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

Mémoire de fin de cycle

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

Option : Génie logiciel

Conception et réalisation d'une application mobile de livraison

---

Réalisé par : Mlle. HADRI Lylia

Devant le jury composé de

Président :	Mme. EL BOUHISSI Houda	Université de Bejaia
Examinateur :	M. AMROUN Kamal	Université de Bejaia
Encadrant :	M. ALLEM Khaled	Université de Bejaia

Promotion 2020 - 2021

Remerciements

Mes remerciements s'adressent au bon Dieu le tout puissant et miséricordieux, qui m'a donné la force, la volonté et la patience pour accomplir ce modeste travail.

Je remercie **M. ALLEM Khaled** pour avoir accepté de diriger ce travail. Ses conseils m'ont été d'une aide inestimable. Je remercie également le responsable de l'agence web Techsys media **M. MELLOUK Cherif** ainsi que les membres du projet pour les informations et les réunions de travail qui ont participé à la réalisation du projet.

Je tiens à remercier sincèrement les membres du jury qui me font le grand honneur d'évaluer ce travail.

Enfin, je remercie mes parents et mon frère pour leur soutien et leurs encouragements tout au long de mes études.

Dédicaces

A mes parents, pour leurs sacrifices déployés à mon égard, pour leur patience, leur amour et leur confiance.

A mon frère, mon guide et ma fierté, que Dieu le garde en bonne santé.

A tous les membres de ma famille.

Qu'ils trouvent dans ce modeste travail, le témoignage de ma profonde affection et de mon attachement indéfectible. Nulle dédicace ne peut exprimer ce que je leur dois.

Table des matières

Table des figures

Liste des tableaux

Glossaire

API (Application Programming Interface) : est un ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications.

FCM (Firebase cloud Messaging) : est une solution cloud multiplateforme pour les messages et les notifications pour les applications Android, iOS et Web.

HTTP (Hypertext Transfer Protocol) : C'est le protocole de transfert sur internet le plus courant.

IDE (Integrated Development Environment) : est une interface qui permet de développer, compiler et exécuter un programme dans un langage donné.

NoSql (Not Only Sql) : désigne une famille de systèmes de gestion de base de données qui s'écarte du paradigme classique des bases relationnelles.

SDK (Sotfware Development Kit) : désigne un ensemble d'outils utilisés par les développeurs pour le développement d'un logiciel destiné à une plateforme déterminée (Linux, Windows, Android, etc.)

SQL (Structured Query Language) : est un langage permettant d'interroger une base de données relationnelle.

UML (Unified Modeling Language) : se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes.

UP (Unified Process) : Le processus Unifié est un processus de réalisation ou d'évolution de logiciel entièrement basé sur UML.

Introduction générale

Les applications informatiques de livraison à la demande commencent à se répandre en Algérie depuis quelques années, ces applications sont disponibles dans les grandes villes comme Alger, Oran et Annaba qui proposent la livraison de nourriture et de produits essentiels ou de transport urbains. En effet, depuis le lancement de jumia en 2014 d'autres plateformes ont vu le jour comme yassir ce qui a donné aux citoyens plus de choix de produits à livrer et a participé à accroître les exigences des utilisateurs, ils demandent désormais une application ergonomique, simple à utiliser, une rapidité dans les services proposés et un service client à l'écoute.

La conception d'une plateforme de livraison présente un intérêt car les utilisateurs ont besoin d'un système capable de faciliter les tâches de la vie quotidienne. De plus, ce service ne doit pas se limiter à l'achat de produits essentiels ou des plats préparés mais peut s'élargir à la livraison de colis personnels et donner une possibilité de commander des produits non listés dans la plateforme.

Notre travail consiste à concevoir et développer une application mobile sous Android qui permettra aux utilisateurs de passer une commande dans laquelle tout produit désiré par l'utilisateur sera facile à livrer et de simplifier la recherche d'un livreur. Notre choix s'est porté sur ce type d'applications car elle est plus accessible aux utilisateurs et à tout éventuel client.

