande à l'utilisateur de supprimer d'abord les réservations et les paiements associés. 4. Une fois les réservations et paiements supprimés,le scénario nominal reprend au point 3. 5. Le système affiche un message de confirmation de la suppression.
Scénario d'exception	**E1 : Suppression d'un mauvais acquéreur.** 1. Le scénario démarre après le point 6 du scénario nominal. 2. L'utilisateur découvre que l'acquéreur supprimé était incorrect. 3. L'utilisateur peut recréer l'acquéreur en suivant le processus d'ajout d'acquéreur. 4. Le système renvoie l'utilisateur à l'interface d'ajout d'acquéreur.
Post-conditions	L'acquéreur est supprimé du système. Si l'acquéreur supprimé était incorrect, il peut être recréé.

TABLE 2.5 – Description du cas d'utilisation Supprimer un acquéreur

Description du cas d'utilisation Ajouter un paiement :

Sommaire	
Titre	Ajouter un paiement
Résumé	Ajout d'un paiement pour un acquéreur.
Acteur	Administrateur.
Description des scénarios	
Pré-condition	L'administrateur doit être authentifié avec accès à la gestion des paiements.
Scénario nominal	**Option 1 : Paiement initial** 1. L'administrateur sélectionne "Ajouter un nouveau paiement". 2. L'administrateur remplit les champs requis. 3. Le système vérifie les informations, crée un nouveau paiement, calcule le total et le reste dû, et propose d'imprimer un bon. **Option 2 : Paiement ultérieur** 1. L'administrateur sélectionne un acquéreur et un lot. 2. L'administrateur remplit les champs requis. 3. Le système vérifie les informations, met à jour le paiement, calcule le total et le reste dû, et propose d'imprimer un bon.
Scénario alternatif	**A1 : Montant incorrect ou champs manquants.** **Option 1 : Paiement initial** Le système affiche un message d'erreur et demande de corriger les informations. Le scénario nominal reprend à l'étape 2. **Option 2 : Paiement ultérieur** Le système affiche un message d'erreur et demande de corriger les informations. Le scénario nominal reprend à l'étape 2.
Scénario d'exception	**E1 : Erreur technique du système.** Le système affiche une panne temporaire et invite à réessayer plus tard.
Post-conditions	Le paiement est enregistré, le total et le reste sont calculés, et un bon peut être imprimé.

TABLE 2.6 – Description du cas d'utilisation Ajouter un paiement

Description du cas d'utilisation Supprimer un projet :

Sommaire	
Titre	Supprimer un projet
Résumé	Suppression d'un projet .
Acteur	Administrateur.
Description des scénarios	
Pré-condition	L'utilisateur doit être authentifié et avoir accès à l'interface de gestion des projets.
Scénario nominal	**1. Supprimer un projet** 1. L'utilisateur accède à l'interface de gestion des projets. 2. L'utilisateur sélectionne le projet à supprimer. 3. Le système affiche une demande de confirmation pour la suppression. 4. L'utilisateur confirme la suppression. 5. Le système supprime le projet du système. 6. Le système affiche un message de confirmation de la suppression. 7. Le système renvoie l'utilisateur à l'interface de gestion des projets.
Scénario alternatif	**A1 : Projet avec réservations associées.** 1. Le scénario démarre après le point 3 du scénario nominal. 2. Le système détecte que le projet a des réservations associées. 3. Le système demande à l'utilisateur de supprimer d'abord les réservations associées. 4. Une fois les réservations supprimées, le scénario nominal reprend au point 5.
Scénario d'exception	**E1 : Erreur technique du système.** 1. Le système rencontre une erreur technique pendant la suppression du projet. 2. Le système affiche une panne temporaire et invite à réessayer plus tard.
Post-conditions	Le projet est supprimé du système. En cas de projet avec réservations, les réservations doivent être supprimées au préalable.

TABLE 2.7 – Description du cas d'utilisation Supprimer un projet

2.2.7 Diagramme de séquence système

Les diagrammes de séquence illustrent le déroulement des interactions entre les acteurs et le système pour chaque cas d'utilisation. Ils permettent de visualiser le flux d'informations et les échanges de messages qui se produisent au sein du système lors de l'exécution des différentes fonctionnalités.

2.2.7.1 Diagramme de séquence système du cas d'utilisation S'authentifier

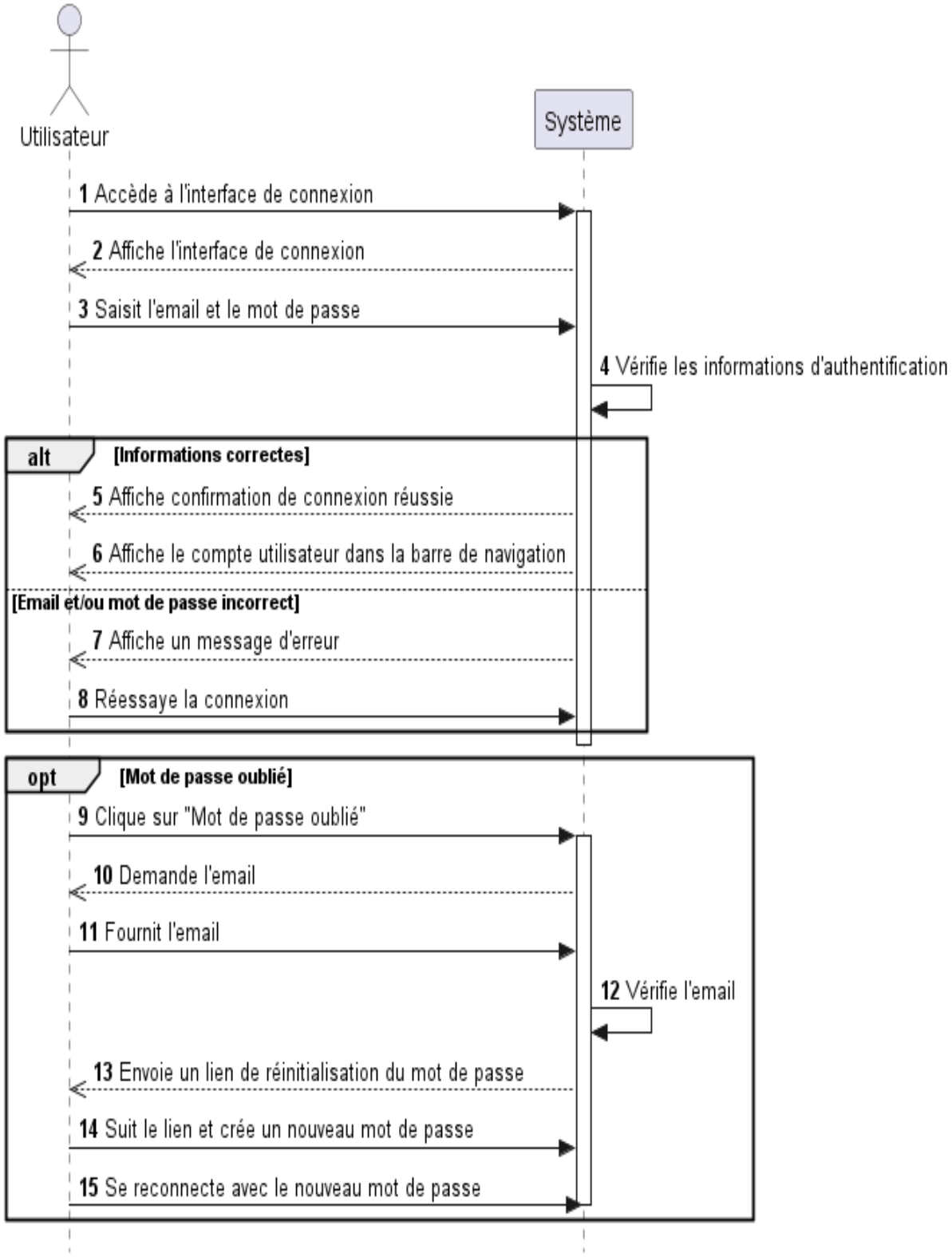

FIGURE 2.4 – Diagramme de séquence système du cas d'utilisation S'authentifier

2.2.7.2 Diagramme de séquence système du cas d'utilisation Ajouter un Administrateur

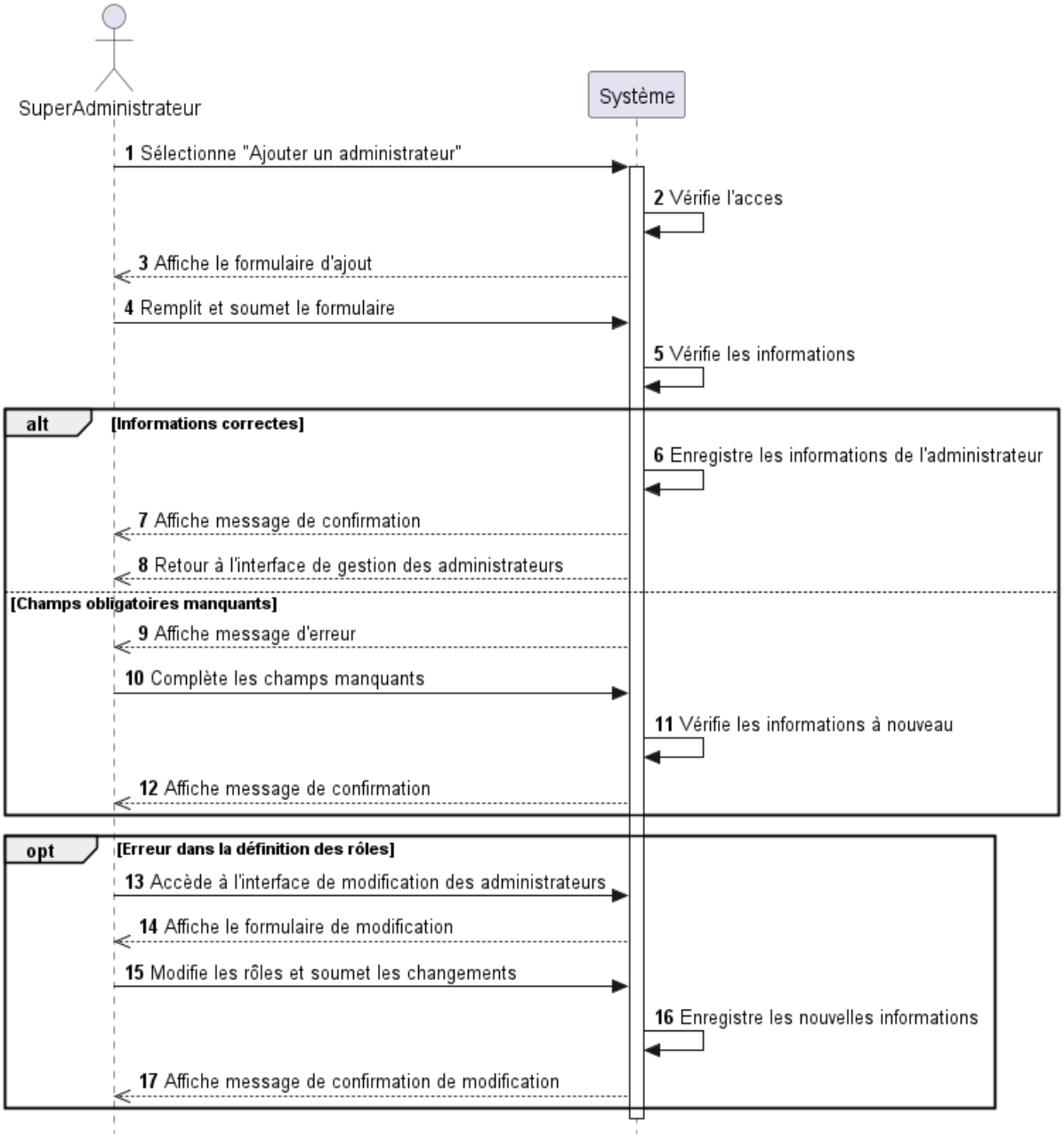

FIGURE 2.5 – Diagramme de séquence système du cas d'utilisation Ajouter un Administrateur

2.2.7.3 Diagramme de séquence système du cas d'utilisation Ajouter un Produit

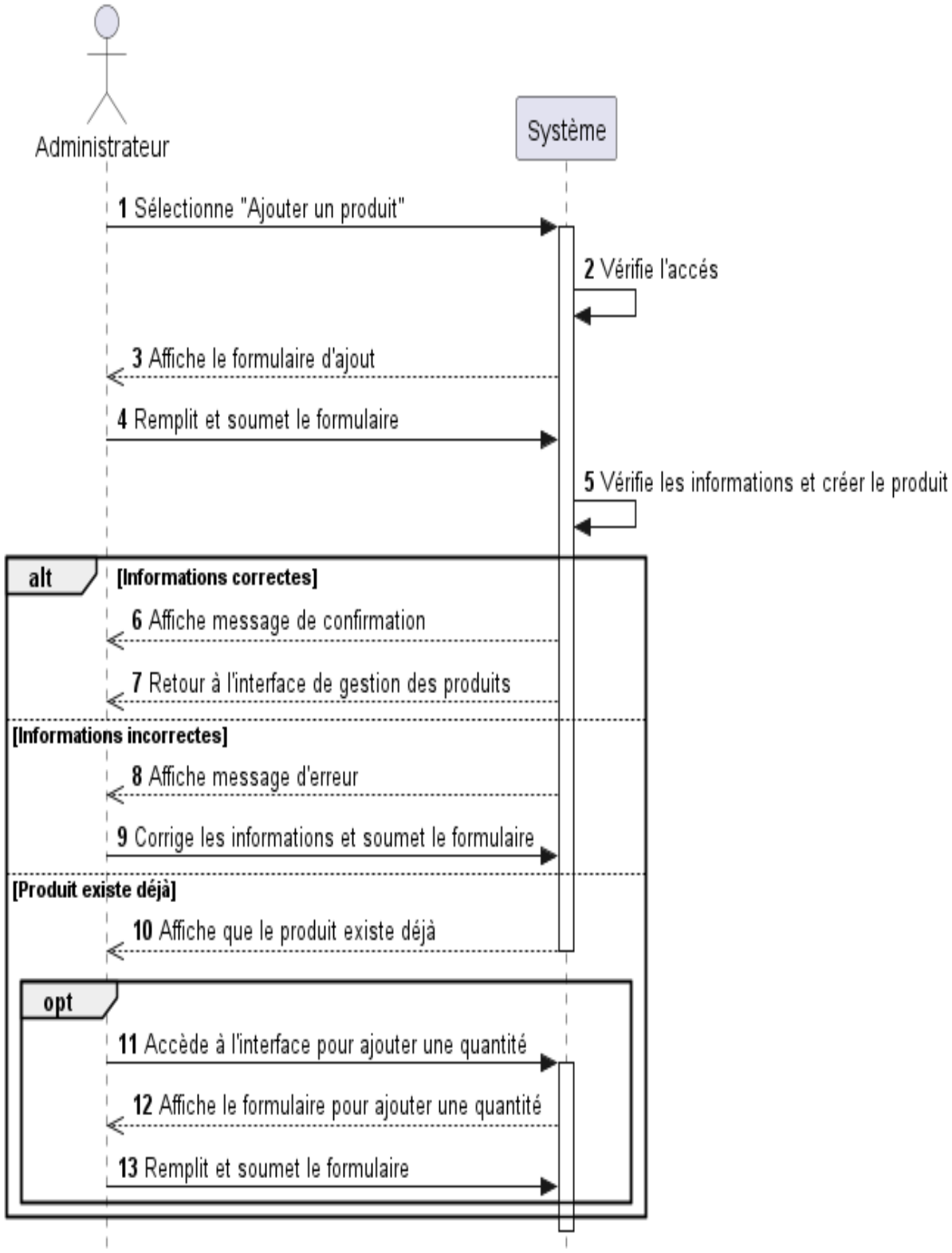

FIGURE 2.6 – Diagramme de séquence système du cas d'utilisation Ajouter un Produit

