PROJET DE FIN D’ETUDE

*5éme Année en Ingénierie Informatique et Réseaux*

**Solution Centralisée de Gestion Client : Développement d'un Tableau de Bord Intégré pour une Gestion Efficace des Réclamations et des Notifications**

***Réalisé par :***

*Hosni Tarik*

***Tuteur (s) :***

*Encadrant Professionnel : Nassim Khalil*

*Encadrant Pédagogique : Rachid Dehbi*

*Au sein de SOGERTEL :*



# Remerciements

Je tiens à exprimer ma profonde gratitude envers **M. Dehbi Rachid** mon encadrant pédagogique et **M. Khalil Nassim** mon encadrant professionnel pour leur accompagnement attentif et leur précieux soutien tout au long de mon projet de fin d’études à **l’École Marocaine des Sciences d’Ingénieur**. Leurs conseils éclairés, leur expertise et leur disponibilité ont été essentiels à la réussite de ce projet.

Je souhaite également remercier chaleureusement **l’École Marocaine des Sciences d’Ingénieur** pour m’avoir offert cette opportunité d’apprentissage et de développement. Les enseignements de qualité que j’ai reçus ont joué un rôle déterminant dans la construction de mes compétences.

En outre, je tiens à exprimer ma reconnaissance envers l’entreprise **SOGERTEL**, où j’ai effectué mon stage. Je suis reconnaissant envers tous les membres de l’équipe qui m’ont accueilli et qui m’ont offert un environnement de travail enrichissant. Leur expertise, leur soutien et leur collaboration ont grandement contribué à mon expérience professionnelle.

Enfin, je souhaite adresser mes remerciements les plus sincères à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet de fin d’études. Votre appui, vos conseils et votre implication ont été d’une valeur inestimable.

Je suis reconnaissant envers **M. Dehbi Rachid**, **M. Khalil Nassim**, **l’École Marocaine des Sciences d’Ingénieur** et l’entreprise **SOGERTEL** pour leur rôle essentiel dans mon parcours académique et professionnel.

# Résumé

Le projet vise à fournir une solution intégrée de gestion des réclamations clients et l'envoi de notifications, spécifiquement conçue pour répondre aux besoins de l'entreprise. Cette application permettra de simplifier et d'accélérer le processus de gestion des réclamations, en réduisant les erreurs et les délais de traitement, tout en améliorant la communication avec les clients.

Grâce à cette application, les administrateurs pourront efficacement gérer les réclamations reçues, les trier par état (en attente, résolues) et mettre à jour leur statut en temps réel. L'envoi de notifications automatisées aux clients les informera de l'avancement de leurs réclamations, assurant ainsi une transparence et une réactivité accrues. De plus, l'application fournira des statistiques détaillées sur le nombre de réclamations, le temps moyen de résolution et d'autres indicateurs clés de performance, facilitant l'identification des domaines nécessitant des améliorations.

L'application permettra également une gestion optimale des comptes clients. Les administrateurs auront la possibilité d'activer ou de désactiver des comptes, de gérer les profils utilisateurs et de consulter les informations des clients de manière centralisée. Cette solution intégrée vise à améliorer l'expérience utilisateur, à optimiser la gestion interne des réclamations et à renforcer la satisfaction client en garantissant une communication fluide et efficace entre l'entreprise et ses clients.

# Abstract

The project aims to provide an integrated solution for managing claims of customers and sending notifications, specifically designed to solve the needs of the company. This application will simplify and accelerate the claims management process, reducing errors and processing times while improving communication with customers.

With this application, administrators will be able to efficiently manage received claims, sort them by status (pending, resolved), and update their status in real-time. Automated notifications sent to customers will inform them of the progress of their claims, ensuring increased transparency and responsiveness. Additionally, the application will provide detailed statistics on the number of claims, average resolution time, and other key performance indicators, facilitating the identification of areas needing improvement.

The application will also enable optimal management of customers accounts. Administrators will have the ability to activate or deactivate accounts, manage user profiles, and view customer information in a centralized manner. This integrated solution aims to enhance the user experience, optimize internal claims management, and increase customer satisfaction by ensuring smooth and effective communication between the company and its clients.

**Table des matières**

[Remerciements 2](#_Toc170826825)

[Résumé 3](#_Toc170826826)

[Abstract 4](#_Toc170826827)

1 [Contexte générale du projet 8](#_Toc170826828)

[1.1 Introduction générale 9](#_Toc170826829)

[1.2 Présentation de l’organisme d’accueil 9](#_Toc170826830)

[1.2.1 Présentation de la société SOGERTEL 9](#_Toc170826831)

[1.2.2 Domaine d’activité de SOGERTEL 9](#_Toc170826832)

[1.2.3 Les clients principaux de SOGERTEL 10](#_Toc170826833)

[1.3 Étude de l’Existant 11](#_Toc170826834)

[1.4 Problématique 11](#_Toc170826835)

[1.5 Objectifs Fonctionnels 12](#_Toc170826836)

[1.6 Objectifs Non Fonctionnels 12](#_Toc170826837)

[1.6 Conclusion 13](#_Toc170826838)

2 [Conduite Du Projet 14](#_Toc170826839)

[La Conduite du Projet 15](#_Toc170826840)

[2.1 Processus de développement 15](#_Toc170826841)

[2.1.1 Méthode Agile 15](#_Toc170826842)

[2.1.2 Méthode Scrum 15](#_Toc170826843)

[2.1.2.1 Pôle de compétences 16](#_Toc170826844)

[2.1.2.2 Rôle et rituels 16](#_Toc170826845)

[2.1.2.3 Artefacts Scrum 17](#_Toc170826846)

[2.1.2.4 Diagramme de Gantt 18](#_Toc170826847)

[2.2 Planification du projet 20](#_Toc170826848)

[2.2.1 Sprint 1 : Analyse et Conception 20](#_Toc170826849)

[2.2.2 Sprint 2 : Développement de la gestion des Réclamations 20](#_Toc170826850)

[2.2.3 Sprint 3 : Développement de la gestion des Notifications et améliorations 20](#_Toc170826851)

[2.2.4 Sprint 4 : Tests et Documentation Technique 21](#_Toc170826852)

[2.3 Conclusion 21](#_Toc170826853)

3 [Analyse et Conception 22](#_Toc170826854)

[3.1 Introduction 23](#_Toc170826855)

[3.2 Le formalisme UML 23](#_Toc170826856)

[﻿3.2.1 Le choix d'UML 23](#_Toc170826857)

[3.3 Démarche conception 24](#_Toc170826858)

[3.3.1 Analyse fonctionnelle 24](#_Toc170826859)

[3.3.2 Analyse fonctionnelle 35](#_Toc170826980)

[3.3.2.1 Diagramme de séquence Authentification 35](#_Toc170826981)

[3.3.2.2 Diagramme de séquence Consulter les Clients 36](#_Toc170826982)

[3.3.2.3 Diagramme de séquence Consulter Les Notifications 37](#_Toc170826983)

[3.3.2.4 Diagramme de séquence Consulter les Réclamations 38](#_Toc170826984)

[3.3.2.5 Diagramme de séquence Traiter les réclamations des clients 39](#_Toc170826985)

[3.3.2.6 Compte activé désactivé admin 40](#_Toc170826986)

[3.3.2.7 Diagramme de séquence Supprimer une Notification 41](#_Toc170826987)

[3.3.2.8 Diagramme de séquence Modifier une Notification 42](#_Toc170826988)

[3.3.2.9 Diagramme de séquence Créer compte pour un client 43](#_Toc170826989)

[3.3.2.10 Diagramme de séquence Client Consulte Ses Réclamations 44](#_Toc170826990)

[3.3.2.11 Diagramme de séquence Client Consulte Ses Notifications 45](#_Toc170826991)

[3.3.2.12 Diagramme de Séquence pour le Processus D’envoyer Réclamation Client 46](#_Toc170826992)

[3.3.2.13 Diagramme de Séquence modifier le contenu de la réclamation 47](#_Toc170826993)

[3.3.2.14 Diagramme de Séquence Supprimer une réclamation 48](#_Toc170826994)

[3.3.2.15 Diagramme de Séquence Client Modifie ses Coordonnées 49](#_Toc170826995)

[3.3.3 Analyse objet 50](#_Toc170826996)

[3.4 L’architecture Des MicroServices de L’application 51](#_Toc170826997)

[Résumé 53](#_Toc170826998)

[3.5 Conclusion 53](#_Toc170826999)

4 [Mise en œuvre 54](#_Toc170827000)

[4.1 Outils et Technologies 55](#_Toc170827001)

[4.1.1 Spring boot 55](#_Toc170827002)

[4.1.2 Mongo DB 55](#_Toc170827003)

[4.1.3 JavaScript 56](#_Toc170827004)

[4.1.4 TypeScript 56](#_Toc170827005)

[4.1.5 React JS 57](#_Toc170827006)

[4.1.6 Redux 57](#_Toc170827007)

[4.1.7 Itellij IDEA 58](#_Toc170827008)

[4.1.8 Visual Studio Code 59](#_Toc170827009)

[4.1.9 Postman 59](#_Toc170827010)

[4.1.10 Tailwind CSS 60](#_Toc170827011)

**Chapitre 1**

# Contexte générale du projet

* **Présentation de l’organisme d’accueil**
* **Etude de l’existant**

## 1.1 Introduction générale

Dans le cadre de ma mission, j'ai été chargé de développer une application de gestion des réclamations et des notifications. Cette application vise à améliorer la gestion des réclamations clients et à assurer une communication efficace via des notifications. Elle simplifie et optimise l'expérience utilisateur, de la soumission de la réclamation à sa résolution et à la réception des notifications pertinentes.

L'application se distingue par sa précision fonctionnelle et sa convivialité. L'objectif est de fournir une solution pratique et intuitive répondant aux besoins des administrateurs et des clients.

Ce rapport détaille chaque étape du développement, en se concentrant sur les choix méthodologiques et stratégiques. Chaque phase du processus est analysée, éclairant les décisions et leurs justifications. Cette documentation servira de base pour la maintenance et l'évolution de l'application, garantissant sa pérennité et son efficacité à long terme.

## 1.2 Présentation de l’organisme d’accueil

### 1.2.1 Présentation de la société SOGERTEL

**SOGERTEL** est une entreprise d’informatique créative qui aide les sociétés et les startups dans leur transformation digitale. Elle offre des services de formation, de planiﬁcation et de design, de déploiement et de support.

Son siège social est à Casablanca, au Maroc, fondée en Février 2008 et son objectif est de faire connaître les dernières technologies et aider à l'intégration aux entreprises technologiques.

Figure 1.1: LOGO DE L'ENTREPRISE

### 1.2.2 Domaine d’activité de SOGERTEL

Acteur majeur sur le marché de la distribution de solution IT et basé sur un modèle de vente exclusivement indirect, c’est en véritable distributeur m valeur ajoutée que nous nous positionnons sur un marché en perpétuelle évolution technologique. A ﬁn d’accompagner efficacement la croissance de nos partenaires, nous mettons m leurs dispositions :

* Plus de 15 ans d’expertise technologique reconnue sur le marché de l’IT
* Une anticipation des exigences et une connaissance accrue de nos clients (PME et Grand Compte)
* Un accompagnement privilégié de nos clients au travers d’une offre sur-mesure

Nous distribuons des solutions IT aux revendeurs aﬁn de répondre aux besoins des clients ﬁnaux. Nous mettons m leurs dispositions des solutions performantes sur l’ensemble des préoccupations actuelles pour permettre un meilleur accompagnement dans la recherche et l’aboutissement de leur projet. Celles-ci sont réparties en 6 grandes familles de solutions :

* Sécurité IT
* Réseau Wi-ﬁ & Radio
* Stockage & Sauvegarde
* Développement
* Vidéo-protection
* Téléphonie/VOIP

### 1.2.3 Les clients principaux de SOGERTEL



Figure 1.2: CLIENTS DE SOGERTEL



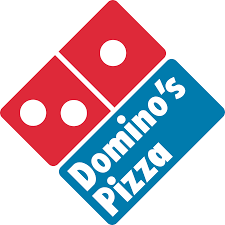
****



Figure 1.3 : CLIENTS DE SOGERTEL

## 1.3 Étude de l’Existant

Avant de commencer le développement de l'application de gestion des réclamations et des notifications, une étude de l'existant a été réalisée. Cette étape a permis d'analyser les systèmes actuels en place, leurs forces et leurs faiblesses. L'objectif était de comprendre comment les réclamations et la communication avec les clients étaient gérées jusqu'à présent, afin d'identifier les points d'amélioration. Nous avons examiné les processus manuels et les outils informatiques utilisés, ainsi que les retours d'expérience des utilisateurs finaux. Cette analyse a mis en lumière les inefficacités, telles que les délais de traitement prolongés, les erreurs de saisie et le manque de suivi des réclamations. Les résultats de cette étude ont servi de base pour définir les spécifications fonctionnelles et techniques de la nouvelle application, en s'assurant qu'elle réponde aux besoins réels des utilisateurs et qu'elle apporte des solutions aux problèmes identifiés.

## 1.4 Problématique

La problématique actuelle au sein de notre entreprise réside dans la gestion inefficace des réclamations clients et de la communication associée. Actuellement, le processus de gestion des réclamations est laborieux, sujet aux erreurs de saisie et aux retards, ce qui entraîne une insatisfaction croissante des clients. Pour résoudre ce problème, il est nécessaire de développer une application intégrée de gestion des réclamations et des notifications. Cette solution doit permettre de simplifier et d'accélérer le traitement des réclamations, tout en garantissant une communication transparente et réactive avec les clients.

Le développement de cette application pose plusieurs défis, notamment en termes de conception technique, de sécurité des données et de conformité aux normes de confidentialité. La création de cette solution interne nécessite une expertise technique avancée et une mise en place rigoureuse pour éviter les erreurs et les retards. De plus, il est crucial de s'assurer que l'application respecte les normes de sécurité et de confidentialité afin d'éviter tout risque de violation des données.

Par ailleurs, la mise en place de cette nouvelle application implique également des changements organisationnels, en termes de formation des employés, d'adaptation des processus de travail et de communication interne. L'entreprise doit donc élaborer une stratégie de gestion du changement pour assurer une intégration harmonieuse de cette nouvelle solution dans les opérations quotidiennes et maximiser son adoption par les utilisateurs finaux.

## 1.5 Objectifs Fonctionnels

1. **Gestion des réclamations clients**

* Permettre aux clients de soumettre des réclamations via une interface intuitive.
* Trier les réclamations par état (en attente, résolues).
* Mettre à jour le statut des réclamations en temps réel.
* Assurer un suivi détaillé de chaque réclamation.

1. **Notifications**

* Envoyer des notifications aux clients pour les informer de l'avancement de leurs réclamations.
* Permettre à l'administrateur d'envoyer des notifications personnalisées.

1. **Authentification sécurisée**

* Exiger une authentification pour accéder à l'application.
* Gérer les profils utilisateurs, avec la possibilité d'activer ou de désactiver des comptes clients.

1. **Statistiques et rapports**

* Générer des rapports sur le nombre de réclamations et de notifications.
* Fournir des indicateurs clés de performance pour identifier les domaines nécessitant des améliorations.

## 1.6 Objectifs Non Fonctionnels

1. **Sécurité**

* Assurer la protection des données des clients conformément aux normes de confidentialité et de sécurité.
* Mettre en place des mesures de sécurité pour prévenir les accès non autorisés.

1. **Performance**

* Garantir une réponse rapide de l'application pour toutes les opérations.
* Optimiser les temps de chargement et de traitement des réclamations.

1. **Fiabilité**

* Assurer une disponibilité maximale de l'application.
* Mettre en place des mécanismes de sauvegarde et de récupération des données.

1. **Scalabilité**

* Concevoir l'application de manière à supporter un nombre croissant d'utilisateurs et de réclamations.

1. **Convivialité**

* Développer une interface utilisateur intuitive et facile à utiliser.
* Assurer une navigation fluide et logique pour les utilisateurs.

1. **Compatibilité**

* Assurer la compatibilité de l'application avec différents navigateurs et dispositifs.

1. **Conformité**

* Respecter les réglementations en vigueur concernant la gestion des données et la protection des utilisateurs.