Ce mémoire est organisé en 4 chapitre :

Le premier chapitre est consacré à la présentation de l'entreprise d'accueil, à l'étude de l'existant ainsi qu'à la méthode de conception suivie pour la réalisation du système. Ensuite, dans le second chapitre nous allons aborder la spécification et l'analyse des besoins du projet, suivi par le troisième chapitre de conception dans lequel les résultats précédents seront exploités pour détailler le fonctionnement du système. Enfin, le quatrième chapitre est dédié à la réalisation de l'application où nous allons présenter l'environnement de développement, les outils et langages de programmation utilisés dans notre travail, puis montrer quelques interfaces graphiques de notre application réalisée. Nous conclurons notre travail par une conclusion générale et quelques perspectives.

Chapitre 1

Présentation de l'entreprise d'accueil

1.1 Introduction

Dans ce chapitre, nous exposons le contexte du projet et son objectif, pour ensuite faire une étude de l'existant et proposer notre solution, nous allons également présenter la méthode de conception suivie dans ce projet.

1.2 Contexte et objectif

Le présent projet s'intitule conception et réalisation d'une application mobile de livraison, a été proposé par l'Agence Web Techsys Media, dans l'objectif d'offrir au grand public une application simple qui facilitera la livraison des commandes, offrant ainsi un service qui fera gagner du temps au citoyen n'ayant plus besoin de se déplacer. C'est aussi un moyen pratique pour assurer la distanciation sociale dû au covid-19.

1.3 Présentation de l'entreprise d'accueil

L'Agence Web Techsys Media est une entreprise orientée web, crée en 2016 située à Bejaia, qui offre une large palette de services professionnels, et développement de solutions techniques sur mesure.

Techsys Media est spécialisée dans la création de sites internet, hébergement web et développement d'applications (PC, Mobile).

Le logo de l'entreprise est présenté dans la figure 1.1.

FIGURE 1.1 – Logo de Techsys media

1.4 Etude de l'existant

L'étude de l'existant est une étape importante pour la réalisation d'un projet, car elle nous permettra d'améliorer un produit existant ou bien d'en créer un.
Nous retrouvons sur le marché plusieurs applications qui offrent le service de livraison, nous avons choisi d'étudier les applications les plus connues au niveau national et international.

1.4.1 Description des solutions disponibles

Les applications de livraison à la demande peuvent être classées selon le type de produit ou service qu'elles offrent comme suit :

1. Livraison de nourriture et de produits essentiels : Ce type d'applications se focalise sur la livraison des produits alimentaires.

    - **Yassir express** : application algérienne de livraison de nourriture, cosmétiques, fruits et légumes [1].

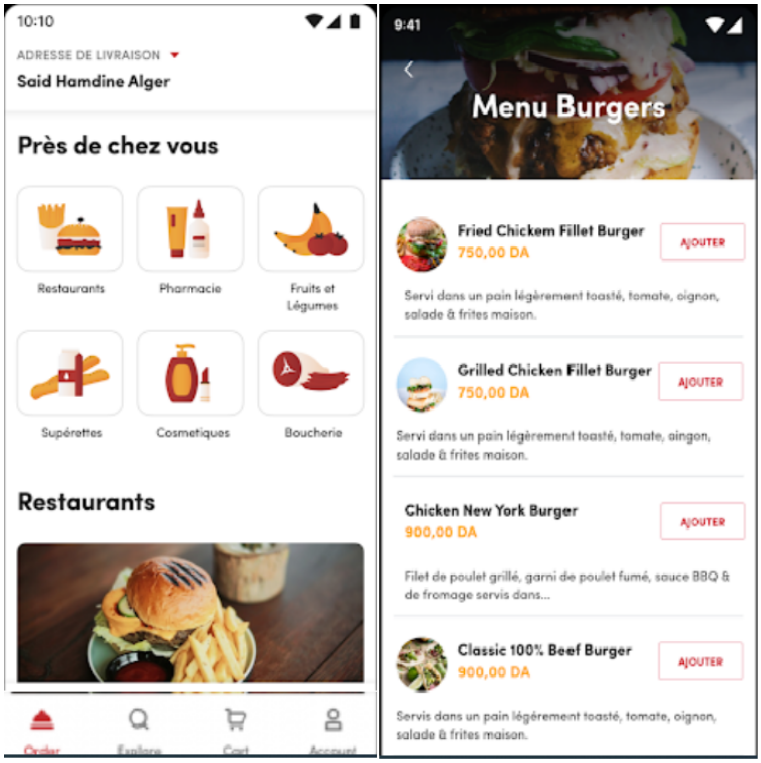

FIGURE 1.2 – Interfaces de Yassir express

- **Jumia food** : application de livraison à domicile de repas, boissons, desserts, produits de supermarché et pharmacie, disponible en Algérie [2].