2.2.7.4 Diagramme de séquence système du cas d'utilisation Ajouter ou modifier un acquereur

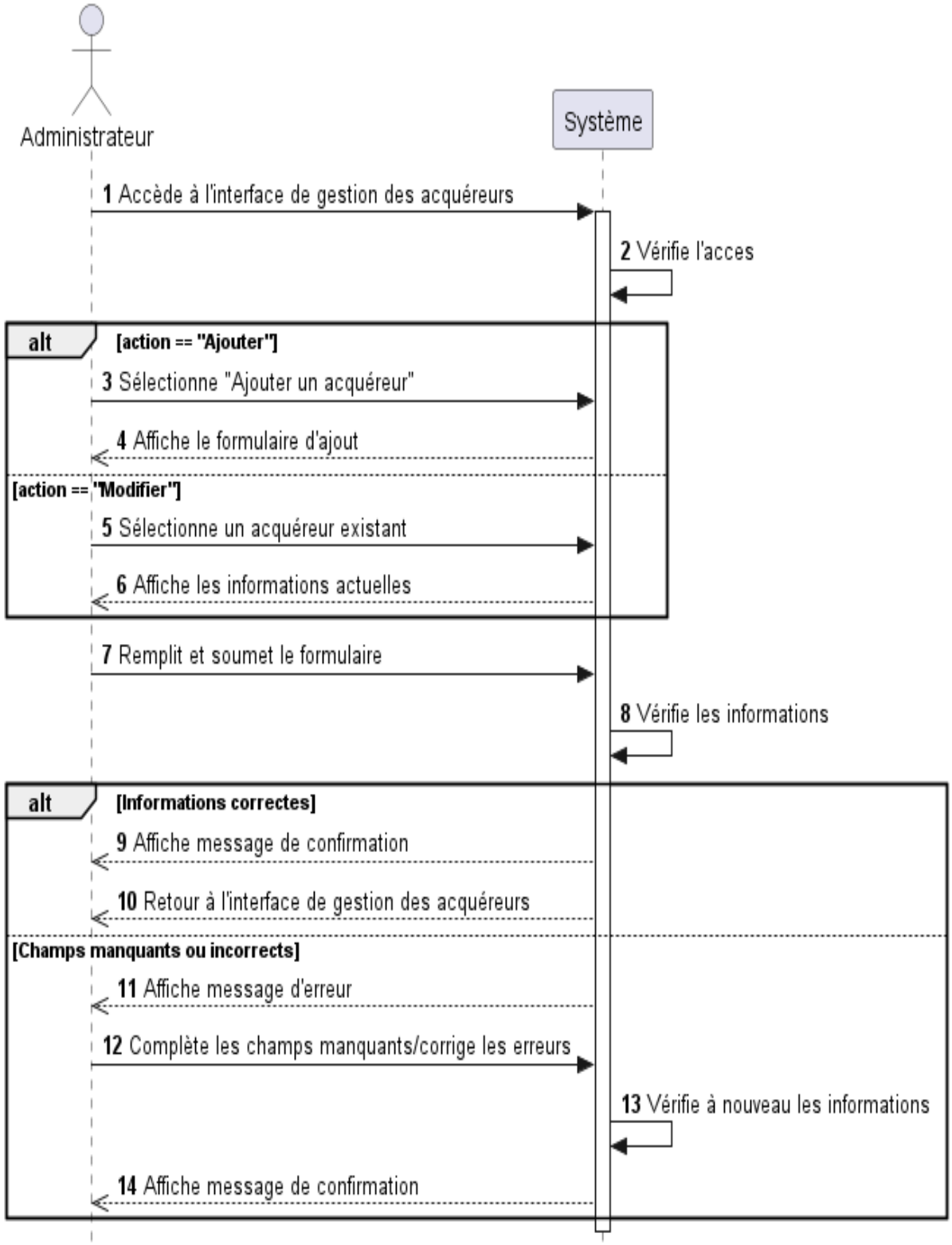

FIGURE 2.7 – Diagramme de séquence système du cas d'utilisation Ajouter ou modifier un acquereur

2.2.7.5 Diagramme de séquence système du cas d'utilisation Supprimer un acquéreur

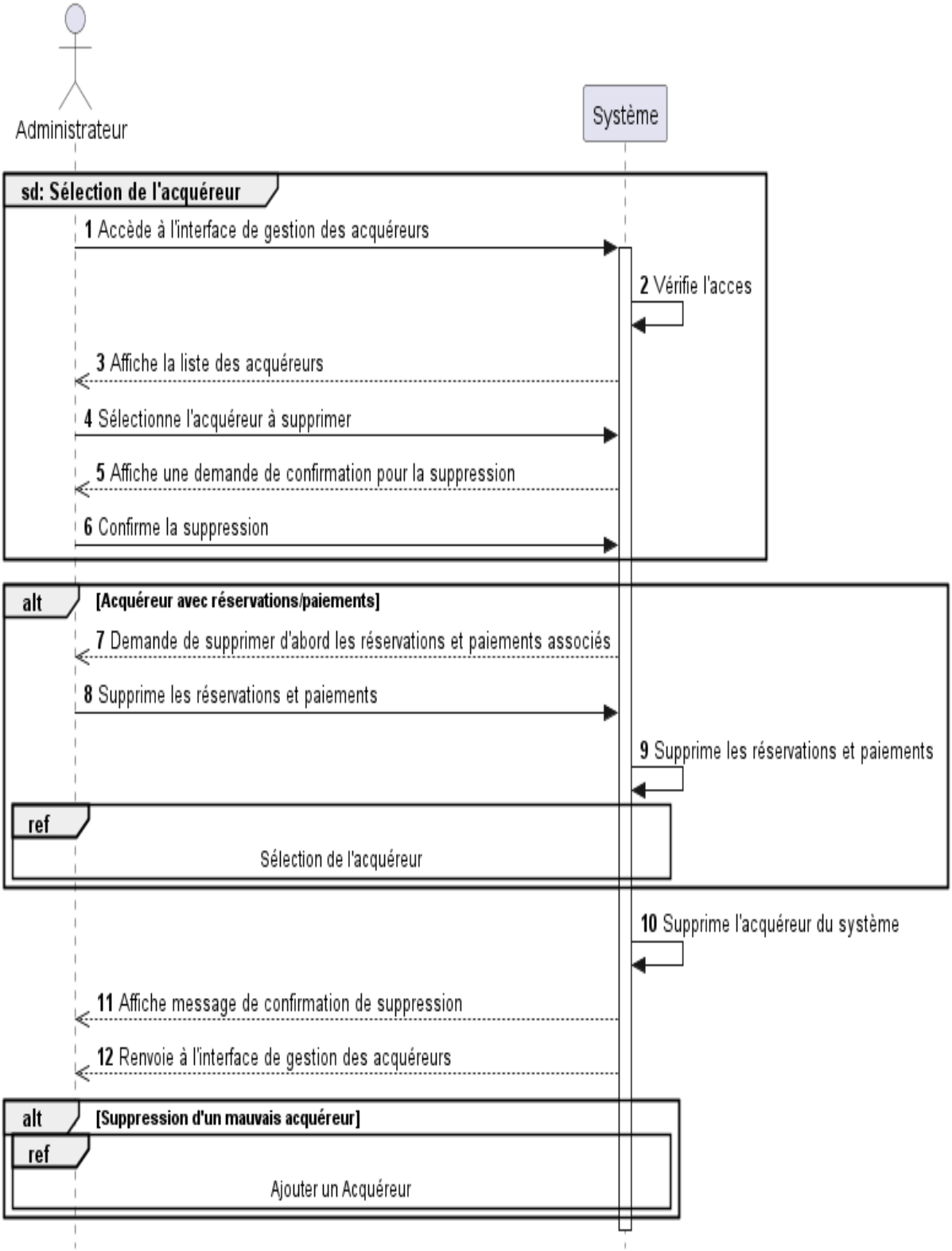

FIGURE 2.8 – Diagramme de séquence système du cas d'utilisation Supprimer un Acquéreur

2.2.7.6 Diagramme de séquence système du cas d'utilisation Ajouter un paiement

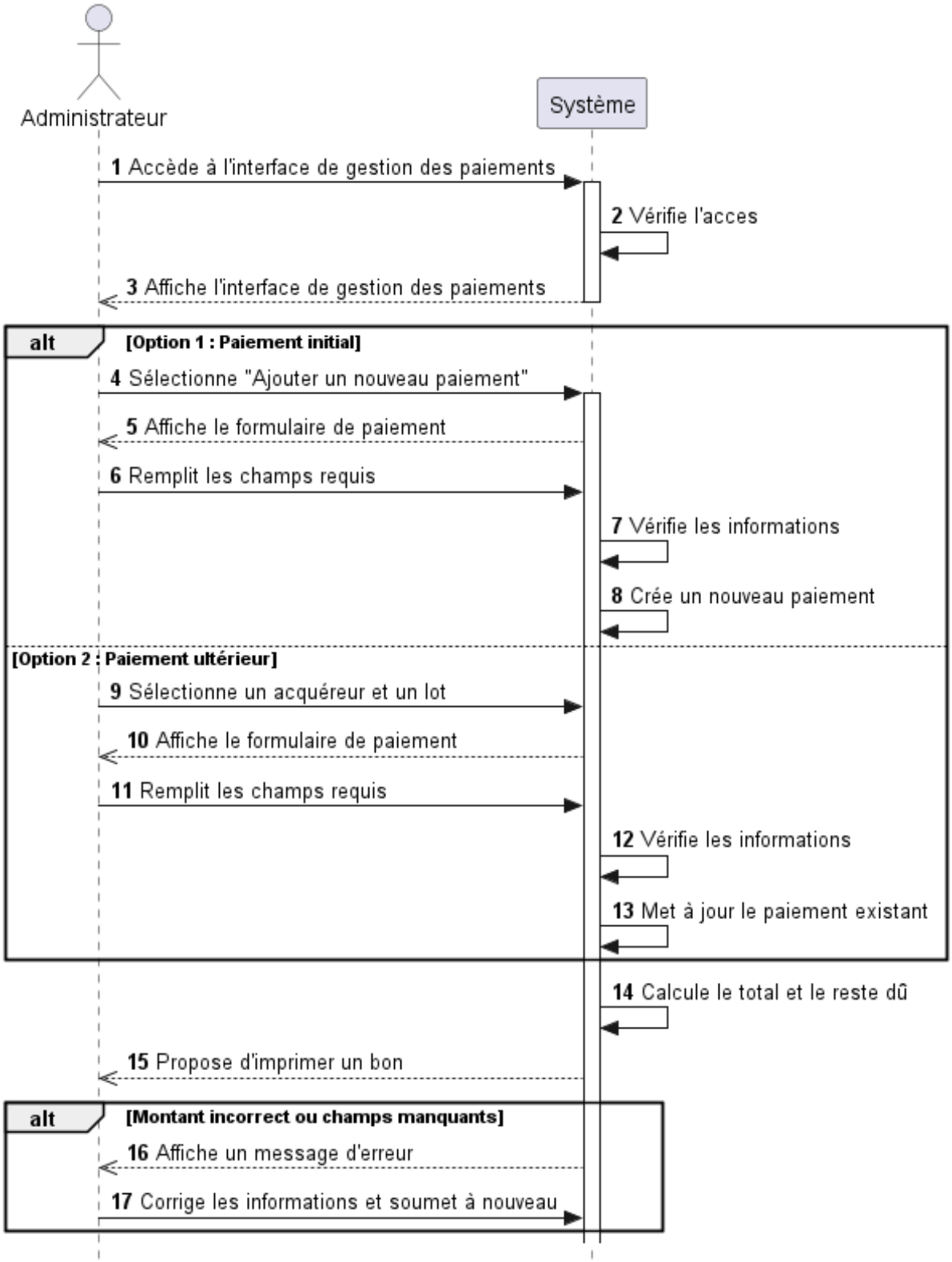

FIGURE 2.9 – Diagramme de séquence système du cas d'utilisation Ajouter un paiement