1. **Maintenance**

* Faciliter la maintenance et les mises à jour de l'application.
* Documenter les fonctionnalités et les processus pour une meilleure prise en main par l'équipe de maintenance.

## 1.6 Conclusion

Ce chapitre a présenté le contexte général du projet, en commençant par une introduction qui met en lumière les objectifs de l'application de gestion des réclamations et des notifications, visant à améliorer l'expérience utilisateur et l'efficacité de la communication. La présentation de l'organisme d'accueil, SOGERTEL, a permis de situer l'entreprise dans son environnement et de comprendre son domaine d'activité et ses principaux clients. L'étude de l'existant a mis en évidence les limitations des systèmes actuels et a servi de base pour définir les spécifications de la nouvelle application. La problématique a clairement défini les défis à surmonter, tandis que les objectifs fonctionnels et non fonctionnels ont fourni une feuille de route détaillée pour le développement de l'application. En conclusion, ce chapitre pose les fondations nécessaires pour aborder les phases suivantes du projet, en assurant que toutes les exigences et contraintes sont bien comprises et intégrées dans le processus de développement.

**Chapitre 2**

# Conduite Du Projet

* **Méthode Agile**
* **Planification**

## La Conduite du Projet

La conduite du projet, également connue sous les termes de gestion de projet ou management de projet, est une approche essentielle pour organiser et mener à bien un projet. Elle vise à structurer toutes les étapes nécessaires afin d’atteindre les objectifs fixés, en mettant l’accent sur quatre axes primordiaux : la qualité des livrables, le respect des délais et des coûts, ainsi que la satisfaction du client.

Ce chapitre se concentrera sur la méthodologie employée tout au long de la réalisation de mon projet. Nous décrirons en détail les phases de planification, l’ordonnancement des tâches, ainsi que d’autres aspects de la gestion du projet. Pour mettre en œuvre mon système, on a utilisé la méthode agile Scrum, qui a joué un rôle déterminant dans le bon déroulement des différentes étapes de ce projet.

## 2.1 Processus de développement

### 2.1.1 Méthode Agile

La méthode agile est une approche de gestion de projet et de développement logiciel qui privilégie la flexibilité, la collaboration et la réactivité aux changements. Contrairement aux méthodes traditionnelles en cascade, l'agile se concentre sur des cycles de développement courts et itératifs, appelés itérations ou sprints. Chaque itération aboutit à un produit livrable ou une fonctionnalité utilisable, permettant des ajustements rapides en fonction des retours des utilisateurs et des parties prenantes. L'agile favorise la communication directe et fréquente entre les membres de l'équipe et avec les clients, assurant une meilleure compréhension des besoins et une adaptation rapide aux exigences évolutives, cette méthode met l'accent sur l'amélioration continue, la satisfaction du client et la livraison rapide de produits de haute qualité.

### 2.1.2 Méthode Scrum

L'utilisation de la méthode Scrum pour notre projet de gestion des réclamations et des notifications nous permet de bénéficier de plusieurs avantages cruciaux.Premièrement, la division du projet en sprints facilite une planification détaillée et itérative, ce qui nous aide à gérer efficacement les délais et à ajuster les priorités en fonction des retours et des exigences changeantes.Deuxièmement, les réunions régulières de revue et de rétrospective encouragent une communication ouverte et continue, permettant de détecter et de résoudre rapidement les problèmes.Enfin, l'approche centrée sur l'amélioration continue assure une qualité optimale des livrables et renforce la satisfaction des utilisateurs finaux.

En somme, Scrum nous aide à maintenir une organisation rigoureuse tout en étant flexibles et réactifs face aux imprévus et aux évolutions du projet.

#### 2.1.2.1 Pôle de compétences

Chaque sprint a un objectif spécifique et est associé à une liste d'éléments de backlog produit (fonctionnalités) à réaliser. L'équipe décompose ces éléments en tâches élémentaires de quelques heures, appelées items de backlog sprint. Pendant un sprint, les items de backlog sprint ne peuvent pas être modifiés. Les éventuels changements sont pris en compte dans le backlog produit et seront réalisés dans les sprints ultérieurs.

﻿Si l'équipe constate en cours de sprint qu'elle ne pourra pas terminer un item du backlog sprint, ou au contraire, qu'elle aura fini plus tôt que prévu, il est possible, avec l'accord de l'équipe et du directeur du produit, de retirer ou d'ajouter un item à ce qui était initialement prévu. Au quotidien, une réunion de suivi permet à l'équipe et au Scrum Master (chef de projet) de faire le point sur l'avancement des tâches et de discuter des difficultés rencontrées.

Une réunion de planification, qui ne doit pas dépasser 4 heures, a lieu au début de chaque sprint. Lors des réunions de suivi, on répond à trois questions :

• Qu'est-ce que j'ai fait ?

• Qu'est-ce que je compte faire ?

• Quelles sont les difficultés que je rencontre ?

À la fin de chaque sprint, une réunion de bilan du sprint est faite. Elle consiste à générer un rapport sur les fonctionnalités qui étaient demandés et une démonstration du produit, qui sera validé par le chef de produit.

#### 2.1.2.2 Rôle et rituels

Scrum est considéré comme un cadre ou "Framework" de gestion du projet. Ce cadre est constitué d'une définition des rôles, de réunions et d'artefacts. Pour les rôles dans Scrum, ils sont au nombre de trois :

* **Le Product Owner** qui porte la vision du produit à réaliser et travaille en interaction avec l'équipe de développement. Il s'agit généralement d'un expert du domaine métier du projet. Il est responsable de la partie fonctionnelle, il est responsable du QUOI" faire ? Quel est le logiciel à développer, avec quelles fonctionnalités ? Afin de bien comprendre et transmettre, communiquer le "QUOI", le Product Owner doit bien maîtriser le "POURQUOI", la raison pour laquelle on doit faire chaque chose.
* **Le Scrum Master** qui doit maîtriser Scrum et s'assurer que ce dernier est correctement appliqué. Il joue donc un rôle de coach à la fois auprès du Product Owner et auprès de l'équipe de développement. Il doit donc faire preuve de pédagogie. Il est également chargé de s'assurer que l'équipe de développement est pleinement productive. Généralement le candidat tout trouvé au rôle de Scrum Master est le chef de projet. Celui-ci devra cependant renoncer au style de management "commander et contrôler" pour adopter un mode de management participatif.
* **L'Equipe de Développement** produisent le logiciel et sont responsables de tout ce qui concerne le code : les choix techniques, la qualité du code, leurs apprentissages et leur manière de s'organiser pour délivrer à la fin d'une itération les fonctionnalités attendues. La méthode fonctionne par itérations courtes pour délivrer souvent. Les développeurs s'occupent du "COMMENT" faire.

L'ensemble d'événements séquentiels, de cérémonies ou de réunions que les équipes Scrum effectuent régulièrement constituent des composantes mieux connues du framework.

* **Organisation du backlog :** Aussi connue sous le nom de préparation du backlog, cette étape est de la responsabilité du Product Owner. Il a deux tâches principales : concrétiser la vision du produit et rester constamment aligné avec le marché et les clients.
* **Planification du Sprint (Sprint = itération) :** Lors de cette réunion, l'équipe de développement sélectionne les éléments prioritaires du "Product Backlog" (une liste ordonnancée des exigences fonctionnelles et non fonctionnelles du projet) qu'elle estime pouvoir réaliser pendant le sprint, en collaboration avec le Product Owner.
* **Sprint :** Un sprint désigne la durée réelle dont l'équipe Scrum a besoin pour terminer un incrément. La durée classique d'un sprint est de deux semaines, mais certaines équipes préfèrent une semaine ou un mois pour livrer un incrément de valeur plus facilement.
* **Mêlée quotidienne ou Daily Scrum :** Une mini-réunion quotidienne qui se tient à la même heure (généralement le matin) et au même endroit pour faciliter les échanges. Bien que la durée de 15 minutes soit couramment visée, ce n'est qu'une indication.
* **Revue de Sprint :** Lors de cette réunion qui a lieu à la fin du sprint, l'équipe de développement présente les fonctionnalités terminées pendant le sprint et recueille les retours du Product Owner et des utilisateurs finaux. C'est également le moment d'anticiper le périmètre des prochains sprints et d'ajuster si nécessaire la planification de la release (nombre de sprints restants).
* **Rétrospective de Sprint :** La rétrospective, qui a généralement lieu après la revue de sprint, offre l'opportunité de s'améliorer (productivité, qualité, efficacité, conditions de travail, etc.) en se basant sur l'expérience du sprint écoulé, selon le principe de l'amélioration continue.

#### 2.1.2.3 Artefacts Scrum

Les artefacts sont des objets que nous créons, tels que des outils pour résoudre des problèmes. Dans Scrum, nous avons trois artefacts essentiels : le backlog produit, le backlog de sprint et l'incrément qui représente les tâches "terminées". Ce sont les trois éléments constants autour desquels une équipe Scrum continue de réfléchir et d'investir pendant les heures supplémentaires.

**• Le backlog produit :** C'est la liste principale des tâches à réaliser. Géré par le Product Owner ou le responsable produit, il s'agit d'une liste dynamique de fonctionnalités, d'exigences, d'améliorations et de correctifs, servant de point de départ pour le backlog de sprint. En résumé, c'est la "to-do list" de l'équipe.

**• Le backlog de sprint :** Il s'agit de la liste d'éléments, d'histoires utilisateurs ou de correctifs de bug que l'équipe de développement a sélectionnés pour les implémenter dans le cycle de sprint en cours.

**• L'incrément (ou l'objectif du sprint) :** C'est le produit final exploitable obtenu pendant le sprint. Il représente l'ensemble des éléments de travail terminés pendant le sprint, prêt à être livré et utilisé.

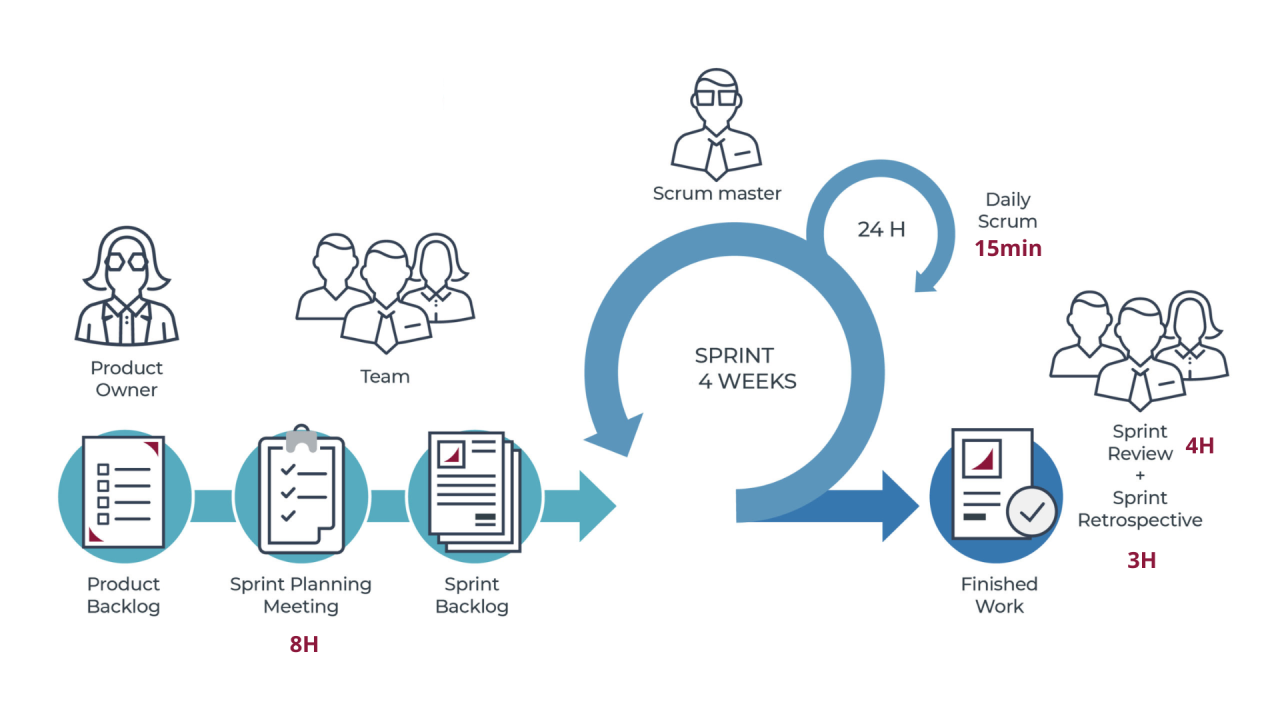


Figure 2.1 : ARCHITECTURE DE LA METHODE SCRUM

#### 2.1.2.4 Diagramme de Gantt

**﻿**Le diagramme de Gantt, largement utilisé en gestion du projet, est un outil extrêmement efficace pour représenter visuellement l'état d'avancement des activités qui composent un projet. Il offre une vue d'ensemble en un coup d'œil, permettant de :

• Identifier les différentes tâches à prendre en compte

• Visualiser les dates de début et de fin de chaque tâche

• Estimer la durée prévue pour chaque tâche

• Repérer les éventuels chevauchements de tâches et leur durée.

• Déterminer les dates de début et de fin du projet dans son ensemble.

Grâce à sa représentation graphique claire, le diagramme de Gantt facilite la planification, la coordination et le suivi des projets, offrant ainsi un moyen pratique de gestion pour les équipes et les parties prenantes. Son utilisation permet de visualiser rapidement l'état d'avancement global du projet et de prendre des décisions éclairées pour optimiser la progression.

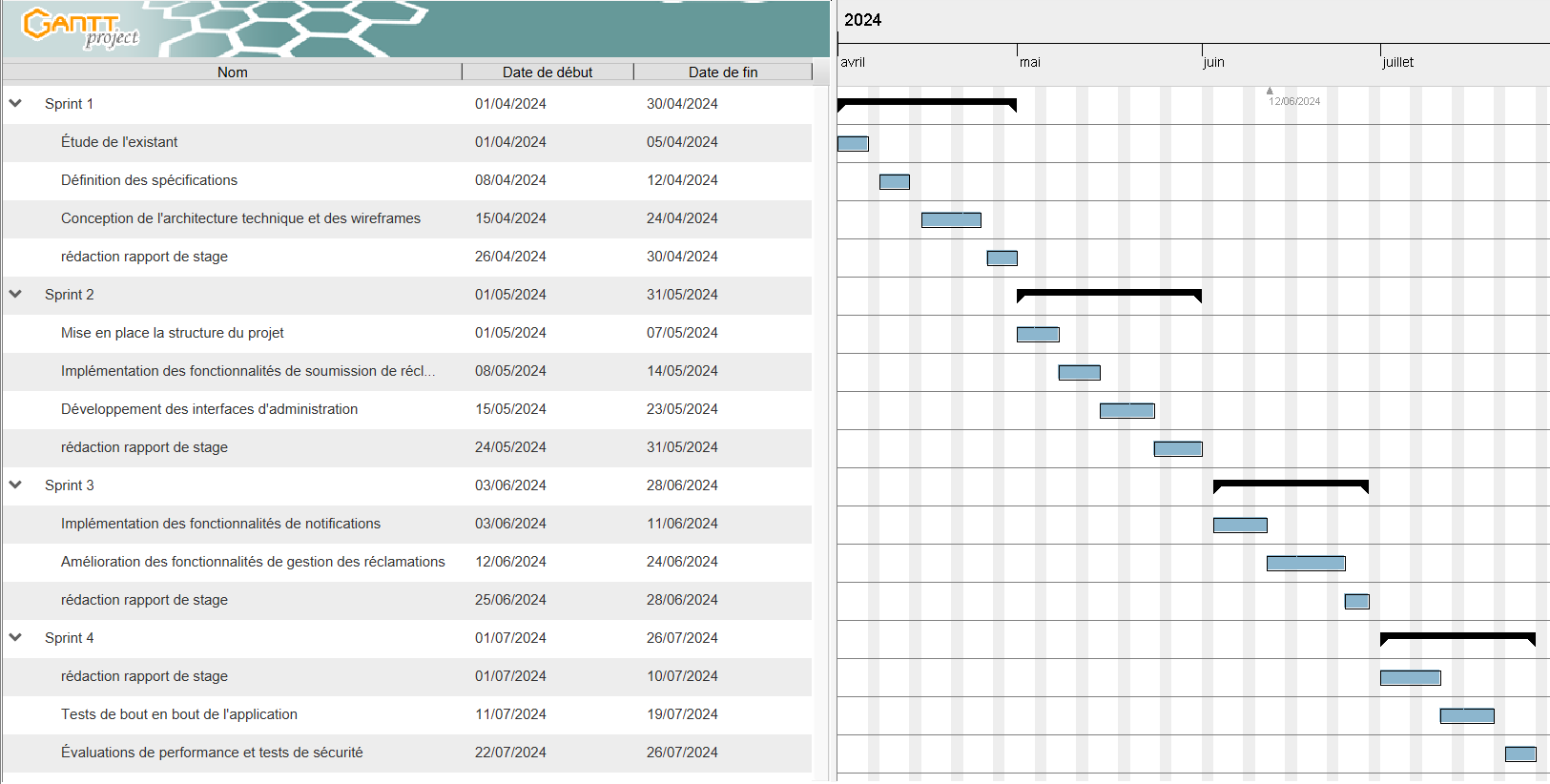


Figure 2.2: DIAGRAMME DE GANTT

## 2.2 Planification du projet

La planification constitue l'une des phases préliminaires les plus cruciales d'un projet, offrant un outil essentiel pour sa gestion. Elle implique l'identification et l'ordonnancement des tâches du projet, l'estimation de leurs charges respectives, l'établissement d'objectifs, la coordination des actions, la maîtrise des ressources, la réduction des risques, le suivi des actions en cours et le rapport sur l'état d'avancement global du projet. Un sprint correspond à une période définie pendant laquelle les équipes accomplissent les tâches de leur backlog produit. La durée des sprints est généralement d'une, deux ou quatre semaines. À la fin d'un sprint, une équipe aura normalement réalisé une version fonctionnelle du produit.