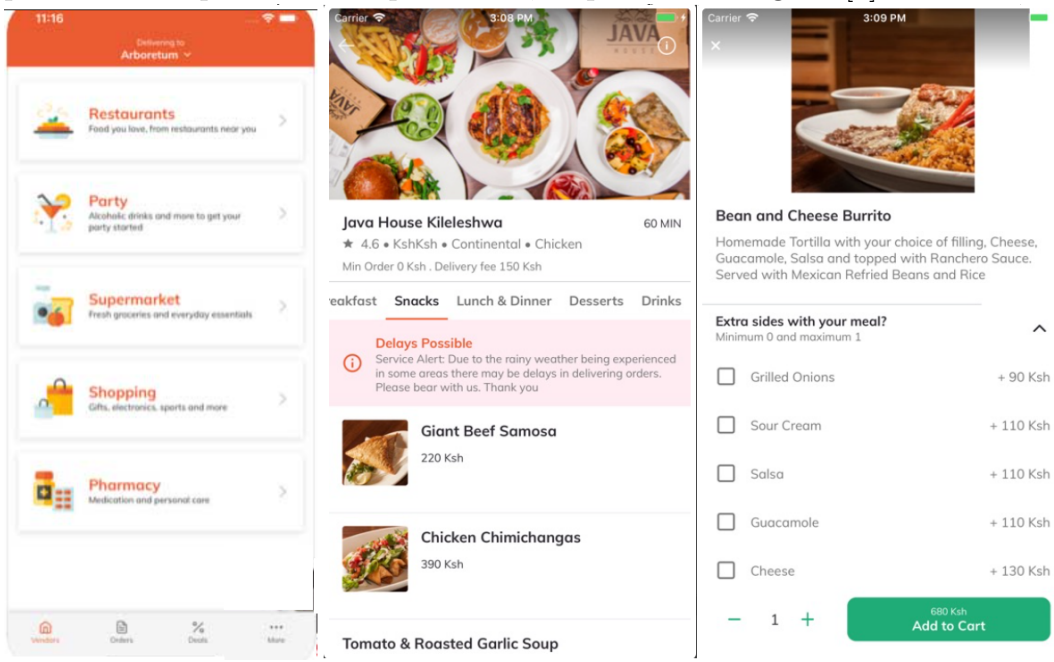

FIGURE 1.3 – Interfaces de Jumia food

2. Livraison de produits divers : cette catégorie propose un large choix d'articles de l'électronique, sport, vêtements, etc.

- **Yassir market** : est une plateforme algérienne d'achat en ligne avec possibilité de paiement à la livraison ou par carte dahabia [3].