2.2.7.7 Diagramme de séquence système du cas d'utilisation Supprimer un projet

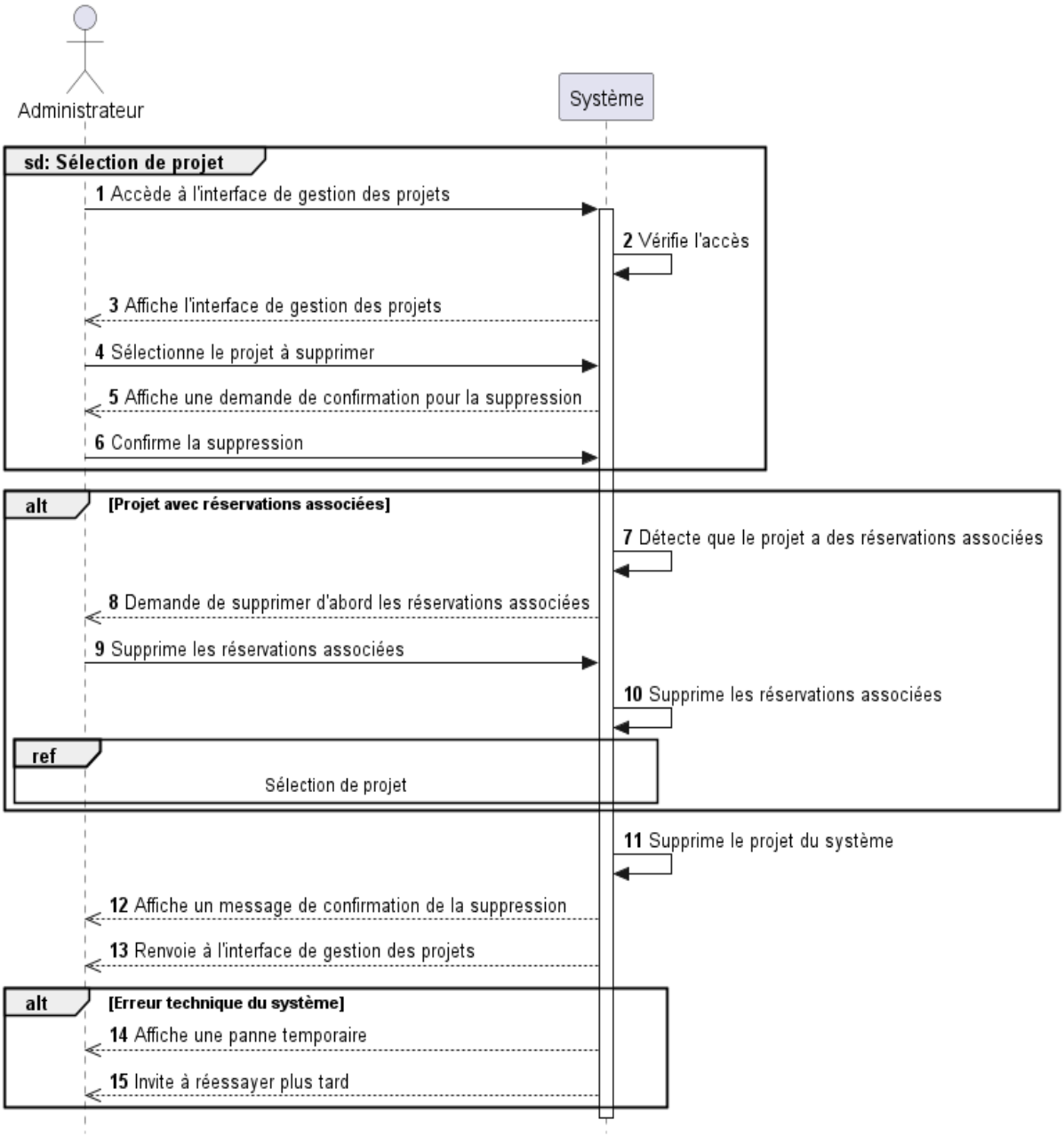

FIGURE 2.10 – Diagramme de séquence système du cas d'utilisation Supprimer un projet

2.3 Conclusion

Ce chapitre a défini les besoins fonctionnels et non fonctionnels du système, ainsi que les interactions clés entre les utilisateurs et le système. Ces éléments assurent une base solide pour la conception de l'architecture du système, qui sera abordée dans le chapitre suivant.

3
Conception

3.1 Introduction

Ce chapitre est dédié à l'exploration des diagrammes de séquence d'interaction, dérivés des diagrammes de séquence système que nous avons élaborés précédemment. Ces nouveaux diagrammes illustrent en détail les échanges entre les objets du système, offrant une vue plus fine des processus décrits dans les cas d'utilisation. L'objectif est de remplacer la vision globale du système par une analyse précise des interactions entre les objets individuels.

Ensuite, nous utiliserons ces objets identifiés pour développer un diagramme de classe complet, qui servira de fondement à la conception de notre architecture logicielle. Nous terminerons en abordant le choix de bases de donnée non relationnelle pour notre projet, en expliquant pourquoi cette technologie est la plus appropriée pour répondre aux exigences de notre système.

3.2 Diagrammes de Séquence d'Interaction

Pour chaque diagramme de séquence système, nous créerons un diagramme de séquence d'interaction correspondant, qui met en lumière les interactions spécifiques entre les composants du système. Notre approche est ancrée dans le modèle **MVC (Modèle-Vue-Contrôleur)**, ce qui nous permet de structurer les objets du système en trois catégories distinctes :

Vue : Représentant l'interface utilisateur, la vue est l'élément du système qui capte les actions de l'utilisateur et affiche les réponses du système. Elle joue un rôle crucial dans l'expérience utilisateur en assurant une communication efficace entre l'utilisateur et le système.

Contrôleur : Cet élément central du système orchestre les traitements logiques déclenchés par les actions de l'utilisateur. Il agit comme un médiateur entre la vue et le modèle, garantissant que les données sont manipulées correctement et que les processus métier sont exécutés conformément aux règles établies.

Modèle : Ce composant gère la structure des données ainsi que la logique métier associée. Le modèle est responsable de la persistance des données et de l'application des règles et des contraintes sur celles-ci. Il interagit souvent avec la base de données pour stocker et récupérer les informations nécessaires au bon fonctionnement du système.

Chaque diagramme de séquence d'interaction que nous établirons détaillera la manière dont ces objets collaborent pour accomplir les tâches requises par les différents cas d'utilisation. Nous analyserons comment la vue recueille les demandes de l'utilisateur, comment le contrôleur traite ces demandes et interagit avec le modèle pour produire un résultat, et comment la vue présente ce résultat à l'utilisateur.

3.2.1 Diagramme de séquence détaillé du cas d'utilisation "Authentification"

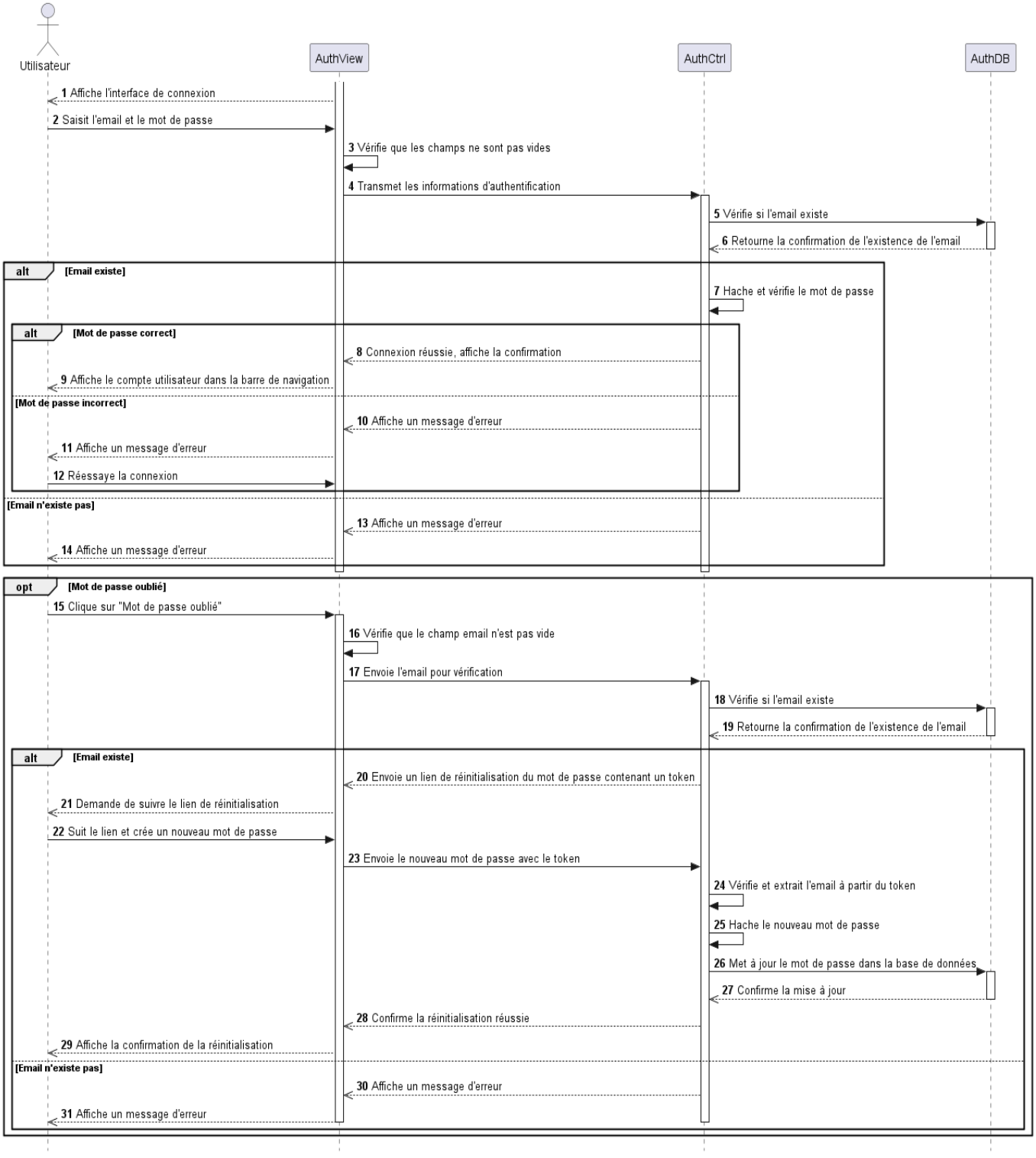

FIGURE 3.1 – Diagramme de séquence détaillé du cas d'utilisation "Authentification"

3.2.2 Diagramme de séquence détaillé du cas d'utilisation "Ajouter un Administrateur"

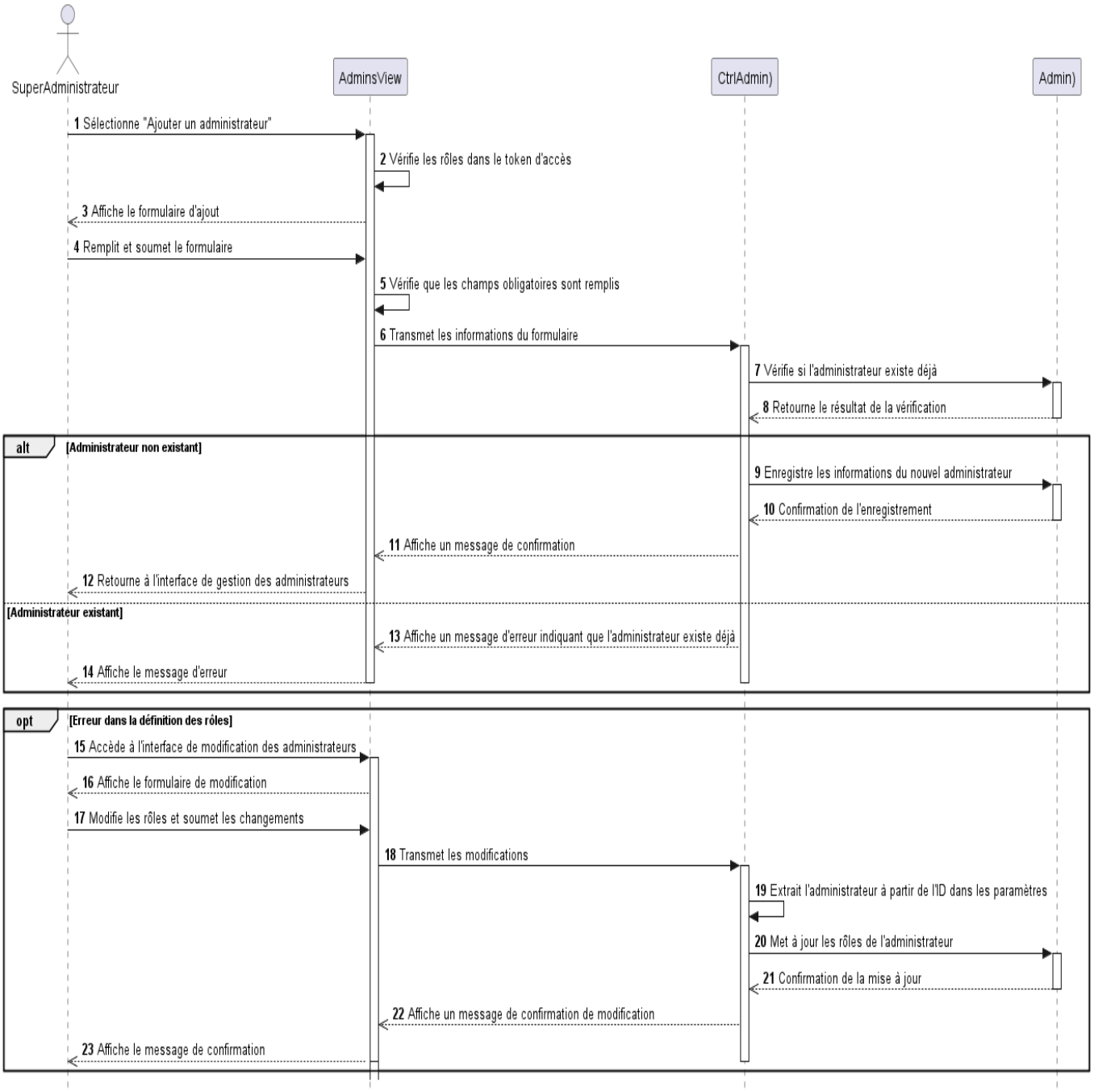

FIGURE 3.2 – Diagramme de séquence détaillé du cas d'utilisation "Ajouter un Administrateur"