### 2.2.1 Sprint 1 : Analyse et Conception

Le premier sprint de notre projet a été consacré à l'analyse et à la conception de l'application de gestion des réclamations et des notifications. Nous avons commencé par réaliser une étude approfondie de l'existant pour comprendre les processus actuels et identifier les inefficacités. Cette phase a impliqué des entretiens avec les parties prenantes, y compris les administrateurs et les clients, pour recueillir leurs besoins et leurs attentes. Ensuite, nous avons défini les spécifications fonctionnelles et non fonctionnelles de l'application. Cela a permis de créer des user stories détaillées, qui ont été priorisées en fonction de leur importance et de leur complexité. Nous avons également élaboré une première version de l'architecture technique, en choisissant les technologies et les Framework les plus adaptés. La conception de l'interface utilisateur a été esquissée sous forme de wireframes pour valider l'ergonomie et l'expérience utilisateur.

### 2.2.2 Sprint 2 : Développement de la gestion des Réclamations

Le deuxième sprint a été consacré au développement des fonctionnalités de base de la gestion des réclamations. L'équipe de développement a commencé par mettre en place l'infrastructure du projet, en configurant les environnements de développement, de test et de production. Ensuite, nous avons implémenté les premières fonctionnalités permettant aux clients de soumettre des réclamations et aux administrateurs de les recevoir et de les gérer. Cela a inclus la création des formulaires de soumission de réclamations, la mise en place de la base de données pour stocker les réclamations et le développement des interfaces d'administration pour trier et consulter les réclamations. Nous avons également développé des API pour permettre une communication fluide entre le front-end et le back-end. Des tests unitaires et d'intégration ont été réalisés pour s'assurer de la fiabilité du code. À la fin du sprint, nous avons organisé une démonstration des fonctionnalités développées aux parties prenantes pour recueillir leurs retours et ajuster notre travail en conséquence.

### 2.2.3 Sprint 3 : Développement de la gestion des Notifications et améliorations

Le troisième sprint a porté sur le développement des fonctionnalités de base de la gestion des notifications. L'équipe de développement a commencé par mettre en place l'infrastructure nécessaire pour le système de notifications, en configurant les environnements de développement, de test et de production. Ensuite, nous avons implémenté les premières fonctionnalités permettant aux clients de recevoir des notifications concernant l'avancement de leurs réclamations. Cela a inclus la création des templates de notification, la mise en place de la base de données pour stocker les informations de notification et le développement des interfaces permettant aux administrateurs de configurer et gérer les notifications. Nous avons également développé des API pour permettre une communication fluide entre le système de gestion des réclamations et le système de notifications. Des tests unitaires et d'intégration ont été réalisés pour s'assurer de la fiabilité du code. À la fin du sprint, nous avons organisé une démonstration des fonctionnalités développées aux parties prenantes pour recueillir leurs retours et ajuster notre travail en conséquence.

### 2.2.4 Sprint 4 : Tests et Documentation Technique

Le quatrième et dernier sprint a été consacré principalement aux tests exhaustifs et à la création de la documentation technique correspondante pour garantir une utilisation fluide et efficace du système. Nous avons entrepris des tests de bout en bout pour vérifier l'intégrité et la cohérence de l'application, en testant rigoureusement chaque fonctionnalité, des cas d'utilisation basiques aux scénarios avancés. Ces tests incluaient des évaluations de performance et des tests de sécurité pour corriger toute vulnérabilité potentielle. Parallèlement, une documentation technique complète a été rédigée, couvrant tous les aspects de l'application, de l'architecture système aux détails des API et des configurations de déploiement. Des guides détaillés ont été créés pour les administrateurs, ainsi que des manuels d'utilisation pour les utilisateurs finaux, afin de les guider dans la soumission des réclamations et la consultation des notifications. Des procédures de maintenance future, incluant la mise à jour, la sauvegarde et la restauration des données, ont également été documentées. Ce sprint s'est conclu par une revue finale avec toutes les parties prenantes, où les résultats des tests et la documentation ont été présentés et validés, confirmant que l'application était prête pour le déploiement et l'utilisation en production.

## 2.3 Conclusion

Après avoir exposé une vue d’ensemble de la méthodologie utilisée lors de la réalisation de mon projet, en particulier la méthode agile Scrum et la planification des itérations pour sa mise en œuvre, il est maintenant temps de se pencher sur l’architecture du projet, en vue de concrétiser la solution.

**Chapitre 3**

# Analyse et Conception

* **Analyse Fonctionnelle**
* **Analyse Dynamique**
* **Analyse Objet**

﻿

## 3.1 Introduction

Dans ce chapitre, nous allons aborder l'analyse et la conception du système. Nous allons commencer par une analyse des besoins et des exigences pour comprendre les fonctionnalités attendues du système. Ensuite, nous allons présenter la conception du système en utilisant des diagrammes UML pour modéliser les différents aspects du système. Cette étape est cruciale pour garantir que le système répond aux besoins et aux exigences identifiés lors de l'analyse.

## 3.2 Le formalisme UML

UML, (Unified Modeling Language), que l'on peut traduire par "langage de modélisation unifié" est une notation permettant la modélisation d'un problème. Ce langage est né de la fusion de plusieurs méthodes existantes auparavant, et est devenu la référence en termes de modélisation objet. Entre 1910 et 1990, de nombreux analystes ont mis au point des approches orientées objets, ainsi en 1994 il existait plus de 50 méthodes objet. Toutefois seules 3 méthodes ont véritablement émergé :

* La méthode OMT de Rumbaugh.
* La méthode BOOCH'93 de Booch.
* La méthode OOSE de Jacobson (Object Oriented Software Engineering).

A partir de 1994. Rumbaugh et Booch (rejoints en 1995 par Jacobson) ont uni leurs efforts pour mettre au point la méthode unifiée incorporant les avantages de chacune des méthodes précédentes. La méthode unifiée à partir de la version 1.0 devient UML (Unified Modeling Language), une notation universelle pour la modélisation objet. UML 1.0 est soumise à l'OMG1 en janvier 1997 mais elle ne sera acceptée qu'en novembre 1997 dans sa version 1.1, date à partir de laquelle UML devient un standard international cette méthode représente un moyen de spécifier, représenter et construire les composantes d'un système informatique. En effet, la notation unifiée définit 9 diagrammes pour représenter les différents points de vue de modélisation. Ces diagrammes permettent de visualiser et de manipuler les éléments de modélisation. Les diagrammes définis par UML, sont les suivants :

• **Les diagrammes de cas d’utilisation** **:** représentation des fonctions du système du point de vue de l'utilisateur.

• **Les diagrammes de séquence :** représentation temporelle des objets et de leurs Les diagrammes d'activités représentation du comportement d'une opération en termes d'actions.

• **Les diagrammes de classes :** représentation de la structure statique en termes de classes et de relations.

### ﻿**3.2.1 Le choix d'UML**

Une modélisation objet apparait la plus adaptée. En effet, l'objet a fait ses preuves dans la réalisation d'applications temps réel. C'est pourquoi j'ai opté pour UML comme langage de modélisation. Ce choix peut être justifié également par plusieurs raisons. La notation UML facilite la compréhension et la communication d'une modélisation objet. Durant mon étude du système, j'ai utilisé trois diagrammes d'UML, il s'agit du diagramme des cas d'utilisation, les diagrammes de séquences, le diagramme d'activité, et le diagramme de classes.

## 3.3 Démarche conception

La conception est la plus importante étape du cycle du développement d'application. Elle se base essentiellement sur la bonne spécification et l'analyse des besoins. A présent, nous sommes dans la phase de concevoir la solution, et définir les composants globaux de notre projet.

### 3.3.1 Analyse fonctionnelle

L'objectif de l'analyse fonctionnelle de l'application de gestion des réclamations et des notifications est de comprendre comment intégrer ce nouveau système dans l'infrastructure existante de l'entreprise pour répondre aux besoins de gestion et de communication avec les clients. Cette analyse doit examiner les processus actuels de traitement des réclamations et d'envoi de notifications, identifier les points d'amélioration et proposer des solutions pour automatiser et optimiser ces tâches.

Plus spécifiquement, il s'agit de déterminer les fonctionnalités essentielles que l'application doit offrir, telles que la soumission de réclamations par les clients, la gestion des réclamations par les administrateurs, l'envoi de notifications automatisées, et la gestion des informations clients. L'analyse fonctionnelle doit également prendre en compte les contraintes techniques et organisationnelles, telles que la nécessité d'une interface utilisateur conviviale et la capacité à intégrer le système dans les flux de travail existants sans perturber les opérations courantes.

En résumé, l'analyse fonctionnelle de l'application de gestion des réclamations et des notifications vise à définir les services que ce système doit offrir à ses utilisateurs, tout en proposant des solutions pour intégrer ce nouveau système dans l'infrastructure existante de l'entreprise. Cela permettra d'améliorer la gestion des réclamations, de réduire les erreurs et les délais de traitement, et d'assurer une communication efficace et transparente avec les clients.

**Les acteurs :**

* **Administrateur :** L'administrateur est chargé de superviser le bon fonctionnement de l'application. Ses responsabilités comprennent la gestion des réclamations, l'envoi de notifications aux clients, la gestion des comptes clients. Pour accéder à ces fonctionnalités, l'administrateur doit s'authentifier à l'aide d'un identifiant et d'un mot de passe.
* **Client :** Le client est l'utilisateur principal de l'application. Il a la possibilité de soumettre des réclamations, de consulter l'état de ses réclamations, de recevoir des notifications, de mettre à jour son profil. La connexion est obligatoire pour pouvoir soumettre une réclamation. Il convient de noter que le visiteur du site et le client sont considérés comme un unique acteur.

Le diagramme ci-dessous, représente la description globale du fonctionnement de mon projet (Admin) :

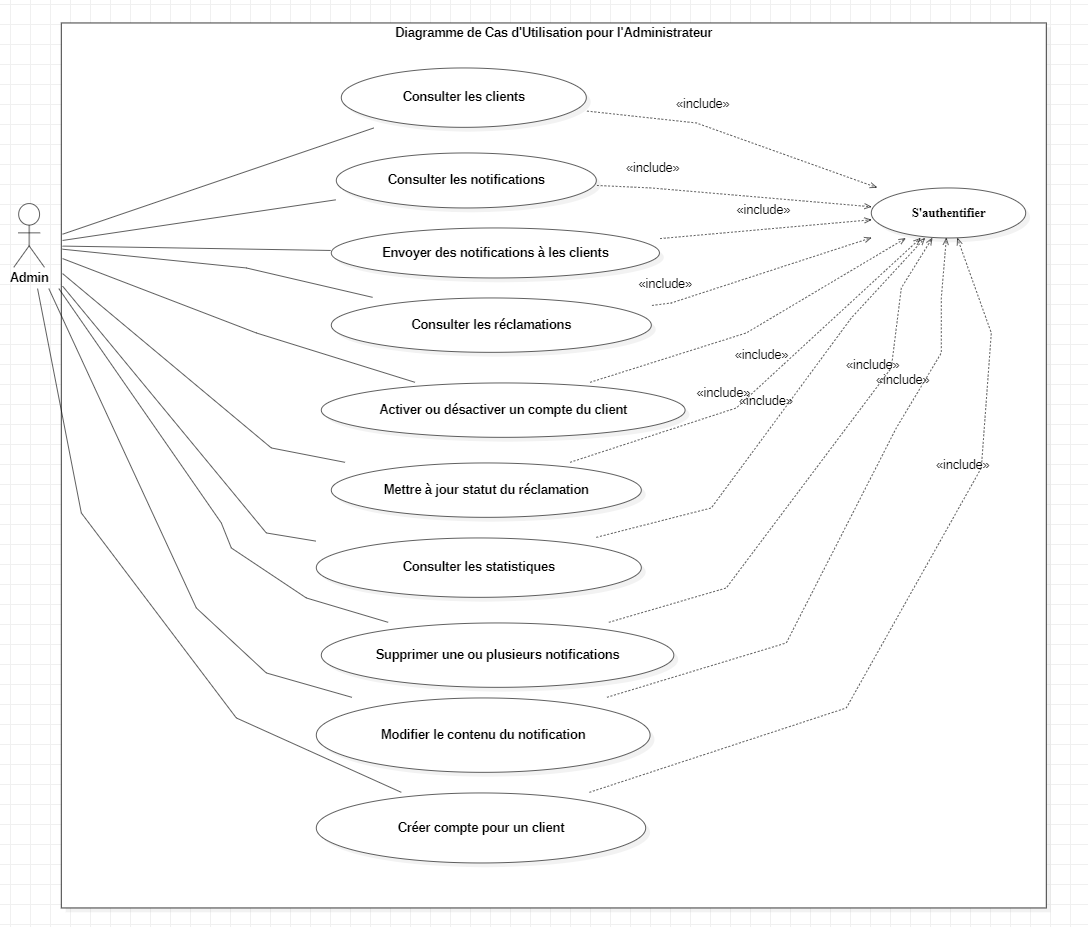


Figure 3.1 : DIAGRAMME CAS D’UTILISATION

**Table 3.1 : Cas d'utilisation : Consulter les clients**

|  |
| --- |
| **Cas d’utilisation :** Consulter les clients |
| **But :** Permettre à l'administrateur de visualiser la liste des clients |
| **Acteur système :** Administrateur |
| **Pré-condition :** L'administrateur doit être authentifié |
| **Post-condition :** Les informations des clients sont affichées. |
| **Scénario principale :** |
| * L’administrateur sélectionne "Consulter les clients" après s'être authentifié. * Le système affiche la liste des clients. * L’administrateur sélectionne un client pour voir plus de détails. * Le système affiche les détails du client sélectionné. |
| **Scénario alternatif :** Si la base de données est inaccessible, afficher un message d'erreur |

**Table 3.2 : Cas d'utilisation : Consulter les notifications**

|  |
| --- |
| **Cas d’utilisation :** Consulter les notifications |
| **But :** Permettre à l'administrateur de visualiser les notifications |
| **Acteur système :** Administrateur |
| **Pré-condition :** L'administrateur doit être authentifié |
| **Post-condition :** Les notifications sont affichées |
| **Scénario principale :** |
| * L'administrateur sélectionne "Consulter les notifications" après s'être authentifié. * Le système affiche la liste des notifications. * L'administrateur sélectionne une notification pour voir plus de détails. * Le système affiche les détails de la notification sélectionnée. |
| **Scénario alternatif :** Si la base de données est inaccessible, afficher un message d'erreur |

**Table 3.3 : Cas d'utilisation : Envoyer des notifications aux clients**

|  |
| --- |
| **Cas d’utilisation :** Envoyer des notifications aux clients |
| **But :** Permettre à l'administrateur d'envoyer des notifications aux clients |
| **Acteur système :** Administrateur |
| **Pré-condition :** L'administrateur doit être authentifié |
| **Post-condition :** Les notifications sont envoyées aux clients |
| **Scénario principale :** |
| * L'administrateur sélectionne "Envoyer Notification" après s'être authentifié. * Le système affiche un formulaire pour créer une notification. * L'administrateur remplit les détails de la notification et sélectionne le client ciblé. * L'administrateur soumet la notification. * Le système envoie la notification au client sélectionné. |
| **Scénario alternatif :** |
| * Si le système détecte une erreur dans les détails de la notification, il affiche un message d'erreur et demande à l'administrateur de corriger les informations avant de soumettre à nouveau. |

**Table 3.4 : Cas d'utilisation : Consulter les réclamations**

|  |
| --- |
| **Cas d’utilisation :** Consulter les réclamations |
| **But :** Permettre à l'administrateur de consulter et gérer les réclamations des clients |
| **Acteur système :** Administrateur |
| **Pré-condition :** L'administrateur doit être authentifié |
| **Post-condition :** Les réclamations sont affichées et leur état peut être mis à jour |
| **Scénario principale :** |
| * L'administrateur sélectionne "Consulter les réclamations" après s'être authentifié. * Le système affiche la liste des réclamations. * L'administrateur sélectionne une réclamation pour voir plus de détails. * Le système affiche les détails de la réclamation sélectionnée. * L'administrateur peut mettre à jour l'état de la réclamation. |
| **Scénario alternatif :** Si la base de données est inaccessible, afficher un message d'erreur |

**Table 3.5 : Cas d'utilisation : Activer ou désactiver un compte de client**

|  |
| --- |
| **Cas d’utilisation :** Activer ou désactiver un compte de client |
| **But :** Permettre à l'administrateur d'activer ou désactiver les comptes clients |
| **Acteur système :** Administrateur |
| **Pré-condition :** L'administrateur doit être authentifié |
| **Post-condition :** Le compte client est activé ou désactivé |
| **Scénario principale :** |
| * L'administrateur sélectionne "Activer ou désactiver un compte de client" après s'être authentifié. * Le système affiche la liste des clients. * L'administrateur sélectionne un client et choisit d'activer ou de désactiver le compte. * Le système met à jour le statut du compte du client |
| **Scénario alternatif :** |
| * Si le système rencontre une erreur lors de la mise à jour du statut du compte, il affiche un message d'erreur et demande à l'administrateur de réessayer. |

**Table 3.6 : Cas d'utilisation : Mettre à jour le statut de la réclamation**

|  |
| --- |
| **Cas d’utilisation :** Mettre à jour le statut la réclamation |
| **But :** Permettre à l'administrateur de mettre à jour l'état des réclamations |
| **Acteur système :** Administrateur |
| **Pré-condition :** L'administrateur doit être authentifié |
| **Post-condition :** Le statut de la réclamation est mis à jour |
| **Scénario principale :** |
| * L'administrateur sélectionne "Mettre à jour la réclamation" après s'être authentifié * Le système affiche la liste des réclamations * L'administrateur sélectionne une réclamation et modifie l'état * L'administrateur soumet les modifications * Le système enregistre les modifications |
| **Scénario alternatif :** |
| * Si le système détecte une erreur lors de la soumission des modifications, il affiche un message d'erreur et demande à l'administrateur de corriger les informations avant de soumettre à nouveau. |

**Table 3.7 : Cas d'utilisation : Consulter les statistiques**

|  |
| --- |
| **Cas d’utilisation :** Consulter les statistiques |
| **But :** Permettre à l'administrateur de consulter les statistiques des réclamations et des notifications |
| **Acteur système :** Administrateur |
| **Pré-condition :** L'administrateur doit être authentifié |
| **Post-condition :** Les statistiques sont affichées |
| **Scénario principale :** |
| * L'administrateur sélectionne "Consulter les statistiques" après s'être authentifié. * Le système affiche un tableau de bord avec diverses statistiques (nombre de réclamations, état des réclamations, etc.). * L'administrateur peut filtrer et visualiser les statistiques selon différents critères. |
| **Scénario alternatif :** Si la base de données est inaccessible, afficher un message d'erreur |

**Table 3.8 : Cas d'utilisation : Modifier le contenu d'une notification**

|  |
| --- |
| **Cas d’utilisation :** Modifier le contenu d'une notification |
| **But :** Permettre à l'administrateur de modifier le contenu d'une notification |
| **Acteur système :** Administrateur |
| **Pré-condition :** L'administrateur doit être authentifié |
| **Post-condition :** La notification est modifié |
| **Scénario principale :** |
| * L'administrateur sélectionne la notification à mettre à jour * L'application envoie une requête à la base de données * La base de données met à jour la notification * L'application confirme la mise à jour à l'administrateur |
| **Scénario alternatif :** Si la base de données est inaccessible, afficher un message d'erreur |

**Table 3.9 : Cas d'utilisation : Supprimer une ou plusieurs notifications**

|  |
| --- |
| **Cas d’utilisation :** Supprimer une ou plusieurs notifications |
| **But :** Permettre à l'administrateur de supprimer des notifications |
| **Acteur système :** Administrateur |
| **Pré-condition :** L'administrateur doit être authentifié |
| **Post-condition :** Les notifications sont supprimées |
| **Scénario principale :** |
| * L'administrateur sélectionne les notifications à supprimer * L'application envoie une requête à la base de données * La base de données supprime les notifications * L'application confirme la suppression à l'administrateur |
| **Scénario alternatif :** Si la base de données est inaccessible, afficher un message d'erreur |

**Table 3.10 : Cas d'utilisation : Créer Compte pour un Client**

|  |
| --- |
| **Cas d’utilisation :** Créer compte pour un client |
| **But :** Permettre à l'administrateur de créer un nouveau compte pour un client |
| **Acteur système :** Administrateur |
| **Pré-condition :** L'administrateur doit être authentifié |
| **Post-condition :** Un nouveau compte client est créé |
| **Scénario principale :** |
| * L'administrateur entre les informations du nouveau client * L'application envoie une requête à la base de données * L'application confirme la création du compte à l'administrateur * La base de données crée un nouveau compte client |
| **Scénario alternatif :** Si la base de données est inaccessible, afficher un message d'erreur |

Le diagramme ci-dessous, représente la description globale du fonctionnement de mon projet (Client) :

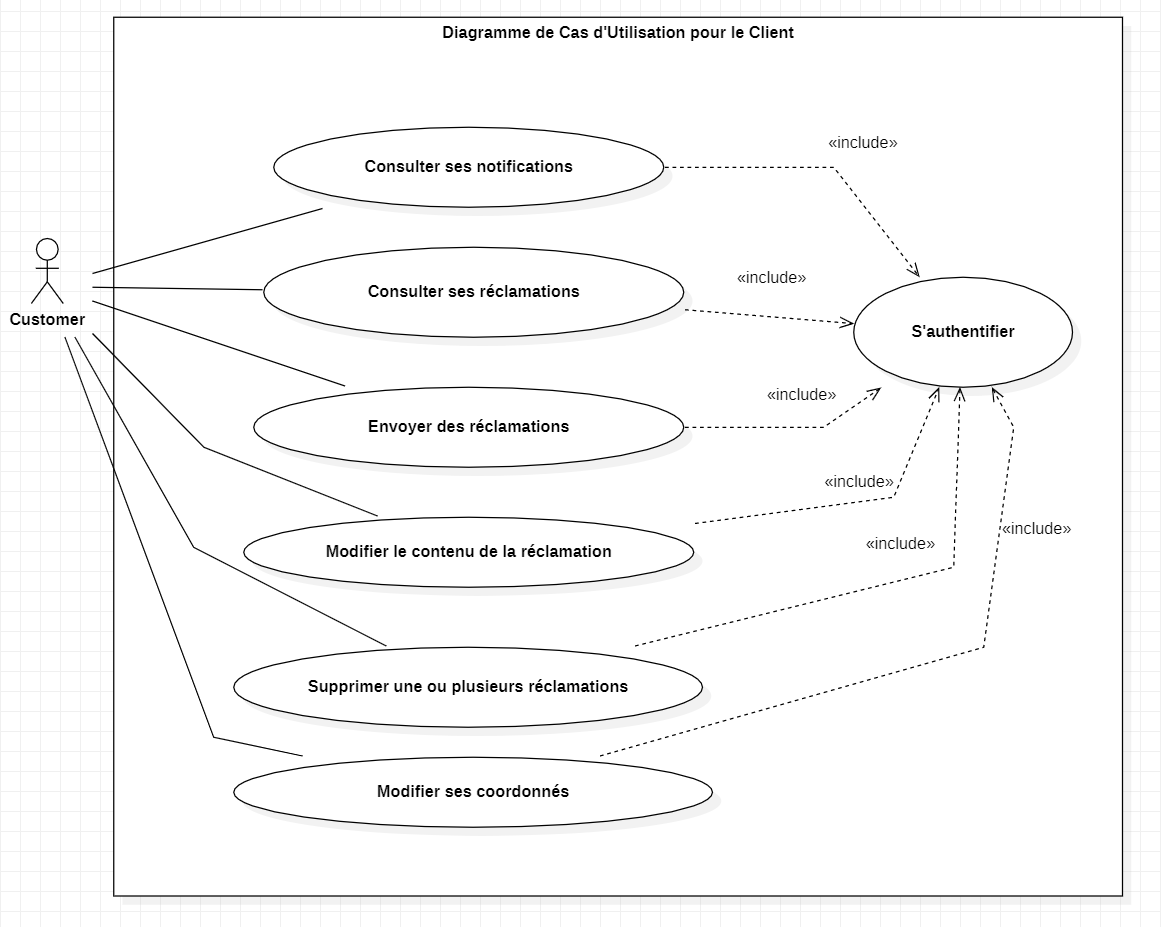


Figure 3.2 : DIAGRAMME CAS D’UTILISATION

**Table 4.1 : Cas d'utilisation : Consulter ses notifications**

|  |
| --- |
| **Cas d’utilisation :** Consulter ses notifications |
| **But :** Permettre au client de visualiser les notifications reçues concernant ses réclamations et d'autres informations pertinentes |
| **Acteur système :** Client |
| **Pré-condition :** Le client doit être authentifié |
| **Post-condition :** Les notifications sont affichées à l'écran pour le client |
| **Scénario principale :** |
| * Le client s'authentifie. * Le client sélectionne l'option pour consulter les notifications. * Le système affiche la liste des notifications récentes. * Le client peut lire le contenu des notifications |
| **Scénario alternatif :** |
| * Si aucune notification n'est disponible, le système informe le client qu'il n'y a pas de nouvelles notifications |

**Table 4.2 : Cas d'utilisation : Consulter ses réclamations**

|  |
| --- |
| **Cas d’utilisation :** Consulter ses réclamations |
| **But :** Permettre au client de voir la liste de ses réclamations, y compris leur statut et les détails associés |
| **Acteur système :** Client |
| **Pré-condition :** Le client doit être authentifié |
| **Post-condition :** Les réclamations du client sont affichées à l'écran |
| **Scénario principale :** |
| * Le client s'authentifie. * Le client sélectionne l'option pour consulter ses réclamations. * Le système affiche la liste des réclamations du client. * Le client peut visualiser les détails de chaque réclamation |
| **Scénario alternatif :** |
| * Si aucune réclamation n'existe, le système informe le client qu'il n'y a pas de réclamations à afficher |

**Table 4.3 : Cas d'utilisation : Envoyer des réclamations**

|  |
| --- |
| **Cas d’utilisation :** Envoyer des réclamations |
| **But :** Permettre au client de soumettre une nouvelle réclamation via l'application |
| **Acteur système :** Client |
| **Pré-condition :** Le client doit être authentifié |
| **Post-condition :** La réclamation est soumise et enregistrée dans le système |
| **Scénario principale :** |
| * Le client s'authentifie. * Le client sélectionne l'option pour envoyer une réclamation. * Le client remplit le formulaire de réclamation avec les détails nécessaires. * Le client soumet la réclamation. * Le système enregistre la réclamation et envoie une confirmation au client. |
| **Scénario alternatif :** |
| * Si les informations fournies sont incomplètes, le système demande au client de compléter les champs obligatoires avant de soumettre |

**Table 4.4 : Cas d'utilisation : Modifier le contenu de la réclamation**

|  |
| --- |
| **Cas d’utilisation :** Modifier le contenu de la réclamation |
| **But :** Permettre au client de modifier les détails d'une réclamation existante |
| **Acteur système :** Client |
| **Pré-condition :** Le client doit être authentifié et avoir au moins une réclamation existante |
| **Post-condition :** La réclamation modifiée est enregistrée dans le système |
| **Scénario principale :** |
| * Le client s'authentifie * Le client sélectionne une réclamation existante à modifier * Le client met à jour les informations de la réclamation * Le client soumet les modifications * Le système enregistre les modifications et envoie une confirmation au client |
| **Scénario alternatif :** |
| * Si les modifications ne sont pas valides (par exemple, champs manquants), le système informe le client et lui demande de corriger les erreurs |

**Table 4.5 : Cas d'utilisation : Supprimer une ou plusieurs réclamations**

|  |
| --- |
| **Cas d’utilisation :** Supprimer une ou plusieurs réclamations |
| **But :** Permettre au client de supprimer une ou plusieurs réclamations existantes |
| **Acteur système :** Client |
| **Pré-condition :** Le client doit être authentifié et avoir au moins une réclamation existante |
| **Post-condition :** Les réclamations sélectionnées sont supprimées du système. |
| **Scénario principale :** |
| * Le client s'authentifie. * Le client sélectionne les réclamations à supprimer. * Le client confirme la suppression. * Le système supprime les réclamations sélectionnées et envoie une confirmation au client |
| **Scénario alternatif :** |
| * Si une réclamation sélectionnée pour suppression n'existe plus ou ne peut être supprimée (par exemple, en cours de traitement critique), le système informe le client de l'erreur et propose une solution alternative |

**Table 4.6 : Cas d'utilisation : Modifier ses coordonnées**

|  |
| --- |
| **Cas d’utilisation :** Modifier ses coordonnées |
| **But :** Permettre au client de mettre à jour ses informations de contact |
| **Acteur système :** Client |
| **Pré-condition :** Le client doit être authentifié |
| **Post-condition :** Les coordonnées du client sont mises à jour dans le système |
| **Scénario principale :** |
| * Le client s'authentifie. * Le client accède à la section de modification des coordonnées * Le client saisit les nouvelles coordonnées * Le système vérifie la validité des nouvelles coordonnées * Le système enregistre les nouvelles coordonnées dans la base de données * Le système confirme la mise à jour des coordonnées au client |
| **Scénario alternatif :** |
| * Si les nouvelles coordonnées sont invalides, le système affiche un message d'erreur et demande au client de corriger les informations |

### 3.3.2 Analyse fonctionnelle

**﻿Un cas d'utilisation décrit un ensemble de scénarios. Un scénario décrit une exécution particulière d'un cas d'utilisation du début jusqu'à la fin. Il correspond à une sélection d'enchaînements du cas d'utilisation. Il est clair que la combinatoire des enchaînements fait exploser le nombre de scénarios potentiels, je ne pourrai pas donc tous les décrire. Alors j'ai réalisé les diagrammes suivants**

#### ****3.3.2.1 Diagramme de séquence Authentification****

**Le diagramme de séquence ci-joint illustre le processus d'authentification d'un utilisateur sur une plateforme. Le scénario commence avec l'utilisateur entrant son nom d'utilisateur et son mot de passe sur la page de connexion. Ces informations sont ensuite envoyées au serveur, qui vérifie les informations de l'utilisateur en consultant la base de données. Si les informations sont correctes, le serveur retourne une réponse positive et l'utilisateur est redirigé vers la page du tableau de bord. Si les informations sont incorrectes, le serveur renvoie une réponse négative et l'utilisateur est informé que les informations fournies sont incorrectes. Ce diagramme montre clairement les interactions entre l'utilisateur, la page de connexion, le serveur et la base de données, mettant en évidence les étapes cruciales de la vérification des informations d'identification.**