3.2.3 Diagramme de séquence détaillé du cas d'utilisation "Ajouter un Produit"

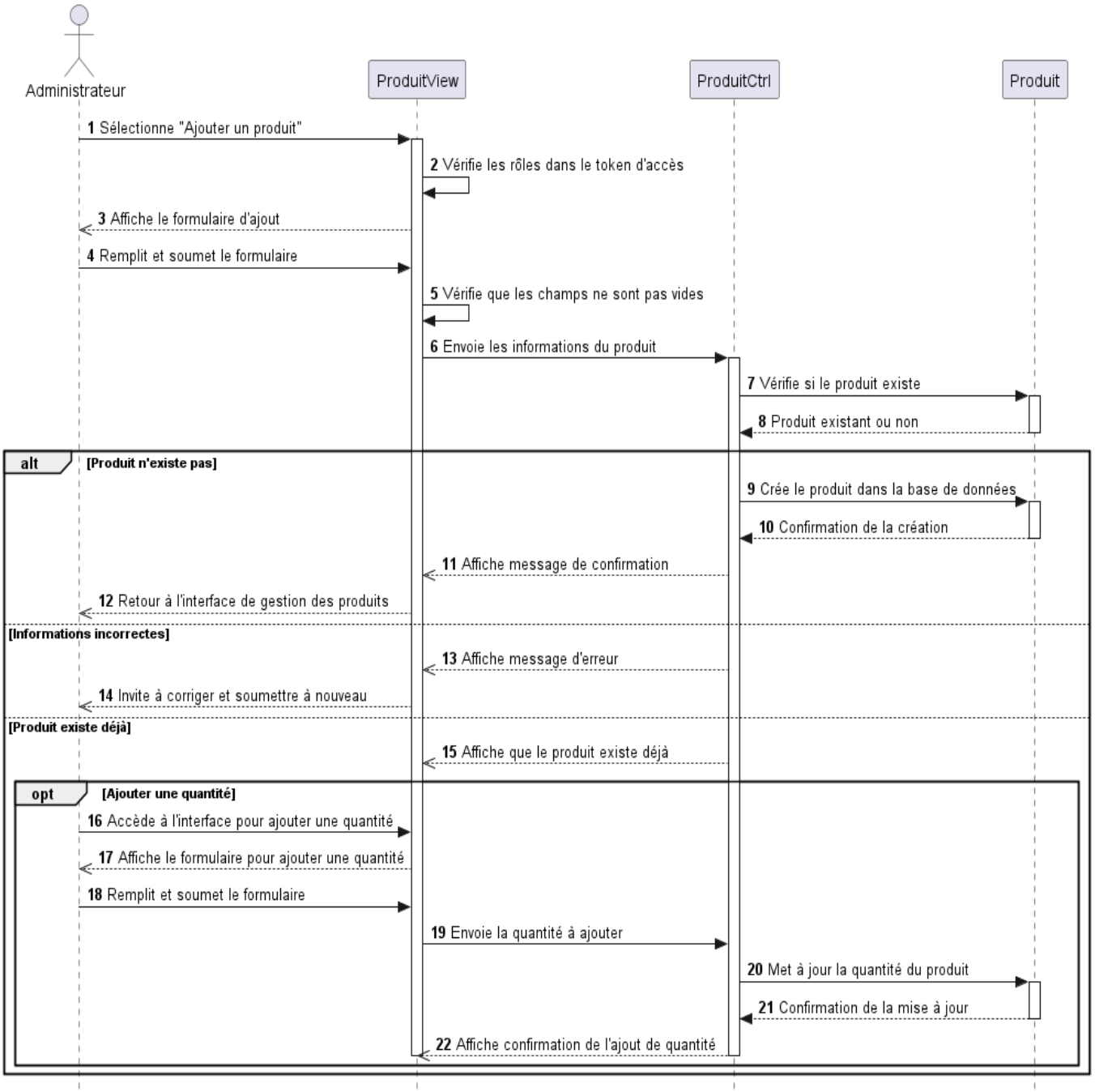

FIGURE 3.3 – Diagramme de séquence détaillé du cas d'utilisation "Ajouter un Produit"

3.2.4 Diagramme de séquence détaillé du cas d'utilisation "Supprimer un Acquéreur"

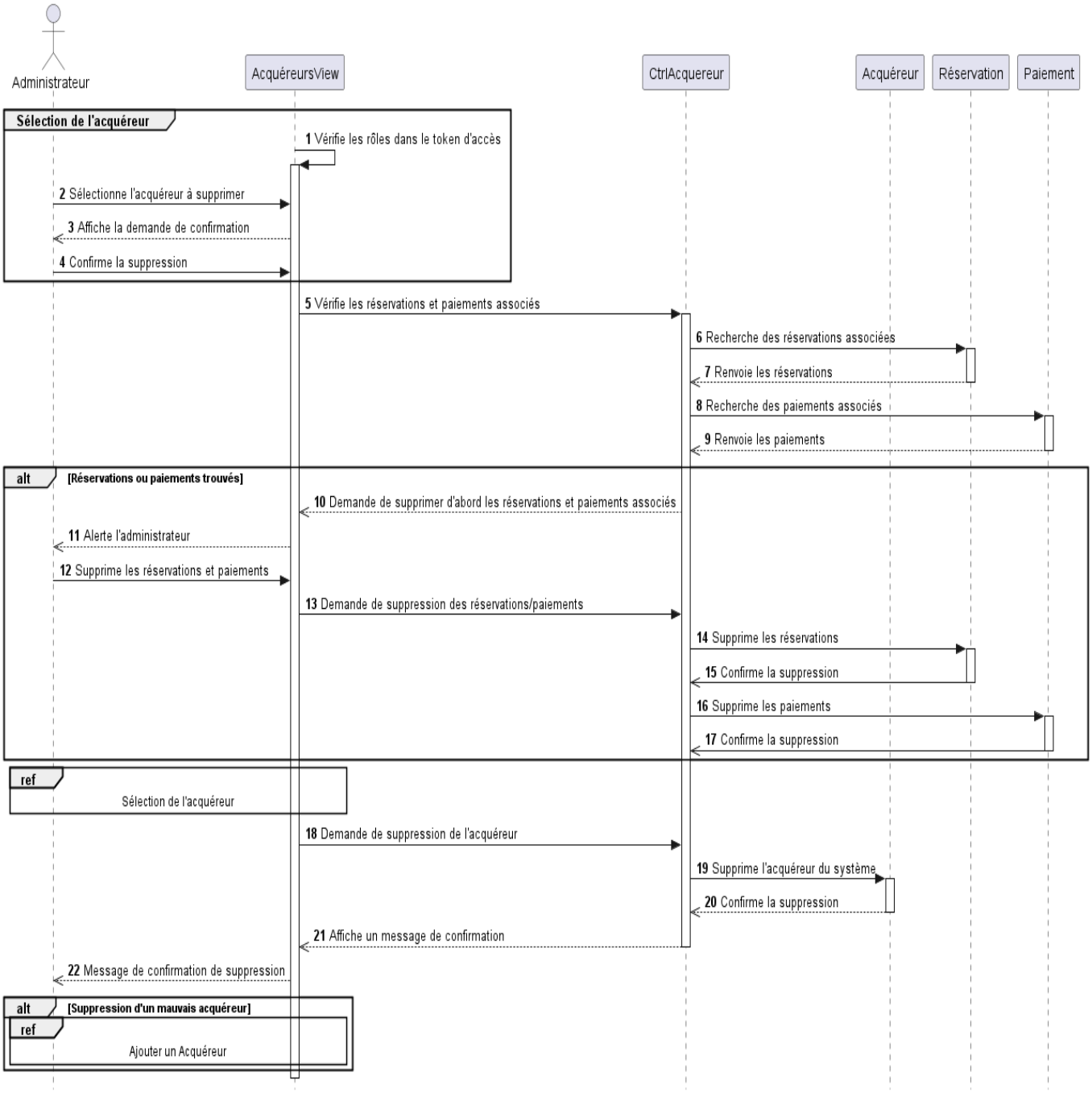

FIGURE 3.4 – Diagramme de séquence détaillé du cas d'utilisation "Supprimer un Acquéreur"

3.2.5 Diagramme de séquence détaillé du cas d'utilisation "Vérifier l'Accès"

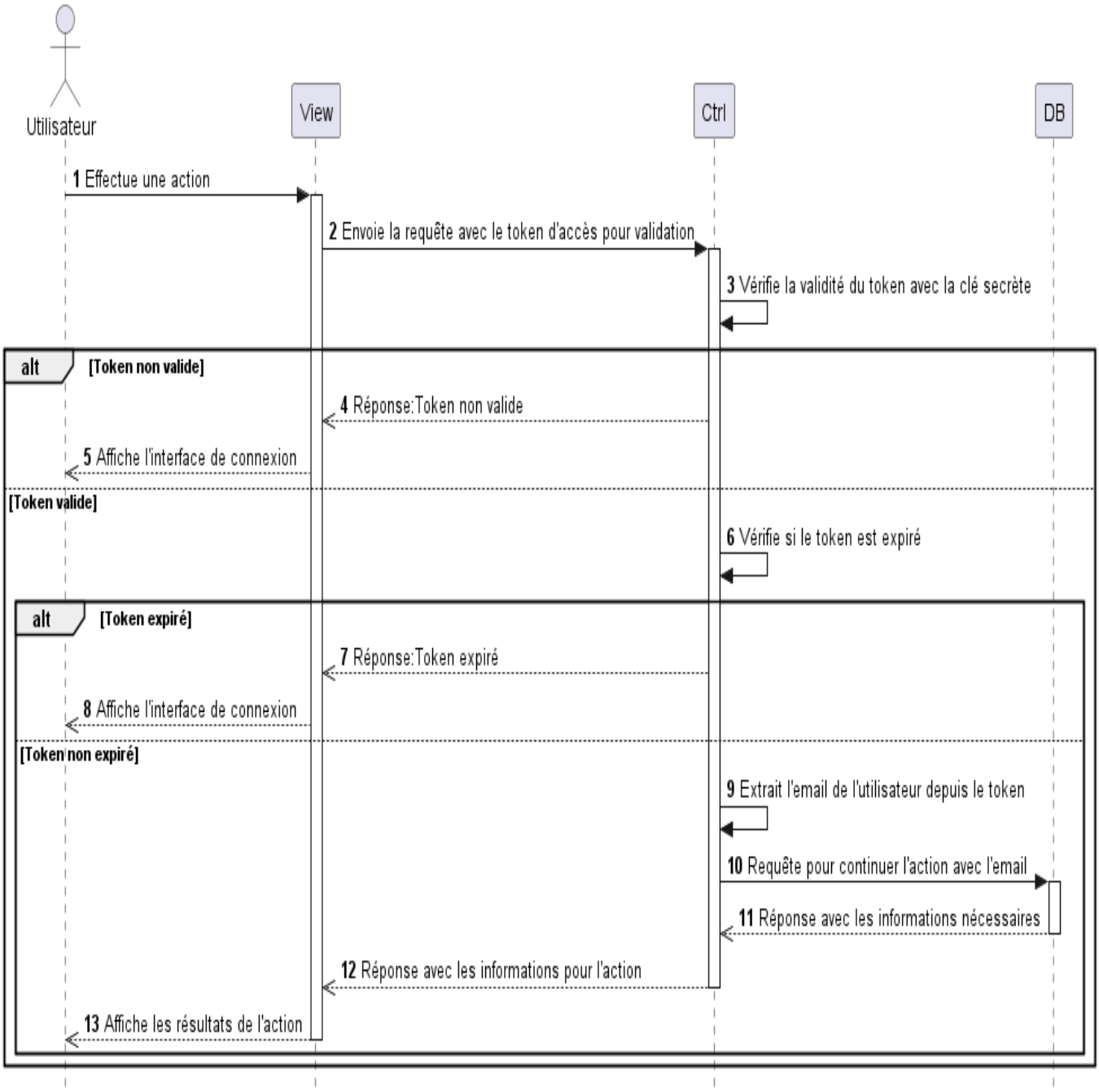

FIGURE 3.5 – Diagramme de séquence détaillé du cas d'utilisation "Vérifier l'Accès"

3.2.6 Diagramme de séquence détaillé du cas d'utilisation "Ajouter un Paiement"

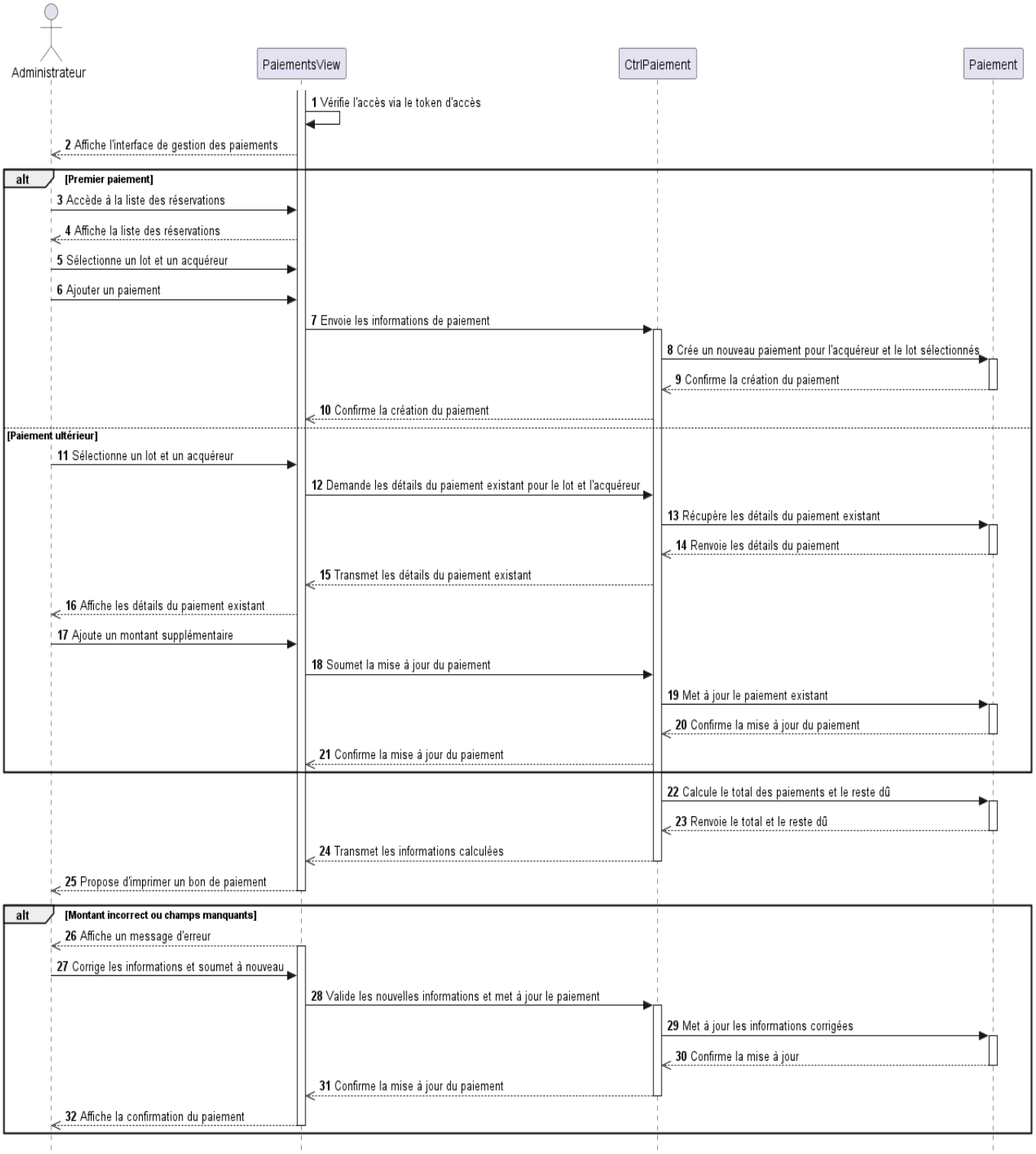

FIGURE 3.6 – Diagramme de séquence détaillé du cas d'utilisation "Ajouter un Paiement"