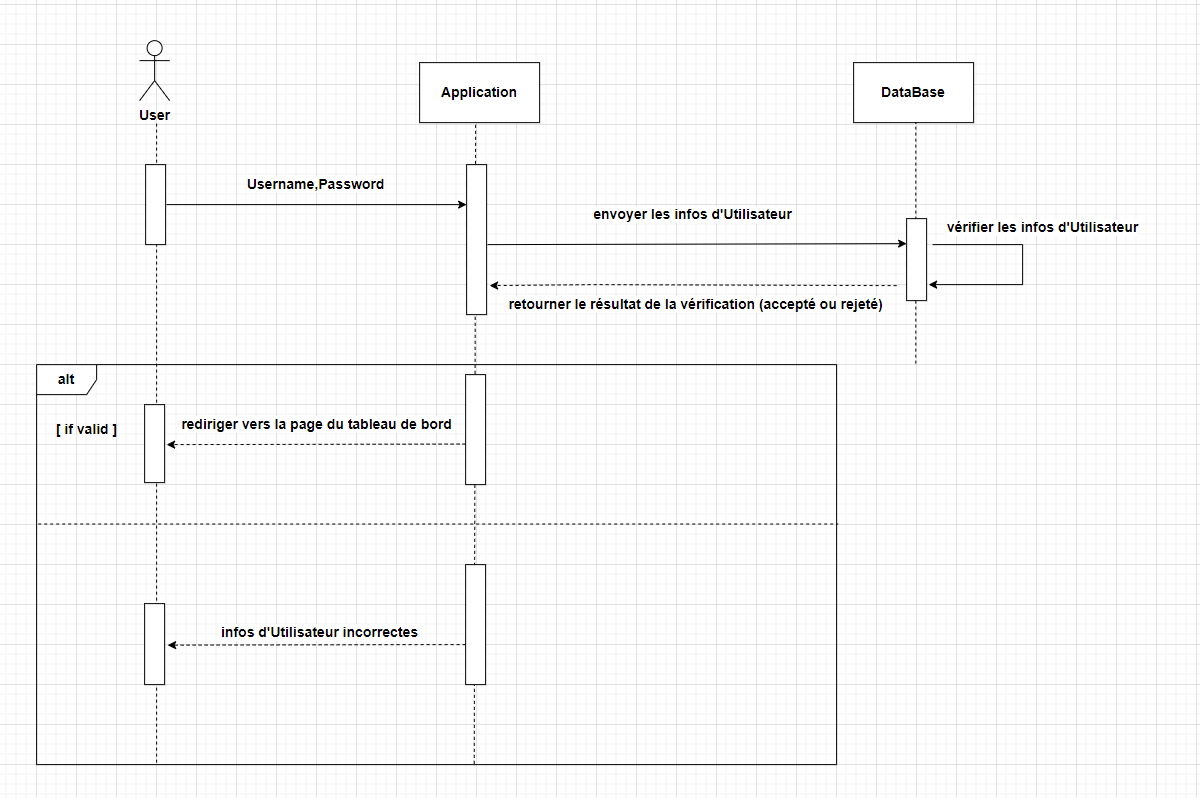


Figure 3.3: DIAGRAMME DE SEQUENCE AUTHENTIFICATION

#### ****3.3.2.2 Diagramme de séquence**** Consulter les Clients

Le diagramme de séquence ci-dessus décrit le processus par lequel un administrateur (Admin) consulte les clients via l'application. Le processus débute lorsque l'administrateur demande à consulter les clients. L'application transmet cette demande à la base de données (Database) pour récupérer la liste des clients pertinents. La base de données renvoie la liste des clients à l'application, qui la transfère ensuite à l'administrateur. Enfin, l'application affiche la liste des clients à l'administrateur. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour consulter les clients.

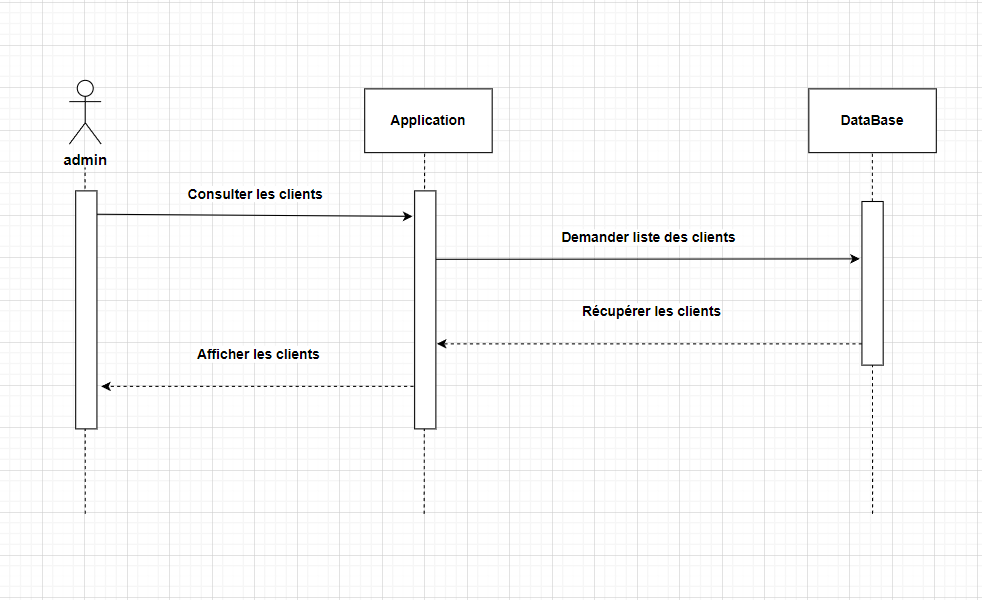


Figure 3.4: DIAGRAMME DE SÉQUENCE CONSULTER LES CLIENTS

#### ****3.3.2.3** **Diagramme de séquence**** Consulter Les Notifications

Le diagramme de séquence ci-dessus décrit le processus par lequel un administrateur (Admin) consulte les notifications via l'application. Le processus débute lorsque l'administrateur demande l'affichage des notifications. L'application transmet cette demande à la base de données (Database) pour récupérer les notifications pertinentes. La base de données renvoie les notifications à l'application, qui les transfère ensuite à l'administrateur. Enfin, l'application affiche les notifications à l'administrateur. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour consulter les notifications.

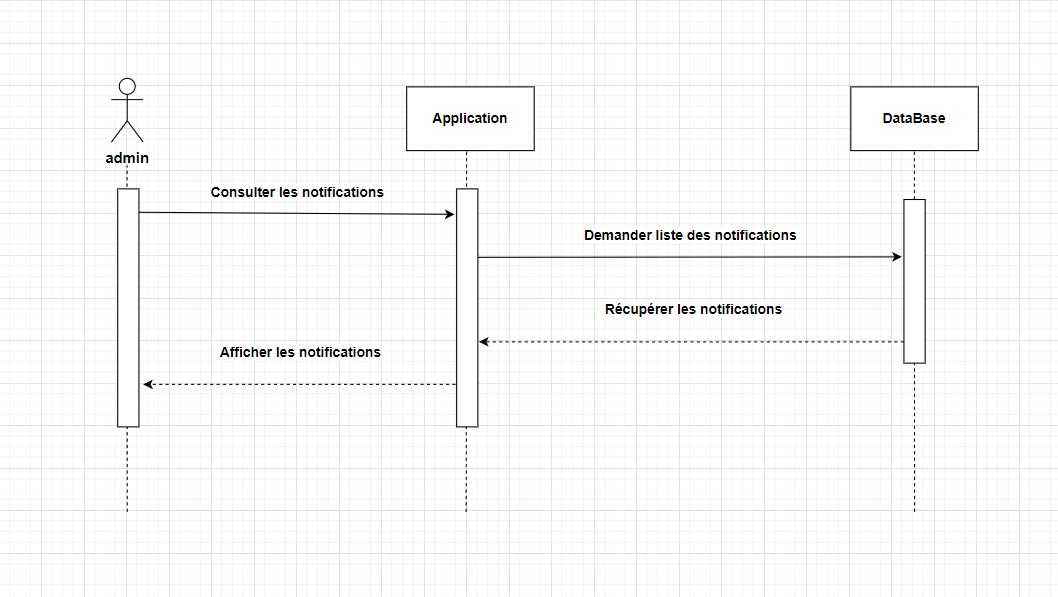


Figure 3.5: DIAGRAMME DE SÉQUENCE CONSULTER LES NOTIFICATIONS

#### ****3.3.2.4 Diagramme de séquence Consulter**** les Réclamations

Le diagramme de séquence ci-dessus décrit le processus par lequel un administrateur (Admin) consulte les réclamations via l'application. Le processus débute lorsque l'administrateur demande à consulter les réclamations. L'application transmet cette demande à la base de données (Database) pour récupérer les réclamations pertinentes. La base de données renvoie les réclamations à l'application, qui les transfère ensuite à l'administrateur. Enfin, l'application affiche les réclamations à l'administrateur. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour consulter les réclamations.

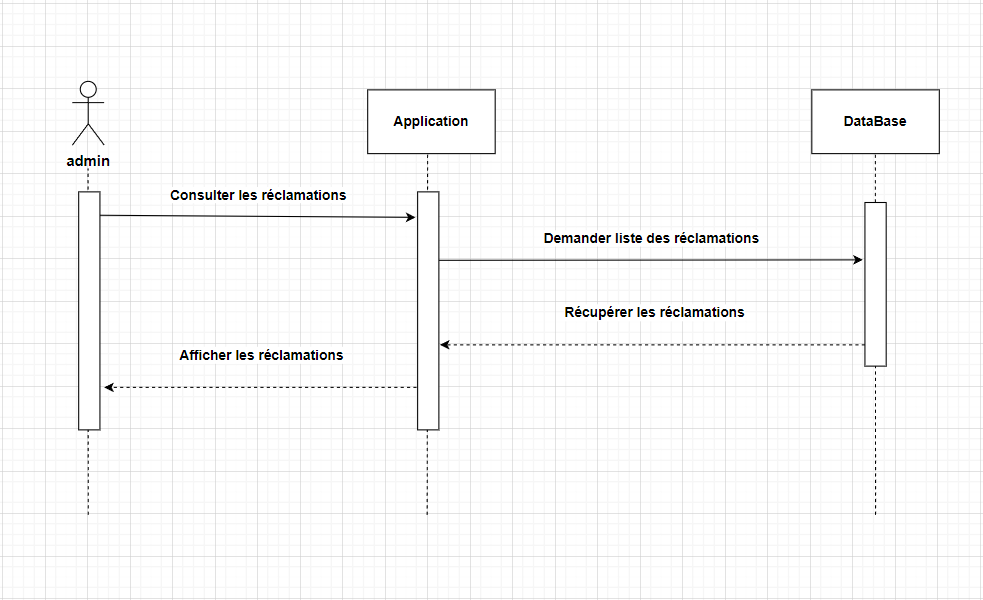


Figure 3.6: DIAGRAMME DE SÉQUENCE CONSULTER LES RÉCLAMATIONS

#### ****3.3.2.5 Diagramme de séquence Traiter les réclamations des clients****

Le diagramme de séquence ci-dessus décrit le processus par lequel un administrateur (Admin) traite une réclamation via l'application. Le processus débute lorsque l'administrateur fait le traitement de la réclamation. L'application transmet cette demande à la base de données (Database) pour mettre à jour le statut de la réclamation. La base de données confirme la mise à jour du statut à l'application, qui à son tour informe l'administrateur que la mise à jour a été réussie. Ensuite, l'administrateur crée une notification relative à cette mise à jour et l'enregistre dans la base de données. Une fois la notification enregistrée, la base de données confirme cet enregistrement à l'application. Enfin, l'application informe l'administrateur que la réclamation a été traitée avec succès. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour traiter une réclamation.

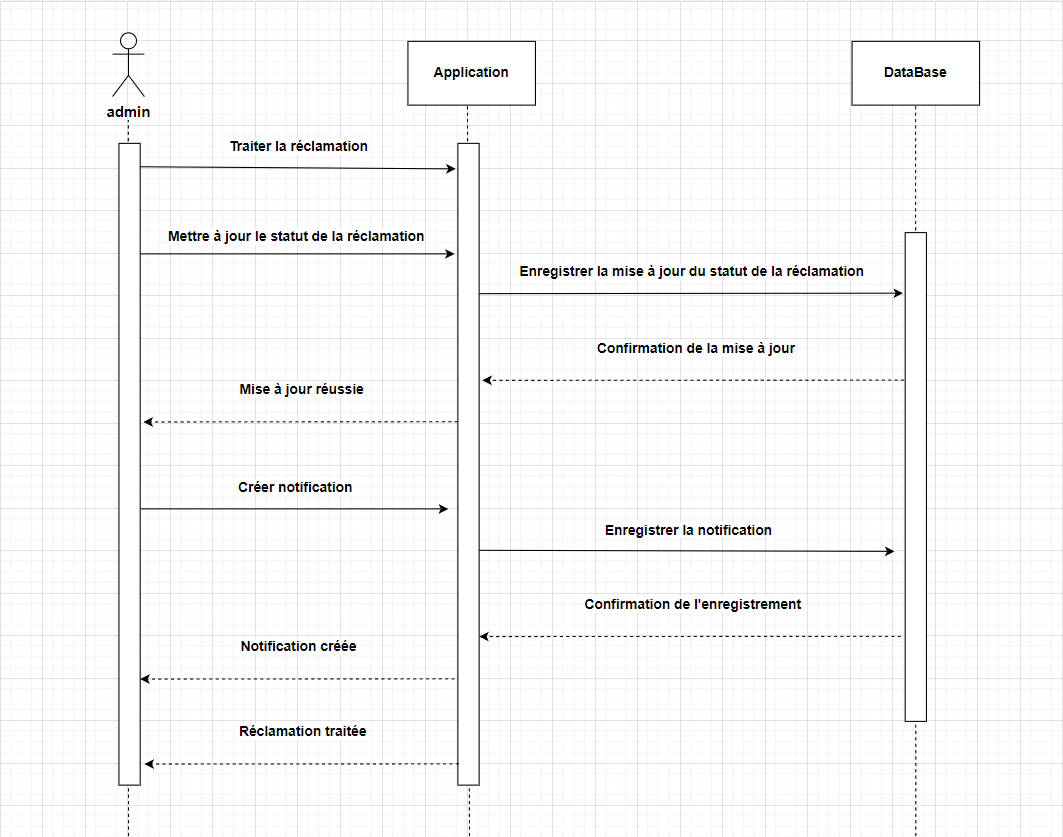
****

Figure 3.7: TRAITER LES RECLAMATIONS DES CLIENTS

#### ****3.3.2.6**** Compte activé désactivé admin

Le diagramme de séquence ci-dessus décrit le processus par lequel un administrateur (Admin) active ou désactive un compte via l'application. Le processus débute lorsque l'administrateur demande l'activation ou la désactivation d'un compte. L'application transmet cette demande à la base de données (Database) pour mettre à jour le statut du compte. La base de données confirme la mise à jour du statut à l'application, qui à son tour informe l'administrateur que le compte a été activé ou désactivé. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour activer ou désactiver un compte.

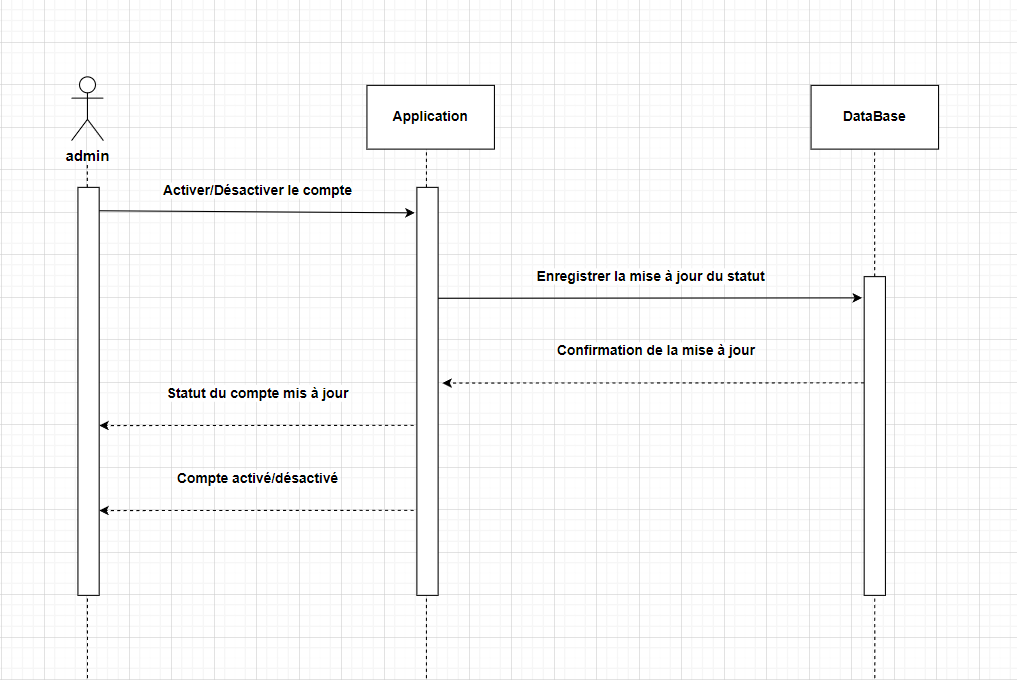


Figure 3.8 : DIAGRAMME DE SÉQUENCE ACTIVER DÉSACTIVER COMPTE CLIENT

#### ****3.3.2.7 Diagramme de séquence Supprimer une Notification****

Le diagramme de séquence ci-dessous décrit le processus par lequel un administrateur (Admin) supprime une notification via l'application. Le processus débute lorsque l'administrateur demande la suppression d'une notification. L'application transmet cette demande à la base de données (Database) pour supprimer une notification. La base de données confirme la suppression de notification à l'application, qui à son tour informe l'administrateur que la notification a été supprimée. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour supprimer une notification.

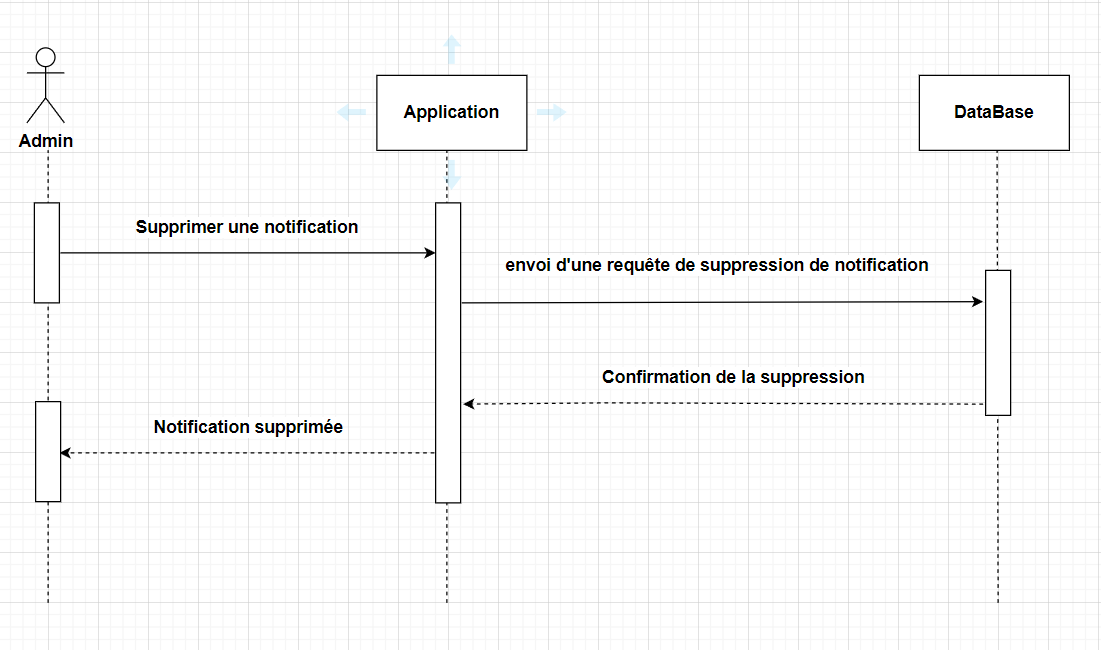


Figure 3.9: DIAGRAMME DE SEQUENCE SUPPRIMER UNE NOTIFICATION

#### ****3.3.2.8 Diagramme de séquence Modifier une Notification****

**Le diagramme de séquence ci-dessous décrit le processus par lequel un administrateur (Admin) modifie le contenu d'une notification via l'application. Le processus débute lorsque l'administrateur demande la modification d'une notification. L'application transmet cette demande à la base de données (Database) pour mettre à jour le contenu de la notification. La base de données confirme la mise à jour du contenu à l'application, qui à son tour informe l'administrateur que la notification a été modifiée. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour modifier le contenu d'une notification.**

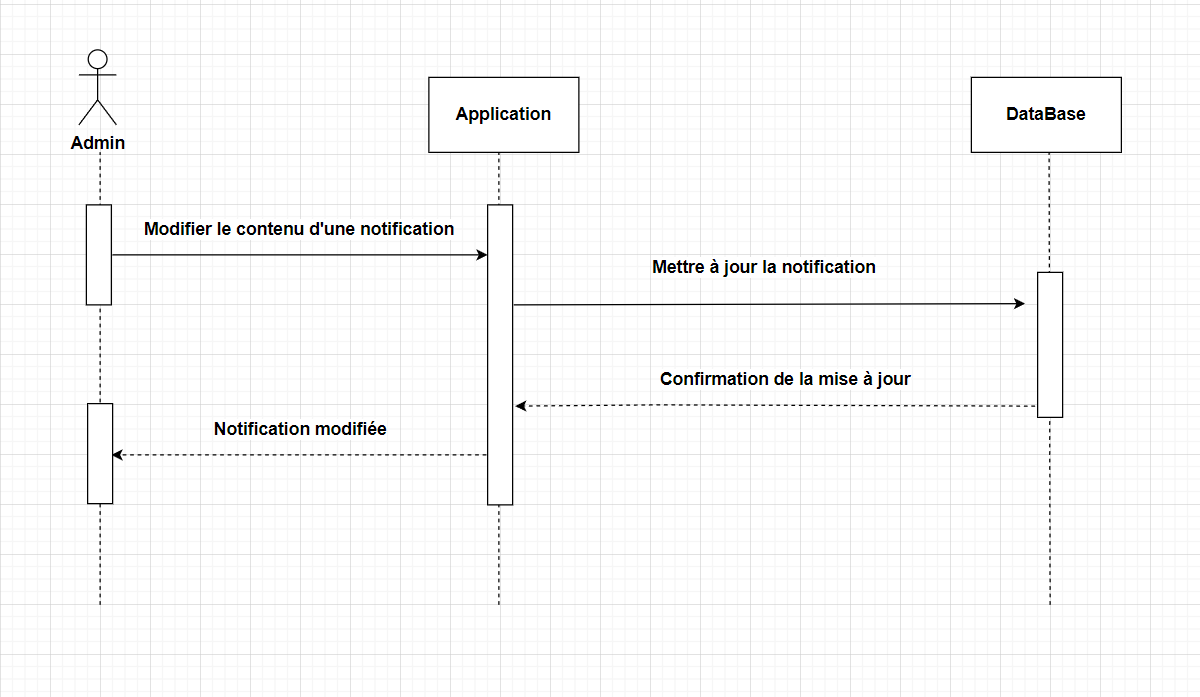


Figure 3.10: DIAGRAMME DE SEQUENCE MODIFIER UNE NOTIFICATION

#### ****3.3.2.9 Diagramme de séquence Créer compte pour un client****

Le diagramme de séquence ci-dessous décrit le processus par lequel un administrateur (Admin) crée un compte pour un client via l'application. Le processus débute lorsque l'administrateur fournit les informations nécessaires pour créer le compte. L'application transmet cette demande à la base de données (Database) pour enregistrer les informations du nouveau client. La base de données confirme l'enregistrement du nouveau compte à l'application, qui à son tour informe l'administrateur que le compte client a été créé. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour créer un compte client.

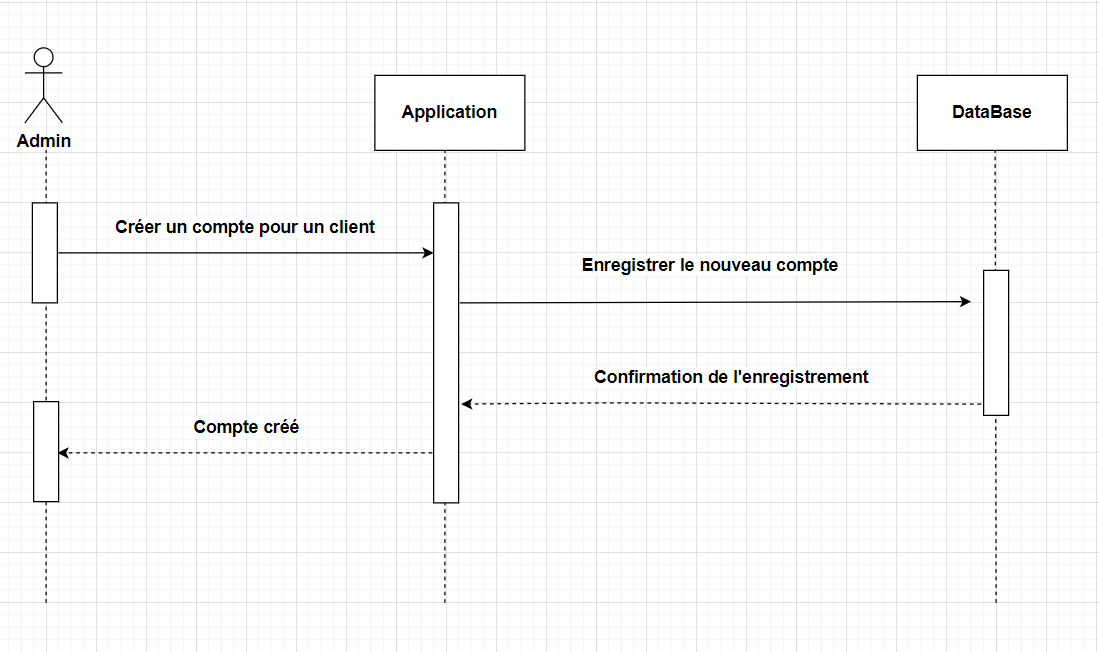


Figure 3.11: DIAGRAMME DE SEQUENCE CREER COMPTE POUR UN CLIENT

#### ****3.3.2.10 Diagramme de séquence Client Consulte**** Ses Réclamations

Le diagramme de séquence ci-dessus décrit le processus par lequel un client (Customer) consulte ses réclamations via l'application. Le processus débute lorsque le client demande à consulter ses réclamations. L'application transmet cette demande à la base de données (Database) pour récupérer les réclamations pertinentes. La base de données renvoie les réclamations à l'application, qui les transfère ensuite au client. Enfin, l'application affiche les réclamations au client. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour consulter les réclamations.

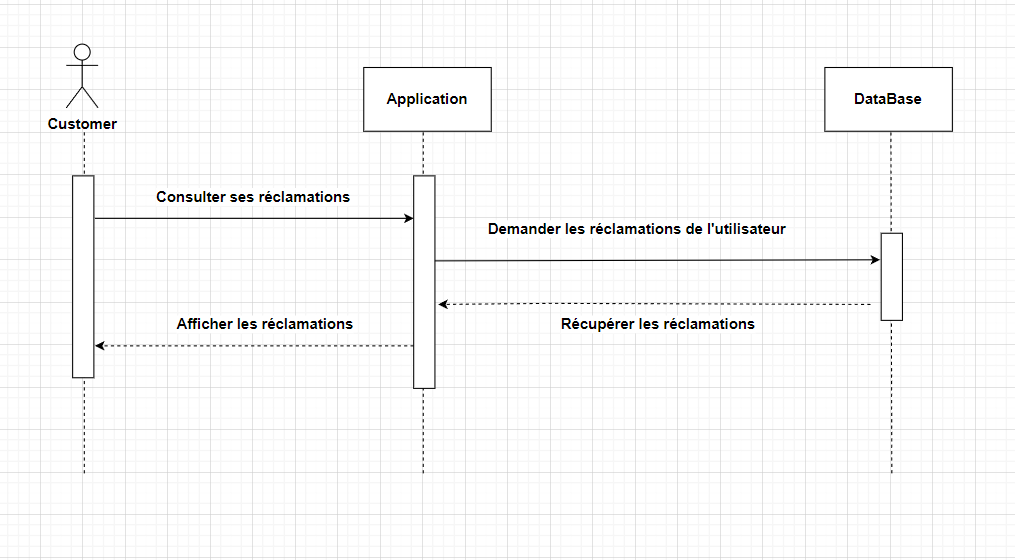


Figure 3.12 : DIAGRAMME DE SÉQUENCE AFFICHER LES RÉCLAMATIONS

#### ****3.3.2.11 Diagramme de séquence Client Consulte**** Ses Notifications

Le diagramme de séquence ci-dessus décrit le processus par lequel un client (Customer) consulte ses notifications via l'application. Le processus débute lorsque le client demande à consulter ses notifications. L'application transmet cette demande à la base de données (Database) pour récupérer les notifications pertinentes. La base de données renvoie les notifications à l'application, qui les transfère ensuite au client. Enfin, l'application affiche les notifications au client. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour consulter les réclamations.

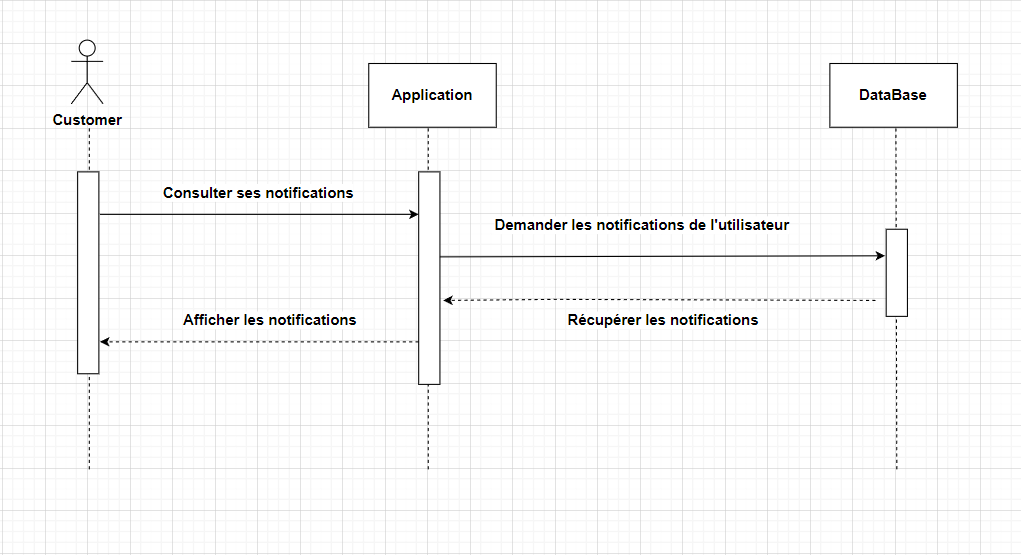


Figure 3.13 : DIAGRAMME DE SÉQUENCE CLIENT CONSULTE SES RÉCLAMATIONS

#### ****3.3.2.12 Diagramme de Séquence pour le Processus D’envoyer Réclamation Client****

Ce diagramme de séquence illustre le processus par lequel un client soumet une réclamation via une interface. Le client commence par saisir les détails de sa réclamation sur la page dédiée. Une fois les informations complétées, la réclamation est envoyée au serveur, qui la traite en l'enregistrant dans la base de données. La base de données confirme ensuite l'enregistrement de la réclamation, indiquant au serveur si l'opération a réussi. Le serveur renvoie cette confirmation à le client, qui affiche alors un message au client. Si l'enregistrement est un succès, la réclamation est affichée comme validée.