3.2.7 Diagramme de séquence détaillé du cas d'utilisation "Supprimer un Projet"

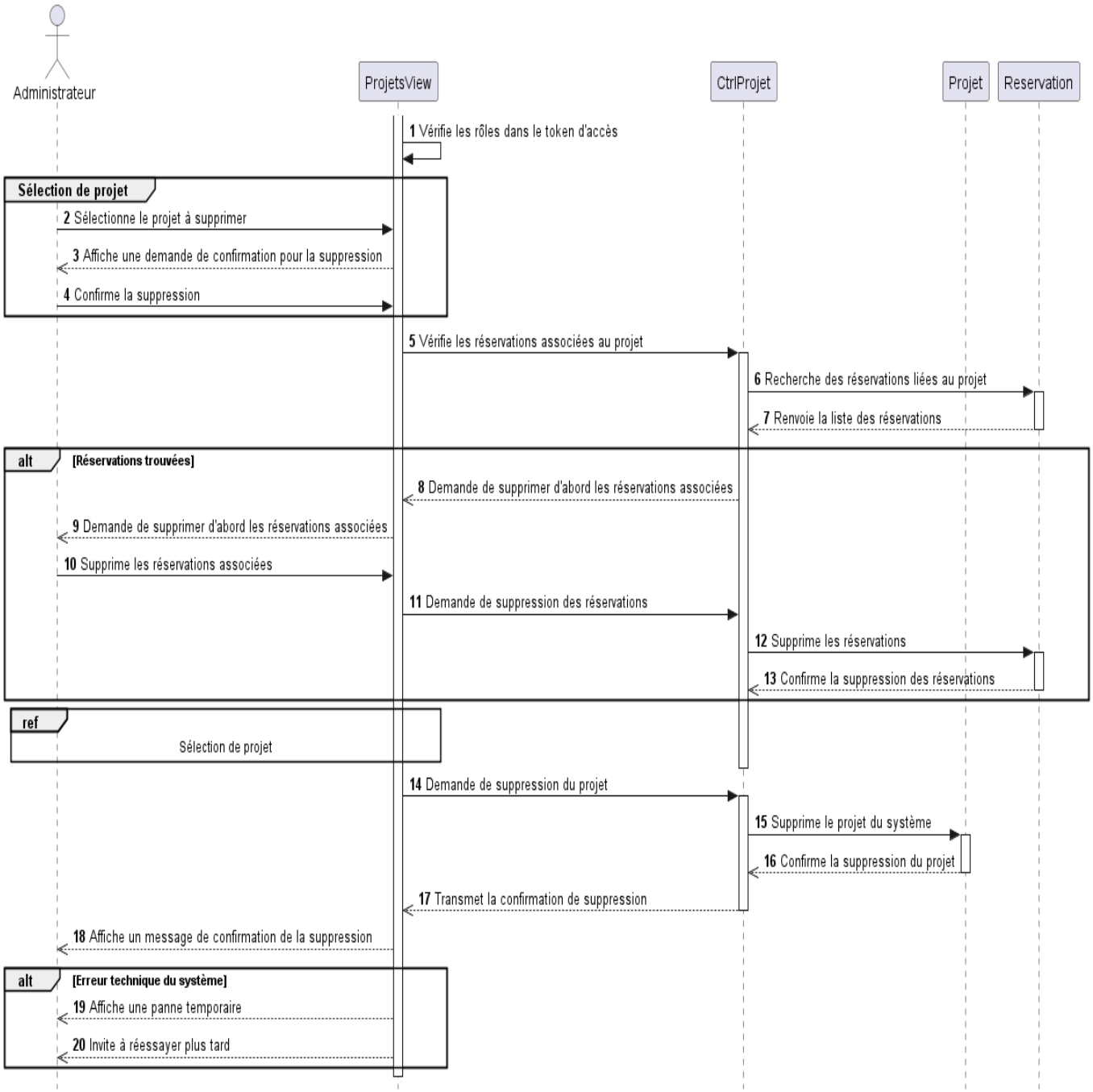

FIGURE 3.7 – Diagramme de séquence détaillé du cas d'utilisation "Supprimer un Projet"

3.3 Diagramme de classes

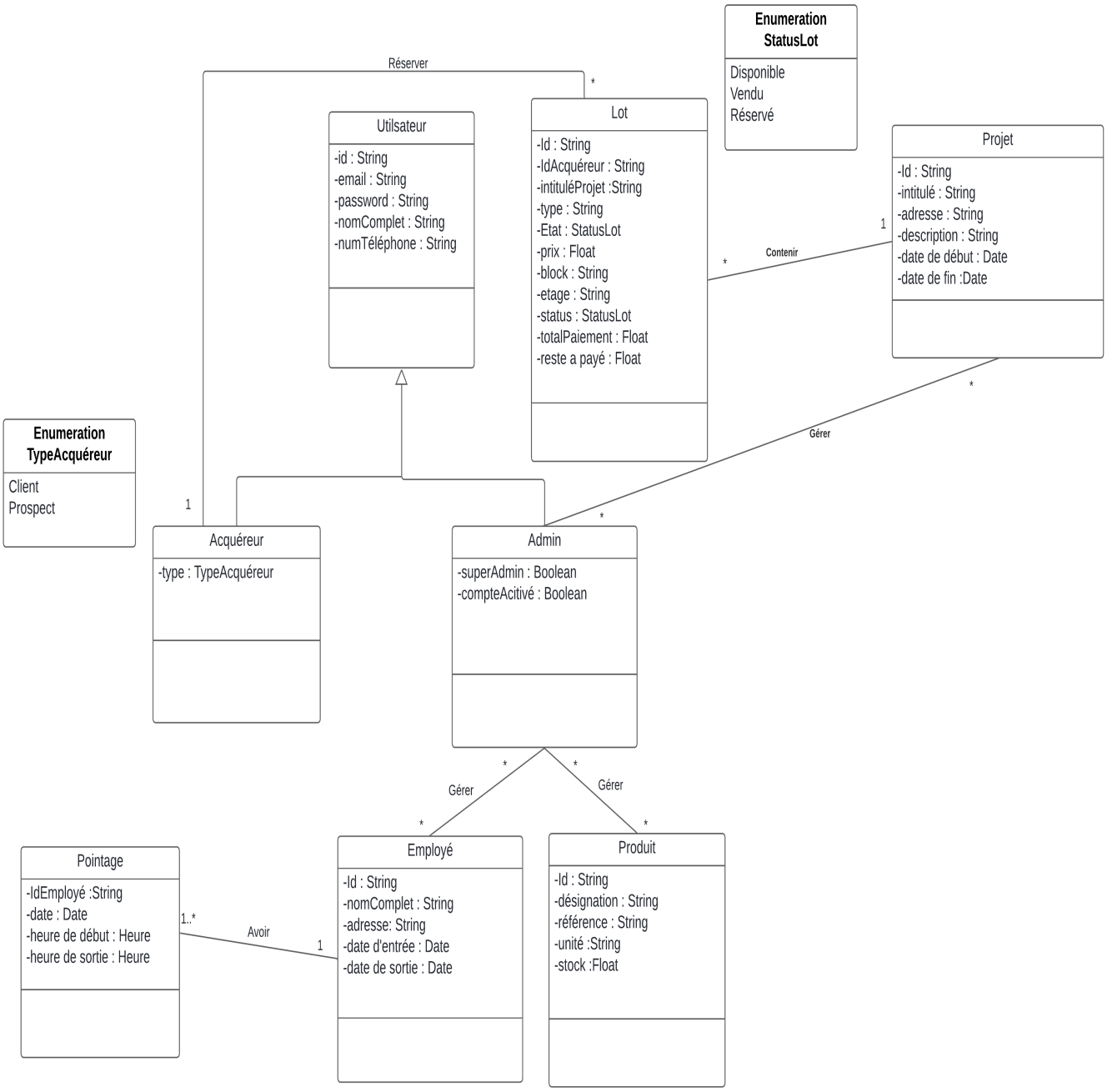

FIGURE 3.8 – Diagramme de classes.

3.4 Base de Données NoSQL :

Les bases de données NoSQL (Not Only SQL) sont une catégorie de systèmes de gestion de bases de données conçus pour gérer des données non structurées ou semi-structurées. Contrairement aux bases de données relationnelles traditionnelles qui utilisent des tables et des schémas fixes, les bases de données NoSQL offrent une flexibilité de schéma, permettant de gérer des données hétérogènes sans imposer de structure rigide. Elles se classent en plusieurs types en fonction de leur modèle de données, notamment les bases de données clé-valeur (comme Redis, DynamoDB), en colonne (comme Cassandra, HBase), documentaires (comme MongoDB, CouchDB), et orientées graphes (comme Neo4j, ArangoDB).

Les bases de données NoSQL offrent une flexibilité de schéma, une scalabilité horizontale, une haute disponibilité avec tolérance aux pannes, et une performance optimisée pour les applications modernes nécessitant des opérations rapides et une gestion efficace des données à grande échelle.

3.4.1 Base de Données NoSQL : MongoDB

MongoDB est l'une des bases de données NoSQL les plus populaires, principalement classée dans la catégorie des bases de données documentaires. Elle stocke les données sous forme de documents BSON (une forme binaire de JSON), ce qui permet une grande flexibilité dans la gestion des données complexes et hétérogènes. MongoDB est particulièrement adapté aux applications modernes qui nécessitent une scalabilité élevée, une gestion agile des données, et une grande performance.

3.4.1.1 Avantages de MongoDB

Flexibilité de Schéma : MongoDB permet de stocker des documents avec des structures variées, offrant une flexibilité maximale pour adapter les données aux besoins évolutifs des applications. Cette caractéristique est particulièrement utile dans les environnements de développement Agile.

Scalabilité Horizontale : MongoDB est conçu pour se scalabilité horizontalement grâce à son mécanisme de sharding, qui permet de distribuer les données sur plusieurs serveurs. Cela assure une gestion efficace des grands volumes de données et une réponse rapide aux requêtes.

Haute Disponibilité : Grâce à la réplication des données, MongoDB offre une haute disponibilité, garantissant que les données restent accessibles même en cas de panne de certains nœuds.

Gestion Efficace des Données Non Structurées : MongoDB est idéal pour stocker et gérer des données non structurées ou semi-structurées, comme les logs, les données de réseaux sociaux, ou les fichiers multimédias.

Facilité d'Intégration et de Déploiement : MongoDB est compatible avec de nombreux langages de programmation et peut être facilement intégré dans diverses architectures, facilitant ainsi son adoption dans les projets.

3.4.1.2 Pourquoi MongoDB pour Notre Projet

MongoDB a été choisi pour notre projet en raison de ses caractéristiques uniques qui répondent parfaitement à nos besoins spécifiques :

Flexibilité des Données : Notre projet nécessite de gérer des données hétérogènes et évolutives, où la structure des documents peut varier considérablement. MongoDB nous permet de stocker ces données sans imposer un schéma rigide, facilitant ainsi l'évolution du projet sans nécessiter de restructuration de la base de données.

Scalabilité : En prévision d'une croissance importante des données et du nombre d'utilisateurs, MongoDB offre une scalabilité horizontale qui nous permet de répartir la charge sur plusieurs serveurs, garantissant ainsi des performances optimales à grande échelle.

Agilité dans le Développement : MongoDB s'intègre parfaitement dans notre environnement de développement Agile, où les exigences peuvent évoluer rapidement. La flexibilité de schéma et la facilité d'ajout de nouvelles fonctionnalités sans impact majeur sur l'existant sont des atouts considérables.

Haute Disponibilité : Étant donné que notre projet nécessite une disponibilité continue, MongoDB, avec ses mécanismes de réplication, assure que notre système reste opérationnel même en cas de défaillance de certains composants.

Adaptation aux Données Non Structurées : Pour les aspects du projet nécessitant le stockage de données non structurées, comme les fichiers multimédias ou les logs d'activité, MongoDB se révèle être la solution idéale, gérant efficacement ces types de données tout en offrant une grande souplesse dans leur manipulation.

3.5 Conclusion

Au cours de ce chapitre nous avons en premier lieu modélisé les diagrammes de séquences qui nous ont donné une vue approfondie sur les interactions entre les objets et cela nous a permis de réaliser le diagramme de classe. Dans le prochain chapitre, nous plongerons dans les outils de développement et la conception des interfaces graphiques pour notre application.

4
Réalisation

4.1 Introduction

Dans ce chapitre, nous aborderons la phase de réalisation de notre projet en présentant les choix technologiques et les outils de développement utilisés. Nous commencerons par définir les outils essentiels, tels que les langages de programmation, les frameworks, et les autres outils qui ont joué un rôle clé dans le développement de l'application. Ensuite, nous explorerons l'architecture du système, en mettant en lumière les aspects de sécurité, notamment les mécanismes d'authentification et d'autorisation, essentiels pour garantir l'intégrité et la protection des données. Enfin, nous décrirons les interfaces utilisateur, en soulignant leur conception intuitive et leur importance pour une expérience utilisateur optimale. Ce chapitre illustre comment chaque décision technologique et chaque composant s'intègrent pour transformer notre vision en une solution logicielle robuste et sécurisée.

4.2 Technologies et Outils de Développement

Dans cette section, nous détaillerons les outils et technologies utilisés pour le développement de notre application, en commençant par les langages et frameworks, puis en abordant les outils de développement, les plateformes de collaboration, et enfin les solutions d'optimisation du code.

4.2.1 Langages et Frameworks

4.2.1.1 JavaScript :

JavaScript est un langage de programmation qui permet de créer des contenus dynamiques et interactifs sur les pages web. Il est principalement utilisé pour ajouter des fonctionnalités aux sites web, comme les menus déroulants, les animations, et les formulaires interactifs [8].

4.2.1.2 React

React est une bibliothèque JavaScript développée par Facebook, utilisée pour construire des interfaces utilisateur, notamment des applications web dynamiques à page unique. React permet aux développeurs de créer des composants réutilisables, qui gèrent leur propre état et peuvent être combinés pour créer des interfaces complexes de manière efficace [9].