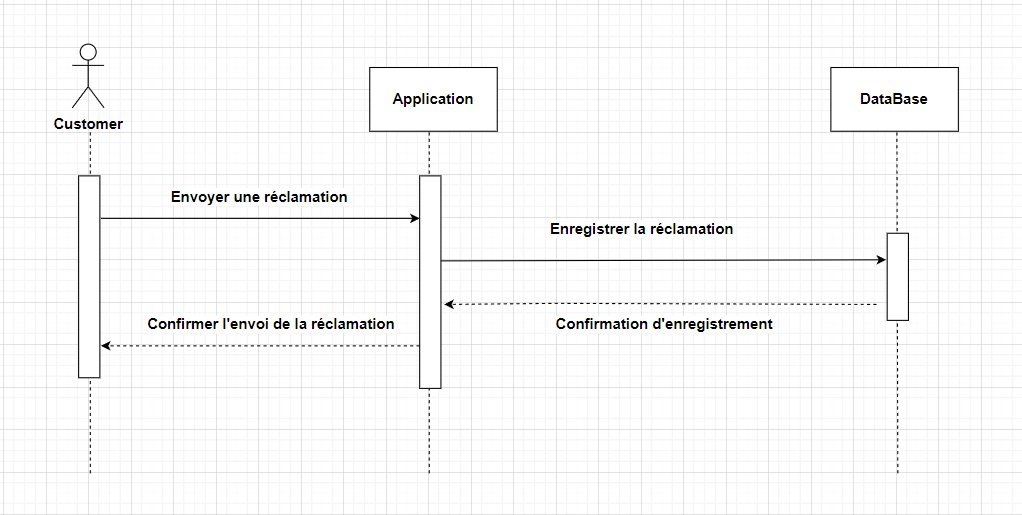


Figure 3.14 : DIAGRAMME DE SÉQUENCE CLIENT ENVOIE UNE RÉCLAMATION

#### ****3.3.2.13 Diagramme de Séquence modifier le contenu de la réclamation****

Le diagramme de séquence ci-dessous décrit le processus par lequel un client modifie le contenu d'une réclamation via l'application. Le processus débute lorsque le client soumet les modifications de la réclamation. L'application transmet cette demande à la base de données (Database) pour mettre à jour le contenu de la réclamation. La base de données confirme la mise à jour de la réclamation à l'application, qui à son tour informe le client que la réclamation a été modifiée. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour modifier le contenu d'une réclamation.

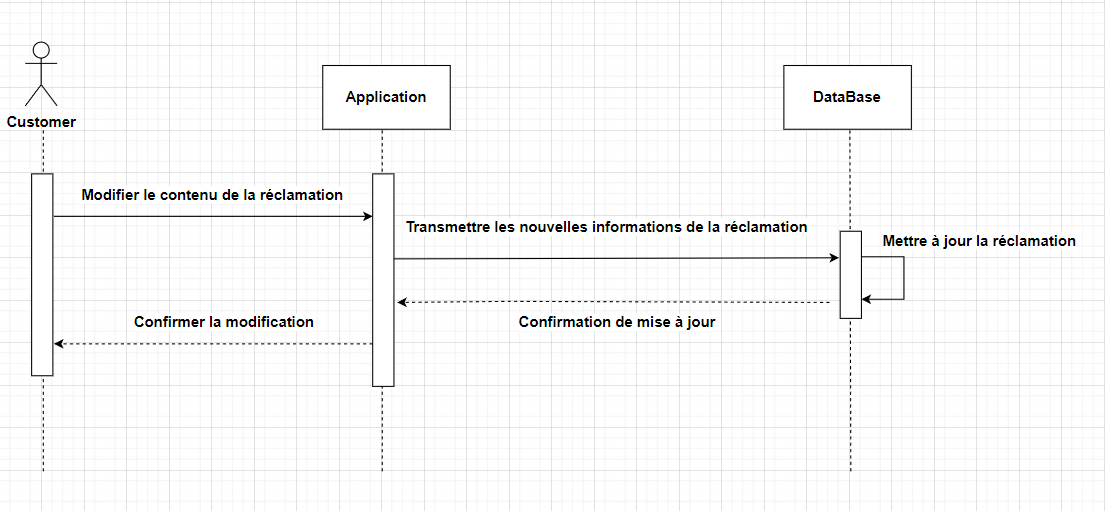


Figure 3.15: DIAGRAMME DE SEQUENCE MODIFIER LE CONTENU DE LA RECLAMATION

#### ****3.3.2.14 Diagramme de Séquence Supprimer une réclamation****

Le diagramme de séquence ci-dessous décrit le processus par lequel un client supprime une réclamation via l'application. Le processus débute lorsque le client demande la suppression d'une réclamation. L'application transmet cette demande à la base de données (Database) pour supprimer la réclamation correspondante. La base de données confirme la suppression à l'application, qui à son tour informe le client que la réclamation a été supprimée. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour supprimer une réclamation.

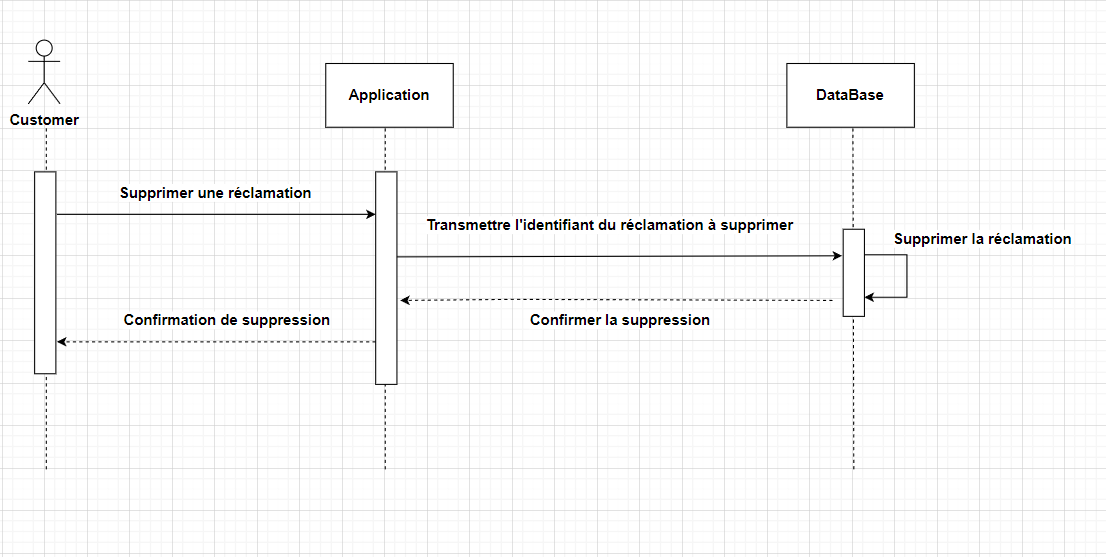


Figure 3.16 : DIAGRAMME DE SÉQUENCE CLIENT SUPPRIME UNE RÉCLAMATION

#### ****3.3.2.15 Diagramme de Séquence Client Modifie ses Coordonnées****

**Le diagramme de séquence ci-dessous décrit le processus par lequel un client modifie ses coordonnées via l'application. Le processus débute lorsque le client a reçu ses coordonnées. L'application transmet cette demande à la base de données (Database) pour mettre à jour les coordonnées du client. La base de données confirme la mise à jour des coordonnées à l'application, qui à son tour informe le client que les coordonnées ont été modifiées. Ce diagramme illustre clairement les interactions et le flux d'informations entre les différents acteurs et composants du système, mettant en évidence les étapes nécessaires pour modifier les coordonnées du client.**

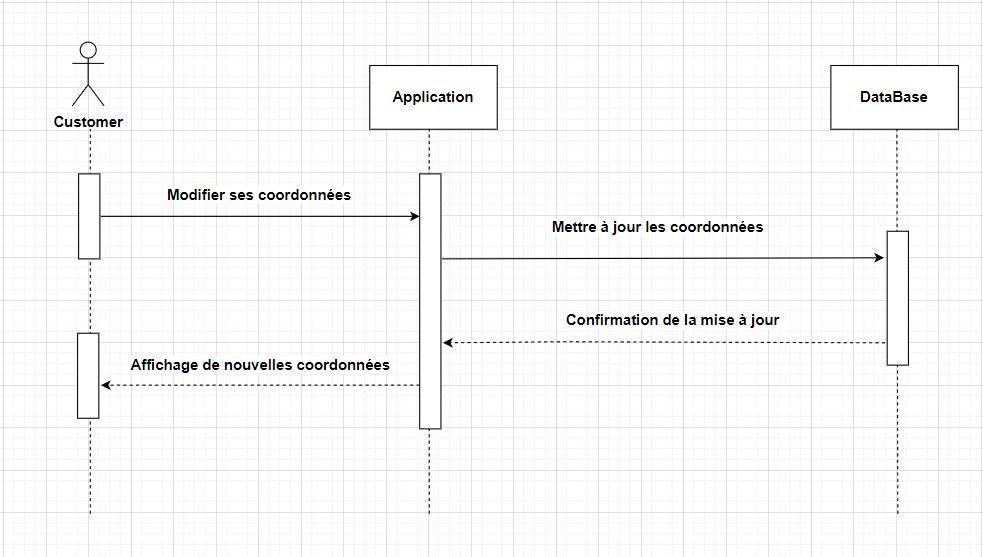


Figure 3.17: DIAGRAMME DE SEQUENCE CLIENT MODIFIER SES COORDONNÉES

### ****3.3.3 Analyse objet****

Le diagramme de classes permet de représenter l’aspect statique du système en termes de classes et des relations entre ces classes, en identifiant les différents attributs et méthodes qui reflètent respectivement les propriétés et les services offerts par notre système. La figure suivante présente le diagramme de classes de ce projet.

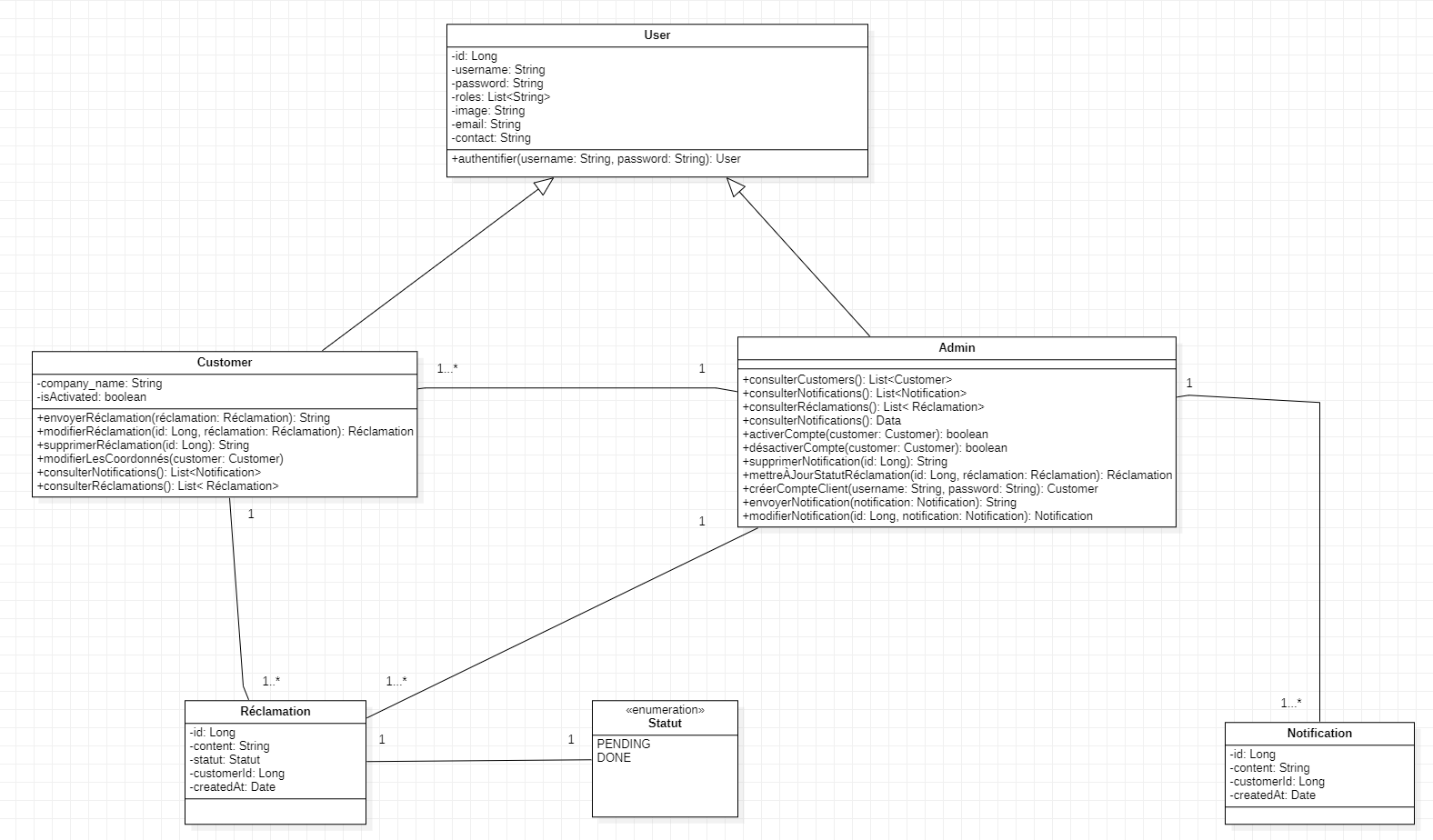


Figure 3.18: DIAGRAMME DE CLASSE

## 3.4 L’architecture des MicroServices de L’application

Cette architecture présente un modèle de microservices pour une application backend, intégrant plusieurs composants essentiels pour la gestion des services et des configurations. Les utilisateurs ou autres systèmes interagissent avec l'application via des interfaces clients. Une passerelle API (Spring Cloud Gateway) sert de point d'entrée unique pour les clients, routant les requêtes vers les services appropriés et assurant la sécurité, la gestion des taux et d'autres fonctionnalités de gestion des API.

Un service de registre (Eureka) permet la découverte dynamique des services, les microservices s'enregistrent auprès d'Eureka, et les autres services peuvent découvrir et interagir avec eux. Un service de configuration centralisé (Config Service) gère les configurations des microservices, assurant une cohérence et une gestion facile des paramètres de configuration. Les microservices comprennent le Claims Service qui gère les réclamations des utilisateurs et stocke les données associées, le Customers Service qui gère les informations des clients, assurant la gestion et l'accès aux profils des utilisateurs, et le Notifications Service qui gère l'envoi et la gestion des notifications aux utilisateurs.

Les clients envoient des requêtes via l'API Gateway, et le Spring Cloud Gateway route ces requêtes vers les microservices appropriés. Les microservices s'enregistrent auprès du Registry Eureka Service pour permettre leur découverte dynamique. Le Spring Cloud Gateway utilise Eureka pour découvrir les microservices. Les microservices peuvent communiquer entre eux directement via des appels REST ou en utilisant Feign pour effectuer des requêtes HTTP. Le Config Service fournit les configurations nécessaires aux microservices, assurant une gestion centralisée et cohérente des paramètres de configuration.

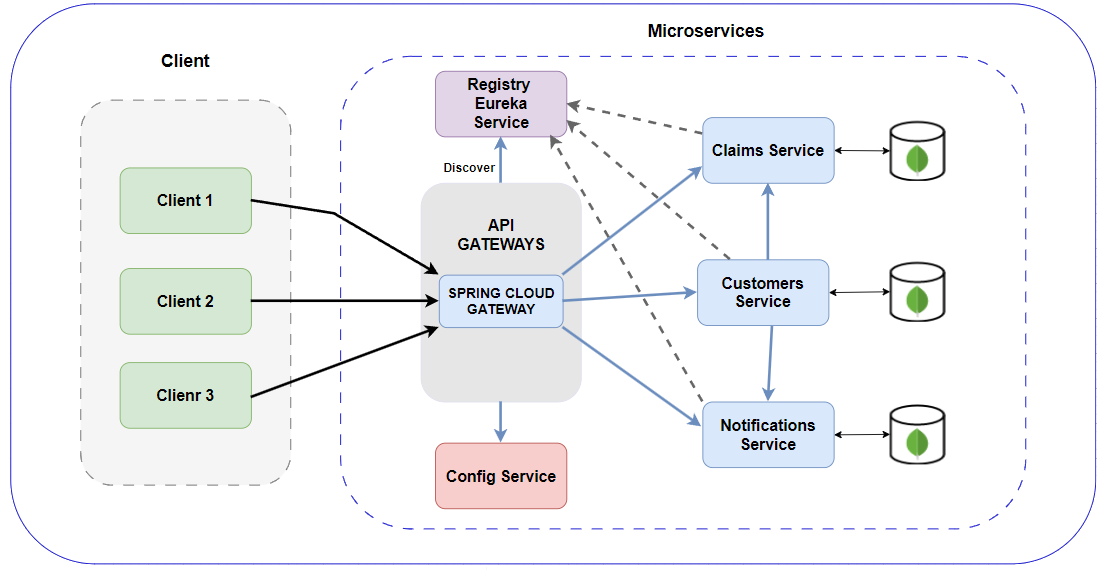


Figure 4.19: ARCHITECTUR MICROSERVICES

## Résumé

Le diagramme illustre une architecture de microservices utilisant Spring Cloud Gateway, Eureka, et divers services backend. Les clients interagissent avec Spring Cloud Gateway, qui agit comme point d'entrée unique, redirigeant les requêtes vers les services appropriés après authentification et autorisation. Les services, incluant Claims Service, Customers Service, et Notifications Service, s'enregistrent auprès de Eureka Registry pour annoncer leur disponibilité, permettant une découverte dynamique par Gateway. Config Service fournit une configuration centralisée aux autres services, facilitant une gestion dynamique des configurations. Les services interagissent entre eux via des appels REST ou Feign, assurant une communication flexible. Cette architecture favorise la flexibilité, la scalabilité et la résilience grâce à la décentralisation, à la découverte de services automatisée et à la configuration centralisée.

## 3.5 Conclusion

En somme, ce chapitre a traité des différentes phases de la conception détaillée, allant de l'amélioration du diagramme de cas d'utilisation au développement de plusieurs diagrammes de séquence pour décrire les interactions entre les acteurs et le système, pour finalement aboutir à la présentation de la structure du module via le diagramme de classe.

**Chapitre 4**

# Mise en œuvre

* **Outils et Technologies**
* **Présentation de la Solution**

## ****4.1 Outils et Technologies****

### ****4.1.1 Spring boot****



Figure 4.1: SPRING BOOT LOGO

**Spring Boot est un framework de développement basé sur Java, conçu pour simplifier la création d'applications autonomes et prêtes à être mises en production. Maintenu par Pivotal Software, il est open source et gratuit. Spring Boot facilite la configuration et le développement grâce à son approche "convention over configuration" et à ses dépendances intégrées. Il permet de créer des applications web, cloud, microservices, et bien plus encore, en réduisant considérablement la complexité et le temps de développement. Grâce à ses puissantes fonctionnalités, comme les démarrages automatiques et la facilité d'intégration avec d'autres technologies, Spring Boot est largement adopté pour développer des applications robustes et évolutives.**

### ****4.1.2 Mongo DB****



Figure 4.2: MONGO DB LOGO

**MongoDB est une base de données NoSQL orientée documents, conçue pour gérer de grandes quantités de données et offrir une grande flexibilité de modélisation des données. Développée par MongoDB Inc., cette base de données open source stocke les données sous forme de documents JSON, permettant une représentation naturelle et intuitive des données complexes. MongoDB est particulièrement adaptée aux applications nécessitant une grande évolutivité et des performances élevées. Elle est utilisée pour une variété de cas d'utilisation, tels que les applications web, les systèmes de gestion de contenu, les solutions Big Data, l'Internet des objets (IoT), et bien plus encore.**

### ****4.1.3 JavaScript****

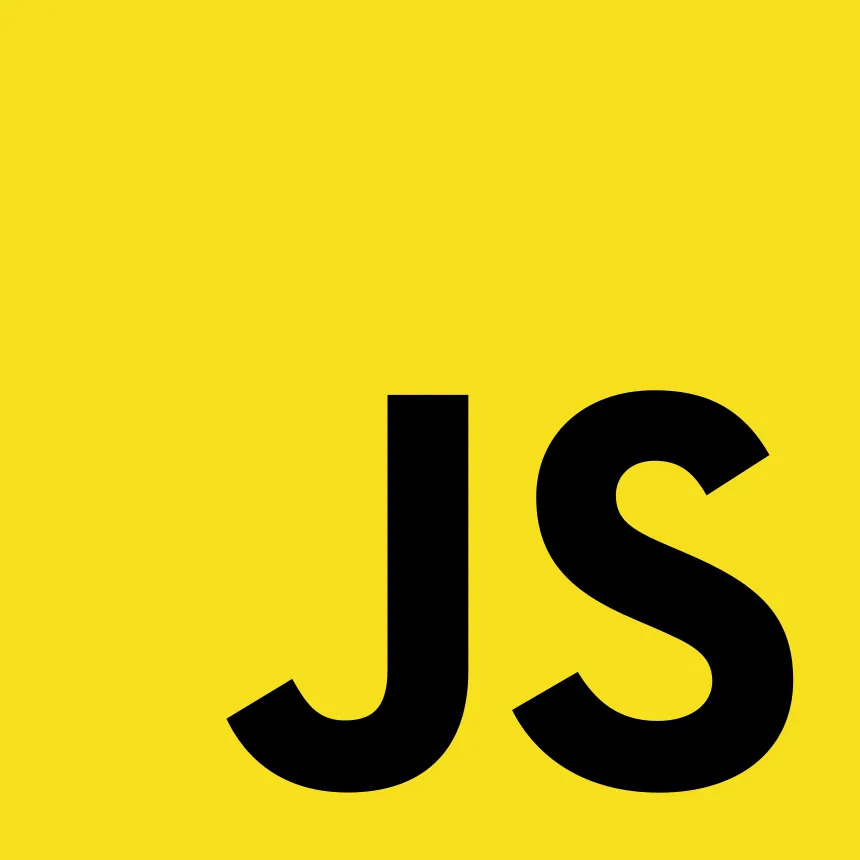


Figure 4.3: JAVASCRIPT LOGO

**JavaScript est un langage de programmation dynamique et interprété, largement utilisé pour le développement de pages web interactives. Il permet d'ajouter des fonctionnalités comme des animations, des mises à jour de contenu en temps réel, et des interactions avec l'utilisateur directement dans le navigateur. JavaScript est également utilisé côté serveur avec des environnements comme Node.js, ce qui permet de créer des applications web complètes avec un seul langage. Grâce à sa flexibilité et à son intégration native dans les navigateurs, JavaScript est devenu un pilier essentiel du développement web moderne, soutenu par une vaste communauté et un écosystème riche en bibliothèques et frameworks comme React, Angular, et Vue.js.**

### ****4.1.4 TypeScript****



Figure 4.4 : TYPESCRIPT LOGO

**TypeScript est un langage de programmation open source développé par Microsoft, conçu comme une surcouche syntaxique de JavaScript. Il ajoute des fonctionnalités telles que le typage statique, les interfaces, et les types énumérés, améliorant ainsi la fiabilité et la maintenabilité des applications. En permettant de détecter les erreurs lors de l'écriture du code, TypeScript facilite la structuration et l'organisation des projets complexes. Il est entièrement compatible avec JavaScript, ce qui permet une adoption progressive dans les projets existants.**

### ****4.1.5 React JS****

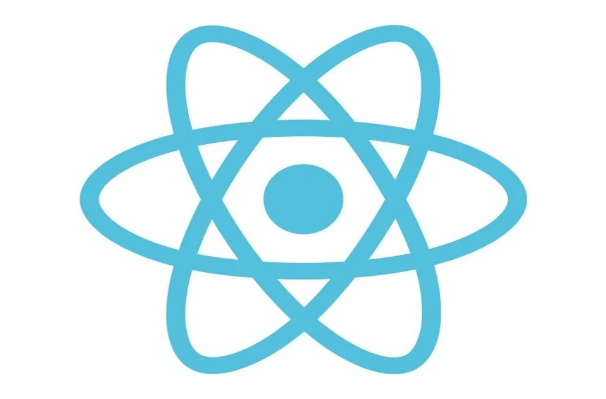
****

Figure 4.5: REACT JS LOGO

**React est une bibliothèque JavaScript open source, principalement utilisée pour construire des interfaces utilisateur dynamiques et réactives. Développé par Facebook, React permet de créer des composants réutilisables qui gèrent leur propre état, facilitant ainsi la création de grandes applications web où les données changent fréquemment. Grâce à son approche basée sur les composants, React rend le développement plus structuré et modulable. Il utilise un DOM virtuel pour optimiser les mises à jour de l'interface utilisateur, améliorant ainsi les performances. React est souvent utilisé en combinaison avec d'autres bibliothèques ou frameworks, comme Redux pour la gestion de l'état global, ou React Router pour le routage. Sa popularité et son écosystème riche font de React un choix privilégié pour les développeurs souhaitant construire des applications web modernes et performantes.**

### ****4.1.6 Redux****

****

Figure 4.6: REDUX LOGO

**Redux est une bibliothèque JavaScript open source conçue pour gérer l'état global des applications. Il est souvent utilisé avec des bibliothèques comme React ou Angular pour construire des applications front-end robustes et maintenables. Redux fonctionne sur le principe d'un seul magasin d'état, ce qui signifie que l'état de l'application est stocké dans un objet unique. Les modifications de l'état sont effectuées via des actions, qui sont des objets décrivant les événements se produisant dans l'application. Ces actions sont traitées par des réducteurs (reducers), des fonctions pures qui déterminent comment l'état doit changer en réponse aux actions.**

**Redux offre une structure prévisible pour la gestion de l'état, facilitant le débogage et les tests, et permettant une meilleure gestion des applications complexes. Avec Redux, les développeurs peuvent suivre l'historique des actions et de l'état, ce qui aide à maintenir la cohérence de l'application. Grâce à sa compatibilité avec les outils de développement, comme Redux DevTools, il devient plus facile de surveiller et de manipuler l'état en temps réel. En somme, Redux est un outil puissant pour la gestion de l'état dans les applications JavaScript modernes, offrant une solution scalable et maintenable.**

### ****4.1.7 Itellij IDEA****



Figure 4.7: ITELLIJ IDEA LOGO

**IntelliJ IDEA est un environnement de développement intégré (IDE) complet et avancé, développé par JetBrains, principalement destiné au développement en Java. Réputé pour ses puissantes fonctionnalités de productivité, IntelliJ IDEA offre une expérience de codage fluide grâce à son auto-complétion intelligente, son analyse de code en temps réel, et ses outils de refactoring robustes. Il supporte une multitude de langages de programmation et de frameworks, incluant Kotlin, Scala, Groovy, et bien d'autres, en plus de Java. IntelliJ IDEA intègre également des outils de débogage et de test, une gestion de version intégrée (Git, SVN, etc.), et une riche collection de plugins disponibles pour étendre ses fonctionnalités. Grâce à ses capacités avancées d'édition et de navigation de code, son interface utilisateur intuitive, et ses outils de développement Web et mobile, IntelliJ IDEA est largement utilisé par les développeurs pour des projets allant des petites applications aux grandes solutions d'entreprise.**

### ****4.1.8 Visual Studio Code****



Figure 4.8: VISUAL STUDIO CODE LOGO

**Visual Studio Code (VS Code) est un éditeur de code source léger mais puissant, développé par Microsoft. Gratuit et open-source, VS Code est apprécié pour sa rapidité et sa capacité à être extensible. Il prend en charge une vaste gamme de langages de programmation grâce à des extensions, permettant aux développeurs de personnaliser leur environnement de travail selon leurs besoins spécifiques. Les fonctionnalités de VS Code incluent une coloration syntaxique, une auto-complétion intelligente, un débogueur intégré, et une intégration Git pour le contrôle de version. De plus, il offre un terminal intégré, des outils de refactoring, et une riche collection d'extensions disponibles sur le marketplace. Grâce à sa flexibilité et sa puissance, VS Code est devenu l'éditeur de choix pour de nombreux développeurs travaillant sur différents types de projets, des petites applications aux grandes solutions d'entreprise.**

### ****4.1.9 Postman****



Figure 4.9: POSTMAN LOGO

**Postman est un outil de développement d'API complet qui simplifie la création, le partage, le test et la documentation des API. Avec une interface utilisateur intuitive, Postman permet aux développeurs de configurer rapidement des requêtes HTTP, d'inspecter les réponses et d'automatiser les tests d'API. Les fonctionnalités de collaboration de Postman facilitent le partage de collections de requêtes, l'exécution de tests en équipe et la gestion de versions d'API. Grâce à son environnement de développement intégré, Postman améliore la productivité des développeurs en fournissant des outils puissants pour simuler des environnements, générer des rapports de tests et garantir la conformité des API. Utilisé largement dans le développement d'API RESTful, Postman est un atout précieux pour toute équipe de développement cherchant à améliorer l'efficacité et la qualité de ses interactions API.**

### ****4.1.10 Tailwind CSS****

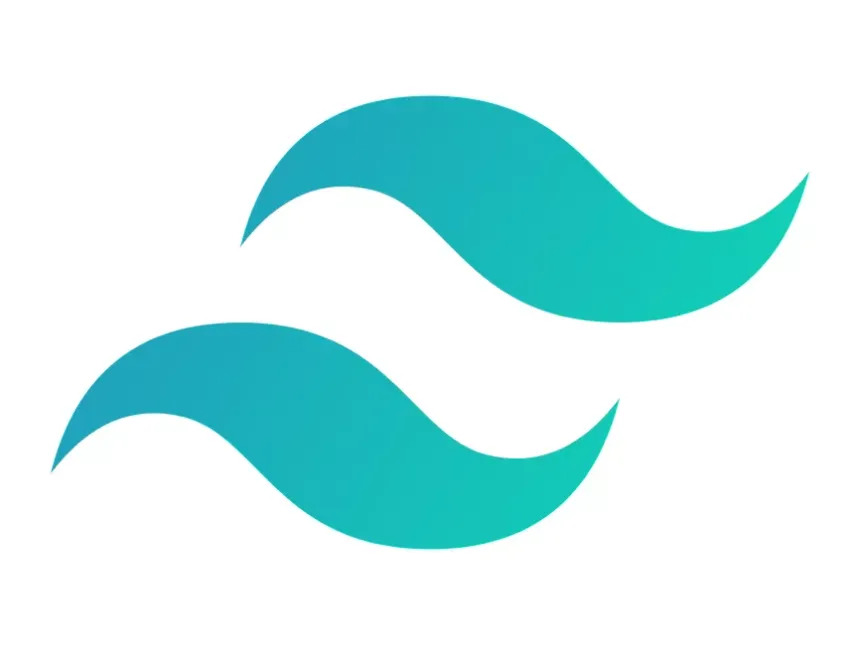


Figure 4.10: TAILWIND CSS LOGO

**Tailwind CSS est un framework CSS utilitaire moderne qui permet de concevoir des interfaces utilisateur rapidement et efficacement. Contrairement aux frameworks traditionnels, Tailwind CSS offre une approche "utility-first" où les classes utilitaires sont utilisées pour appliquer des styles directement dans le balisage HTML. Cette méthode permet une grande flexibilité et un contrôle granulaire sur la conception, éliminant le besoin de styles personnalisés étendus. Tailwind CSS est hautement configurable, permettant aux développeurs de personnaliser facilement les thèmes, les palettes de couleurs et les styles globaux pour répondre aux besoins spécifiques de leur projet. Utilisé pour créer des interfaces utilisateur élégantes et réactives, Tailwind CSS est particulièrement populaire dans les projets web modernes où la rapidité de développement et la maintenance simplifiée sont essentielles.**