4.2.1.3 Express

Express est un framework web rapide, minimaliste et flexible pour Node.js, qui fournit un ensemble de fonctionnalités robustes pour le développement d'applications web et mobiles. Il simplifie la gestion des requêtes HTTP, le routage, et l'intégration de middlewares, permettant aux développeurs de construire des applications web performantes de manière efficace [10].

4.2.2 Outils de Développement

4.2.2.1 Visual Studio Code

Visual Studio Code (VS Code) est un éditeur de code source libre développé par Microsoft. Il est conçu pour être léger et rapide, tout en offrant des fonctionnalités avancées telles que la coloration syntaxique, l'autocomplétion, et le débogage intégré. VS Code supporte une multitude de langages de programmation et est hautement personnalisable grâce à une vaste bibliothèque d'extensions. Ces extensions ajoutent des fonctionnalités comme des outils de linting, des intégrations avec des systèmes de gestion de version, et des environnements de développement personnalisés [11].

FIGURE 4.1 – VsCode Logo

4.2.2.2 Node js

Node.js est un environnement d'exécution JavaScript côté serveur basé sur le moteur V8 de Google Chrome. Il permet aux développeurs de créer des applications réseau rapides et évolutives en utilisant JavaScript. Node.js est conçu pour des opérations non bloquantes et orientées vers les événements, ce qui le rend particulièrement adapté aux applications nécessitant des échanges de données en temps réel, telles que les chats en ligne et les jeux en ligne. Il est également utilisé pour construire des API RESTful et des serveurs web [12].

4.2.2.3 Postman

Postman est un outil crucial pour tester le backend indépendamment du frontend. Il facilite la vérification et la validation des API en simulant des requêtes HTTP et en inspectant les réponses. Grâce à Postman, il a été possible de détecter et de corriger rapidement les bugs du backend avant l'intégration des fonctionnalités avec le frontend. Cet outil assure que le serveur répond correctement aux différentes requêtes, établissant ainsi une base solide pour le développement du frontend [13].

FIGURE 4.2 – Postman Logo

Pour illustrer l'utilisation de Postman dans notre projet, une image ci-dessous montre la configuration des requêtes et les résultats des tests.

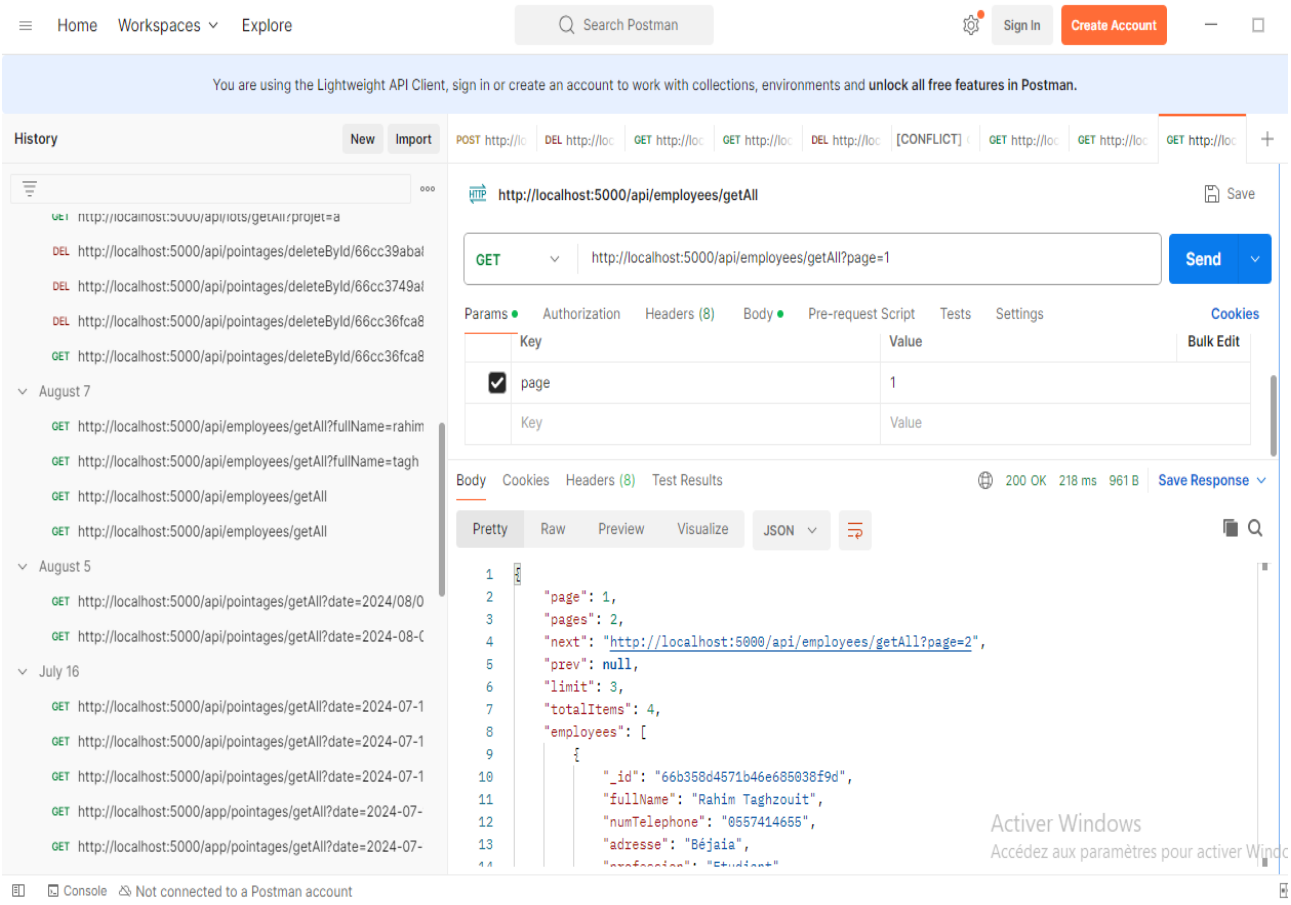

FIGURE 4.3 – Interface de Postman

4.2.3 Outils de Collaboration et Gestion de Projet

4.2.3.1 Git

Git est un système de contrôle de version distribué conçu pour gérer des projets de développement logiciel en suivant les modifications apportées au code source au fil du temps. Développé par Linus Torvalds en 2005, Git permet de gérer efficacement des versions multiples de fichiers et d'organiser les contributions de différents développeurs. Grâce à ses fonctionnalités comme les branches et les merges, Git facilite la collaboration en équipe et le suivi des évolutions du projet tout en maintenant un historique complet des changements effectués. Sa nature distribuée signifie que chaque développeur possède une copie complète de l'historique du projet, ce qui renforce la résilience et la flexibilité du processus de développement [14].

FIGURE 4.4 – Git Logo

4.2.3.2 GitHub

GitHub est une plateforme de développement collaboratif qui repose sur le système de contrôle de version Git. Lancé en 2008, GitHub offre des fonctionnalités supplémentaires telles que l'hébergement de dépôts Git, la gestion des demandes de tirage (pull requests), et la création d'issues pour le suivi des bugs et des tâches. GitHub facilite le partage du code entre développeurs et permet une collaboration fluide en offrant des outils pour la révision de code, l'intégration continue, et la documentation. Il est largement utilisé pour les projets open source et privés, offrant une interface utilisateur conviviale pour la gestion et la contribution au code source [15].

4.2.3.3 ClickUp

ClickUp est un outil de gestion de projet tout-en-un conçu pour améliorer la productivité et la collaboration des équipes. Il propose des fonctionnalités variées telles que la gestion des tâches, la planification de projets, et le suivi du temps. ClickUp permet de créer des listes de tâches, de définir des priorités, et de suivre l'avancement des projets à l'aide de tableaux de bord personnalisables. Il facilite également la communication entre les membres de l'équipe grâce à des outils de collaboration intégrés, comme les commentaires et les notifications. Ce logiciel est adapté à la gestion de projets de toute envergure, offrant une visibilité claire sur les objectifs et les progrès réalisés [16].
Pour illustrer l'utilisation de ClickUp dans notre projet, une image ci-dessous présente l'organisation des tâches et le suivi de l'avancement

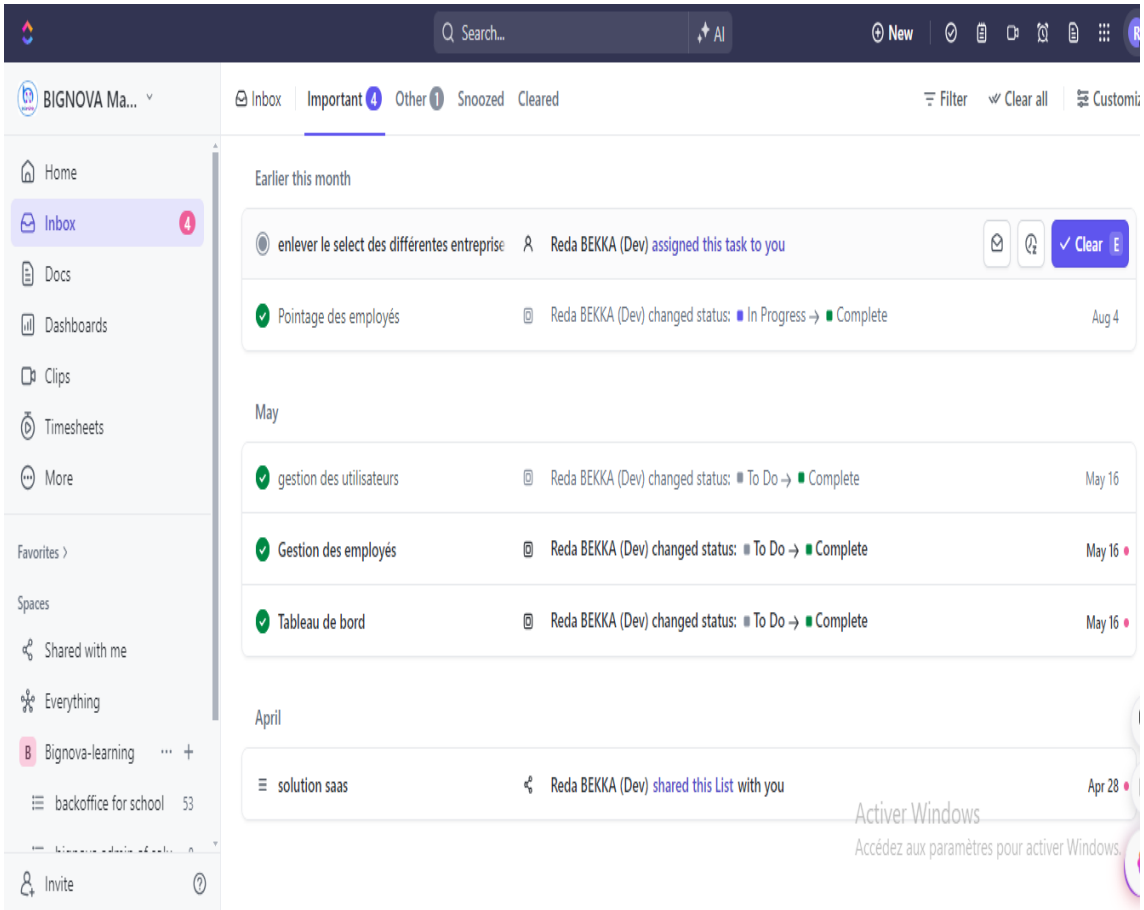

FIGURE 4.5 – clickUp interface

4.2.3.4 Swagger UI

Swagger UI est un outil de documentation et de test d'API qui permet de générer automatiquement une interface utilisateur interactive pour explorer les services API. Il offre une vue visuelle des endpoints d'API, des paramètres requis, et des réponses possibles. Swagger UI facilite la compréhension des API en fournissant une documentation vivante et interactive, permettant aux développeurs et aux intégrateurs de tester les appels d'API directement depuis l'interface. Il contribue à une meilleure communication entre les équipes de développement et les consommateurs d'API, en rendant les spécifications API plus accessibles et compréhensibles [17].

FIGURE 4.6 – Swagger UI Logo

4.2.3.5 MongoDB Atlas

MongoDB Atlas est une solution de base de données NoSQL hébergée dans le cloud, développée par MongoDB Inc. Atlas offre une infrastructure entièrement gérée pour déployer, gérer, et évoluer des bases de données MongoDB. Avec des fonctionnalités telles que la scalabilité automatique, la sécurité intégrée, et les sauvegardes automatisées, MongoDB Atlas permet aux développeurs de se concentrer sur le développement d'applications sans se soucier de la gestion des serveurs de base de données. Il est conçu pour offrir une performance élevée et une flexibilité optimale pour gérer des données non structurées ou semi-structurées dans un environnement cloud sécurisé [18].

4.2.4 Outils d'Optimisation du Code

4.2.4.1 Redux

Redux est une bibliothèque JavaScript dédiée à la gestion de l'état global des applications, principalement utilisée avec React, bien qu'elle puisse être intégrée avec d'autres frameworks. Créée par Dan Abramov et Andrew Clark, Redux repose sur le principe d'un store unique qui centralise l'état de l'application. Cette approche permet de gérer l'état de manière prévisible et uniforme, facilitant ainsi le suivi des changements et la gestion des interactions entre les composants. Redux utilise un modèle de flux de données unidirectionnel, où les actions sont dispatchées pour modifier l'état global à travers des reducers. Ce modèle simplifie le développement et le débogage d'applications complexes en offrant une architecture claire pour la gestion des données [19].

4.2.4.2 Axios

Axios est une bibliothèque JavaScript pour effectuer des requêtes HTTP, largement utilisée pour interagir avec les API côté client. Développée par Matt Zabriskie, Axios offre une interface simple et intuitive pour envoyer des requêtes HTTP, telles que GET, POST, PUT, DELETE, et plus encore. Axios est particulièrement apprécié pour sa capacité à gérer les requêtes et les réponses de manière asynchrone, grâce à des promesses, ce qui facilite la gestion des opérations asynchrones. En plus de simplifier les appels API, Axios intègre des fonctionnalités telles que la gestion des en-têtes HTTP, les interceptors pour manipuler les requêtes et les réponses, ainsi que l'authentification, permettant de configurer les tokens ou autres mécanismes de sécurité nécessaires pour protéger les API [20].

4.3 Architecture Client-Serveur

L'architecture client-serveur est un modèle de conception logicielle dans lequel les tâches sont réparties entre des fournisseurs de services appelés serveurs et des demandeurs de services appelés clients. Dans ce modèle, le client est une application ou un appareil qui envoie des requêtes au serveur pour accéder à des services, des données ou des ressources. Le serveur, quant à lui, reçoit les requêtes du client, les traite, et retourne les réponses appropriées.

Cette architecture est largement utilisée dans les applications web modernes, où le frontend (le client) est souvent une application web ou mobile qui interagit avec un backend (le serveur) via des API (interfaces de programmation d'applications). Cette séparation des responsabilités permet une meilleure gestion des ressources, une sécurité accrue, et une évolutivité facilitée.

4.3.1 Architecture du Backend

Dans le contexte du backend, l'architecture est souvent structurée autour de plusieurs composants clés :

Modèle (Model) : Le modèle représente la structure des données, la logique métier, et les règles de gestion. Il interagit directement avec la base de données pour effectuer des opérations de lecture, écriture, mise à jour, et suppression.

Contrôleur (Controller) : Le contrôleur agit comme un intermédiaire entre le modèle et le client. Il reçoit les requêtes HTTP du client, appelle les méthodes du modèle nécessaires pour traiter ces requêtes, et retourne une réponse sous forme de données, généralement en JSON, au client.

Routeur (Router) : Bien que souvent regroupé avec le contrôleur, le routeur est responsable de l'acheminement des requêtes HTTP vers le contrôleur approprié en fonction de l'URL et de la méthode HTTP utilisée (GET, POST, etc.).

4.3.2 Avantages de l'Architecture Client-Serveur

L'architecture client-serveur présente plusieurs avantages clés :

Modularité et Séparation des Préoccupations : En séparant le frontend et le backend, les équipes de développement peuvent travailler indépendamment sur chaque partie, ce qui permet une spécialisation accrue et une meilleure gestion des projets.

Évolutivité : Cette architecture permet de faire évoluer le serveur ou le client indépendamment. Par exemple, le backend peut être mis à jour pour gérer plus de requêtes sans nécessiter de modifications dans le frontend.

Sécurité : Le backend peut être isolé du public, ce qui permet de protéger les données sensibles et d'appliquer des règles de sécurité strictes. Le frontend n'accède aux données que via des API sécurisées.

Réutilisabilité : Les API fournies par le backend peuvent être réutilisées par différentes applications clientes (web, mobile, desktop), ce qui réduit les efforts de développement.

4.3.3 APIs et Interactions avec HTTP

Une API est un ensemble de définitions et de protocoles qui permettent à deux applications ou plus de communiquer entre elles. Dans le contexte de l'architecture client-serveur, une API expose des fonctionnalités du backend au frontend via des points d'entrée définis, appelés endpoints.

Les interactions entre le client et le serveur se font principalement par le biais de requêtes HTTP. Chaque requête HTTP comprend une méthode (GET, POST, PUT, DELETE), une URL, et éventuellement des en-têtes et un corps de requête. Le serveur traite cette requête en fonction de l'URL et de la méthode, puis renvoie une réponse, qui peut inclure des données (généralement au format JSON), des codes d'état HTTP (comme 200 pour une requête réussie ou 404 pour une ressource non trouvée), et des en-têtes supplémentaires.

Cette interaction basée sur HTTP est à la base des API RESTful, un style architectural qui repose sur l'utilisation de requêtes HTTP pour effectuer des opérations CRUD (Créer, Lire, Mettre à jour, Supprimer) sur les ressources gérées par le serveur.

4.4 Présentation des interfaces de l'application

Dans cette section, nous allons présenter les différentes interfaces utilisateur développées pour l'application. L'interface utilisateur joue un rôle crucial dans l'expérience utilisateur en fournissant un accès intuitif et efficace aux fonctionnalités offertes par le système.

4.4.1 Tableau de Bord

Le tableau de bord est l'interface principale de l'application, offrant une vue d'ensemble des informations essentielles comme l'état des projets, les paiements mensuels, et les réservations. Conçu pour être clair et intuitif, il permet aux utilisateurs de suivre facilement les indicateurs clés et de prendre des décisions rapidement grâce à des données présentées de manière concise et visuelle.

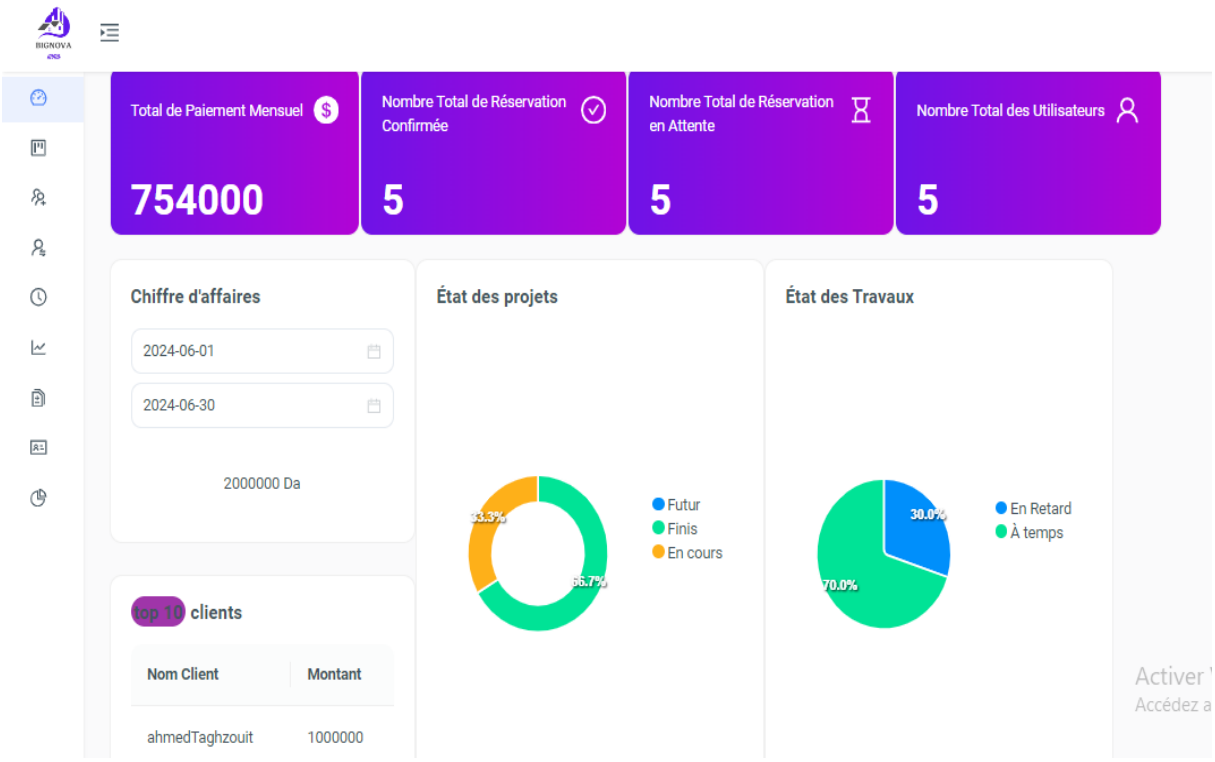

FIGURE 4.7 – Tableau de board

4.4.2 Page d'authentification

c'est l'interface principale de l'application ,elle permet aux utilisateurs d'accéder à l'application en saisissant un email et un mot de passe pour s'authentifier.Depuis cette page ,il pourra accéder aux autres interfaces.

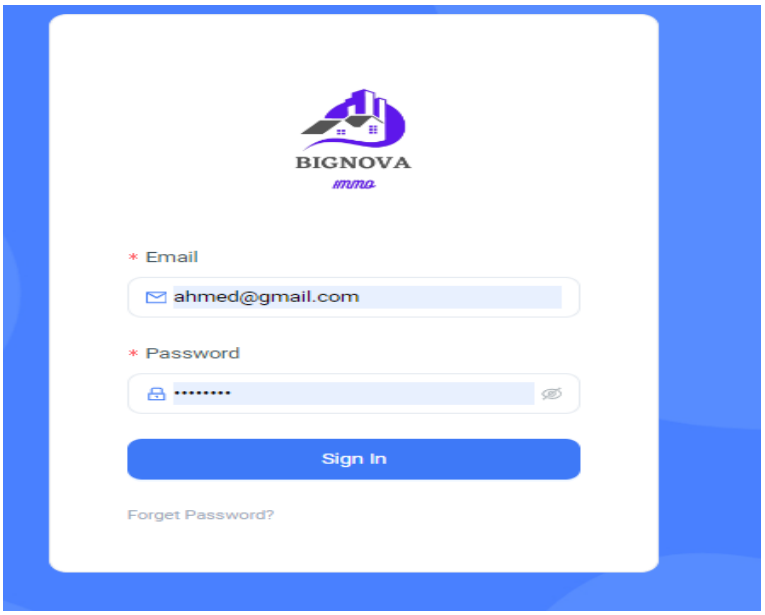

FIGURE 4.8 – page d'authentifiction

4.4.3 Gestion des admins

Pour la gestion des administrateurs, deux pages distinctes ont été développées :

Cette page permet de créer de nouveaux administrateurs et de définir leurs droits d'accès au sein de l'application.

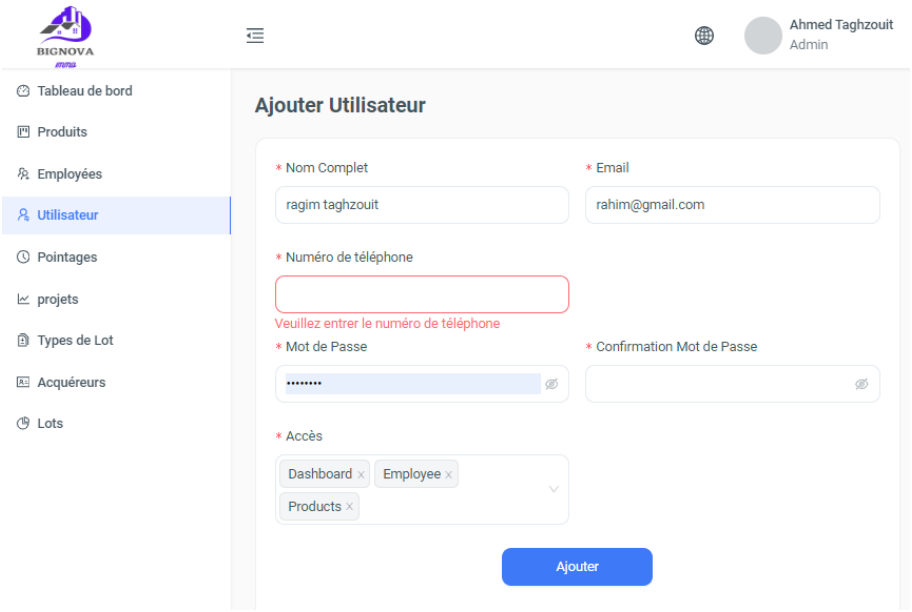

FIGURE 4.9 – Page de création et définition des accès pour les administrateurs

La deuxième page affiche la liste complète des administrateurs avec leurs statuts respectifs (activé ou désactivé).

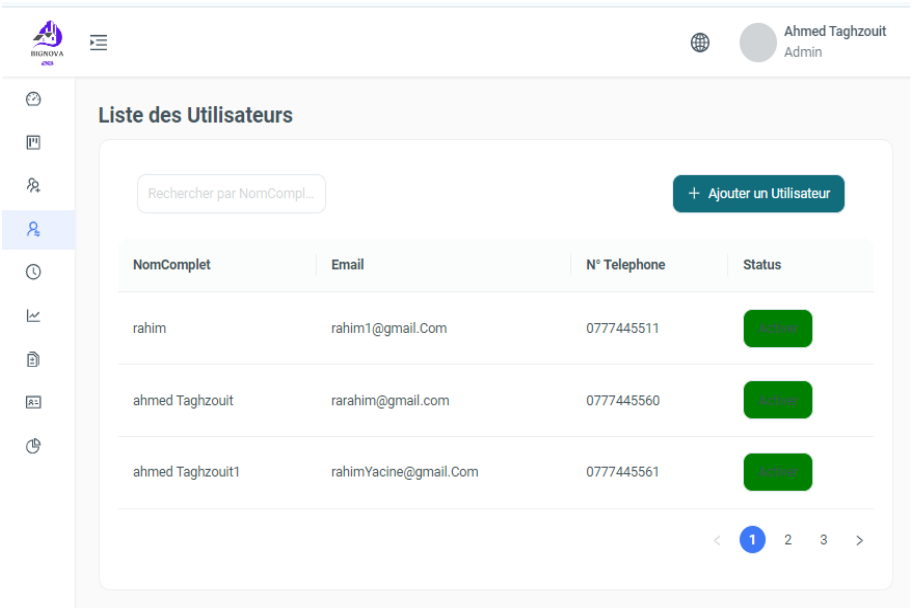

FIGURE 4.10 – Page de gestion de la liste des administrateurs et de leurs statuts

4.4.4 Gestion des employés et leur pointages

Cette page permet de consulter et de modifier les informations détaillées des employés

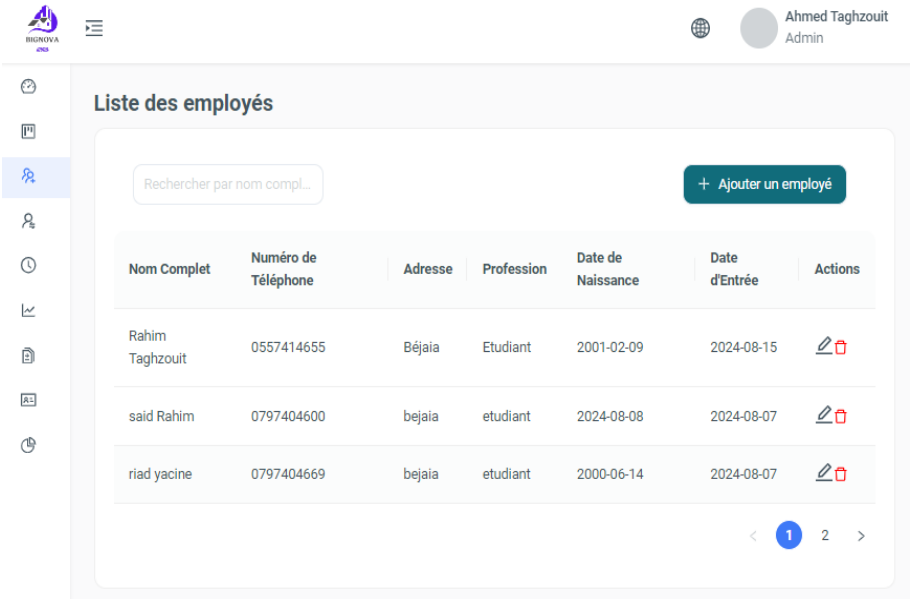

FIGURE 4.11 – Page d'informations sur les employés

La deuxième page affiche la liste complète des pointages effectués par les employés. Elle permet de visualiser les horaires de pointage

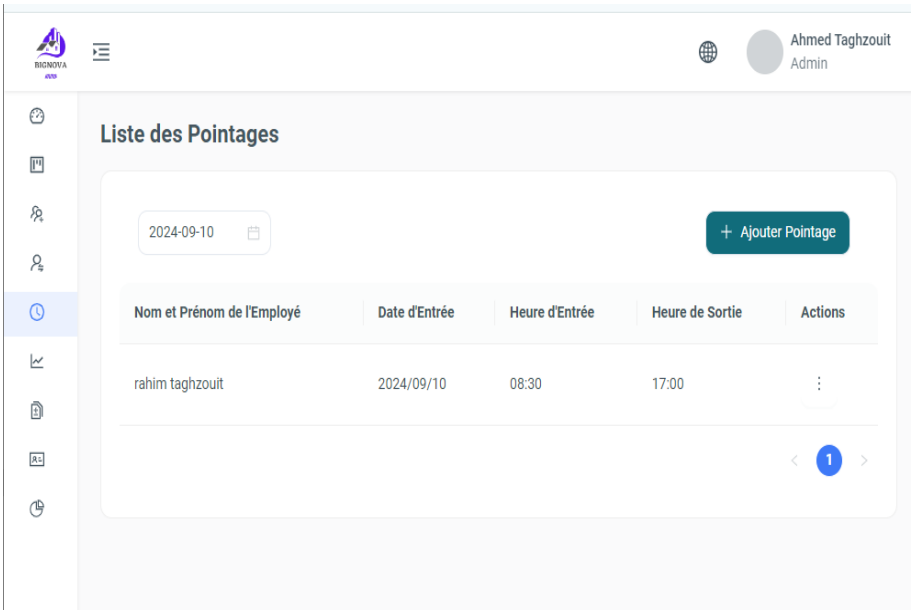

FIGURE 4.12 – Page de gestion des pointages des employés

La troisième page permet l'ajout manuel de nouveaux pointages pour les employés.

FIGURE 4.13 – Page pour ajouter un pointage pour les employés

4.4.5 Gestion des lots et acquéreurs

Cette page permet de consulter les détails complets des lots

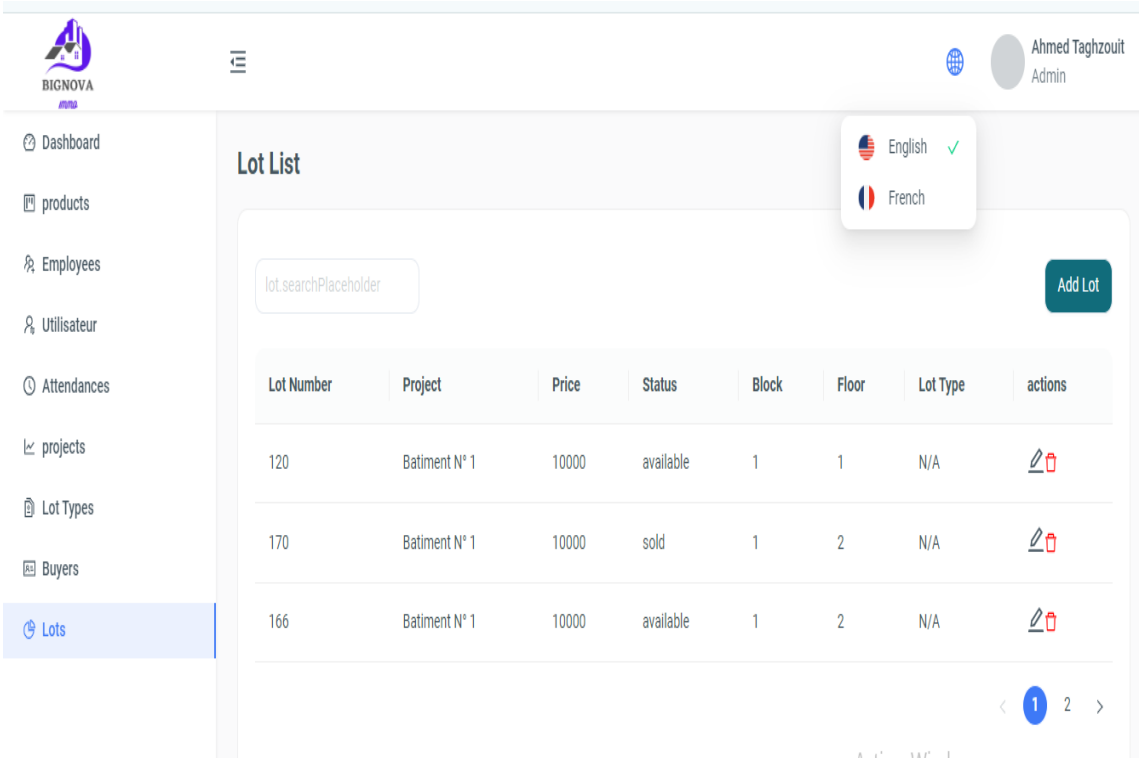

FIGURE 4.14 – Page d'informations détaillées et de modification des lots

La deuxième page est dédiée à la gestion des acquéreurs. Elle affiche la liste des acquéreurs avec leurs informations pertinentes, et permet de suivre leur statut, ainsi que d'effectuer des modifications si nécessaire.

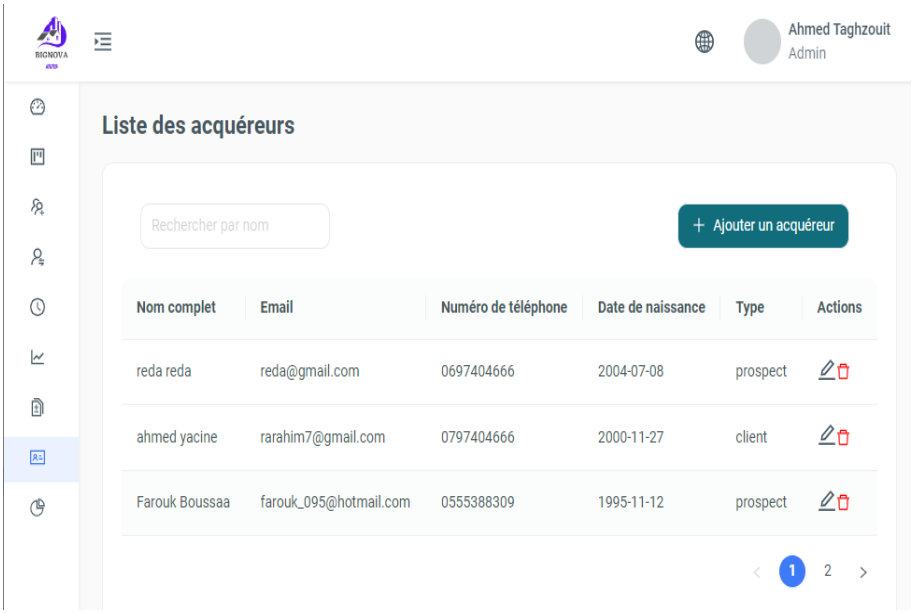

FIGURE 4.15 – Page de gestion des acquéreurs

4.5 Conclusion

Dans ce chapitre, nous avons détaillé la phase de réalisation de notre projet en présentant les choix technologiques et les outils utilisés, ainsi que l'architecture du système. Nous avons exploré les langages de programmation, les frameworks, et les outils de développement, tout en mettant en évidence l'importance de chaque composant pour assurer une solution robuste et sécurisée. Les interfaces utilisateur ont été décrites pour illustrer la manière dont elles facilitent l'interaction avec l'application. Ce chapitre montre comment chaque élément contribue à transformer notre vision en une application fonctionnelle et performante.

Conclusion Générale et Perspectives

Dans ce mémoire, nous avons conçu et développé une application complète pour la gestion des projets immobiliers, répondant aux besoins spécifiques des entreprises du secteur. Le projet s'est déroulé en plusieurs phases distinctes. Tout d'abord, nous avons réalisé une étude approfondie de l'état actuel du marché immobilier et des solutions existantes. Cette analyse nous a permis de cerner les défis et les opportunités, ainsi que de définir les objectifs précis que notre application devait atteindre. En utilisant la méthode Unified Process (UP) et les diagrammes de modélisation UML, nous avons spécifié et analysé les besoins des utilisateurs, ce qui a conduit à la conception détaillée de notre application. Enfin, nous avons développé "Bignova Immo" en intégrant les outils et technologies les plus adaptés, tout en veillant à offrir une solution innovante et efficace pour la gestion des projets immobiliers.

La réalisation de ce projet a été une expérience extrêmement enrichissante, qui a considérablement approfondi mes connaissances en développement web, notamment avec des technologies telles que React et Node.js. Ce travail m'a permis d'acquérir des compétences essentielles en matière de sécurité, notamment en authentification et en autorisation, renforçant ainsi ma compréhension des pratiques sécuritaires nécessaires pour protéger les données sensibles. J'ai également appris à organiser le code de manière professionnelle, en adoptant des pratiques de développement efficaces et structurées. De plus, ce projet m'a permis de renforcer mes compétences en travail d'équipe, en utilisant des outils de gestion de projet tels que Git et ClickUp, ce qui m'a aidé à améliorer mes compétences en collaboration et en gestion de projet. Cette expérience m'a non seulement préparé à relever des défis techniques, mais elle m'a aussi donné un aperçu concret de la dynamique du travail en entreprise et des exigences du milieu professionnel.

Des perspectives d'amélioration pour notre application “Bignova Immo” sont en cours d'exploration afin de répondre de manière encore plus précise aux besoins des entreprises immobilières. Nous envisageons plusieurs développements clés pour enrichir et optimiser notre solution :

Renforcement de la gestion des employés : Nous prévoyons d'intégrer des fonctionnalités avancées pour la gestion des salaires, permettant ainsi une administration plus complète et précise des aspects financiers liés aux employés. Cela inclura le calcul automatique des salaires, la gestion des déductions et des bonus, ainsi que la génération de rapports détaillés.

Amélioration de la gestion des produits : Nous souhaitons

développer davantage la gestion des produits en intégrant la gestion des factures et des fournitures. Cette fonctionnalité permettra de suivre les achats, les coûts associés, et de maintenir un inventaire précis et à jour, facilitant ainsi la gestion financière et opérationnelle des produits.

Optimisation de la gestion des projets : L'ajout de la fonctionnalité de suivi en temps réel des projets est prévu pour améliorer la visibilité et la gestion des projets en cours. Cette fonctionnalité offrira des outils de suivi détaillés, permettant aux utilisateurs de surveiller les progrès en temps réel, de détecter rapidement les problèmes et d'ajuster les plans en conséquence.

Ces améliorations visent à offrir une solution plus complète et intégrée, répondant ainsi aux exigences croissantes des entreprises immobilières et contribuant à une gestion plus efficace et fluide de leurs opérations.

Bibliographie

[1] "Bignova." https://bignova-company.com/.

[2] "Ebp bâtiment." https://www.grafe.fr/solutions/ebp-expert-batiment, Consulté le : 1 juin 2024.

[3] "Fatoura." https://fatoura.app/stock.

[4] "Gp-immo." https://www.ylarsoft.com/gp-immo/.

[5] R. S. Pressman and B. R. Maxim, *Software Engineering : A Practitioner's Approach*. McGraw-Hill Education, 8th ed., 2014.

[6] "Json web token." https://jwt.io/introduction/, Consulté le : 10 juin 2024.

[7] "hachage des mot de passe." https://www.dashlane.com/fr/blog/quest-ce-que-le-hachage-des-mots-de-passe, Consulté le : 10 juin 2024.

[8] "Javascript." https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction, Consulté le : 7 juillet 2024.

[9] "React js." https://react.dev/, Consulté le : 7 juillet 2024.

[10] "Express js." https://expressjs.com/fr/, Consulté le : 7 juillet 2024.

[11] "Vscode." https://code.visualstudio.com/docs,Consulté le : 7 juillet 2024.

[12] "Nodejs." https://nodejs.org/fr, Consulté le : 7 juillet 2024.

[13] "Postman." https://www.postman.com/, Consulté le : 7 juillet 2024.

[14] "Git." https://git-scm.com/, Consulté le : 7 juillet 2024.

[15] "Github." https://docs.github.com/fr, Consulté le : 7 juillet 2024.

[16] "Clickup." https://clickup.com/about, Consulté le : 7 juillet 2024.

[17] "Swagger ui." https://swagger.io/tools/swagger-ui/, Consulté le : 7 juillet 2024.

[18] *MongoDB Atlas*. https://www.mongodb.com/products/platform/atlas-database, Consulté le : 7 juillet 2024.

[19] "Redux." https://redux.js.org/introduction/getting-started#help-and-discussion,Consulté le : 7 juillet 2024.

[20] "Axios." https://axios-http.com/docs/intro, Consulté le : 7 juillet 2024.

Résumé

Ce mémoire présente le développement de l'application Bignova Immo, une solution web et mobile dédiée à la gestion intégrée des projets immobiliers. L'objectif principal est de fournir aux entreprises immobilières un outil puissant et convivial pour optimiser la gestion de leurs projets, produits, et employés. L'application Bignova Immo propose des fonctionnalités complètes telles que la gestion des projets immobiliers, des employés, des produits, ainsi que des outils avancés pour le suivi des paiements et des réservations. En combinant des technologies modernes avec une interface intuitive, cette solution vise à améliorer l'efficacité opérationnelle des professionnels de l'immobilier et à faciliter la coordination au sein de leurs équipes. Ce projet représente une avancée notable dans le domaine du développement logiciel, répondant aux défis spécifiques du secteur immobilier avec une approche centrée sur l'utilisateur.

Mots clés : Gestion immobilière, Application web, Application mobile, Gestion des projets, Gestion des employés, React, Node.js.

Abstract

This thesis presents the development of Bignova Immo, a comprehensive web and mobile application designed for integrated real estate project management. The main objective is to provide real estate companies with a robust and user-friendly tool to optimize the management of their projects, products, and employees. The Bignova Immo application offers a full suite of features including real estate project management, employee management, product management, and advanced tools for tracking payments and reservations. By combining modern technologies with an intuitive interface, this solution aims to enhance the operational efficiency of real estate professionals and facilitate coordination within their teams. This project marks a significant advancement in software development, addressing the specific challenges of the real estate sector with a user-centered approach.

Keywords : Real estate management, Web application, Mobile application, Project management, Employee management, React, Node.js.