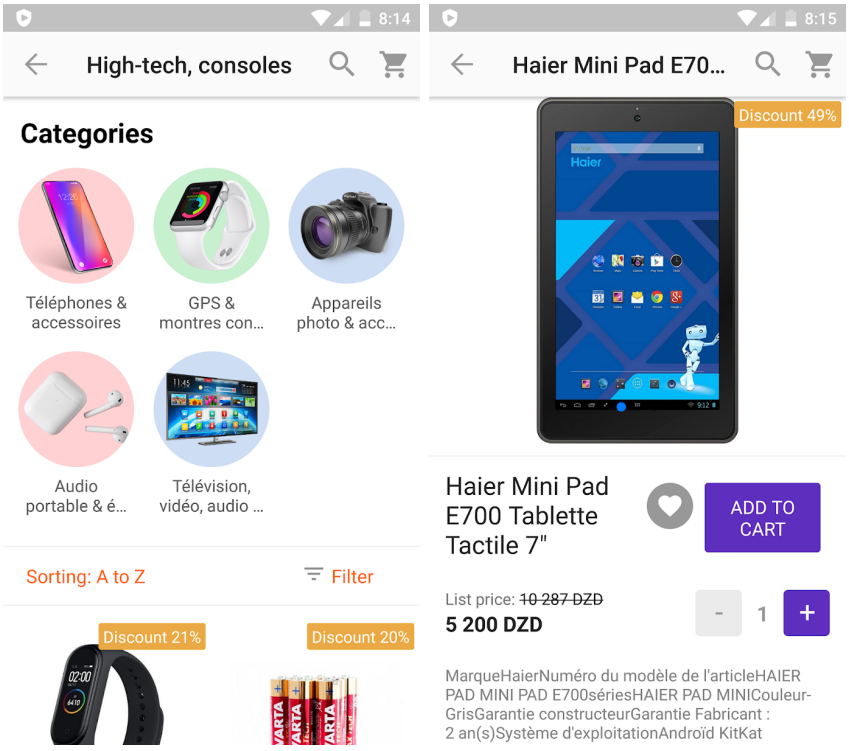

FIGURE 1.4 – Interfaces de Yassir market

- **Jumia achat en ligne** : est une plateforme d'achat en ligne disponible en Algérie, et dans plusieurs pays en Afrique [4].

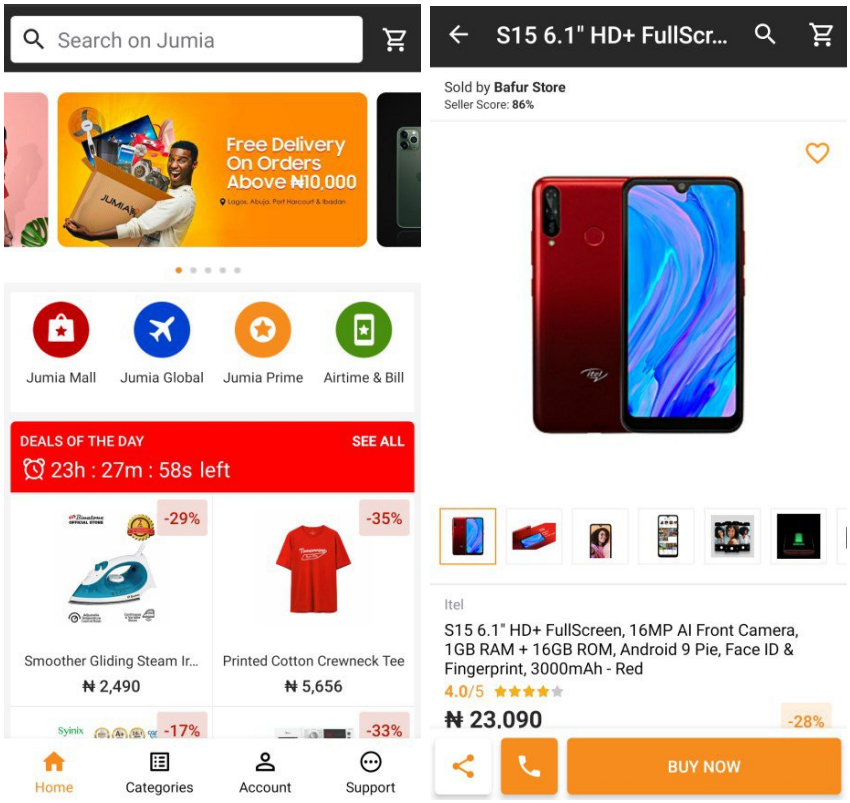

FIGURE 1.5 – Interfaces de Jumia achat en ligne

3. Livraison ouverte à toutes les commandes : ce type d'applications étend les possibilités de livraison, en effet, cette catégorie permet aux utilisateurs de :
   - Organiser la livraison d'un article personnel d'une adresse A à une adresse B (par exemple, récupérer un colis dans une boutique ou une commande auprès d'un restaurant).

   - Demander au livreur d'accomplir une tâche puis la livrer (par exemple, faire ses courses au supermarché).

   - **Mrsool** : application saoudienne de livraison, disponible dans le moyen orient, l'option de commander n'importe quel article (order anything) permet la livraison d'un article spécifié par l'utilisateur [5].

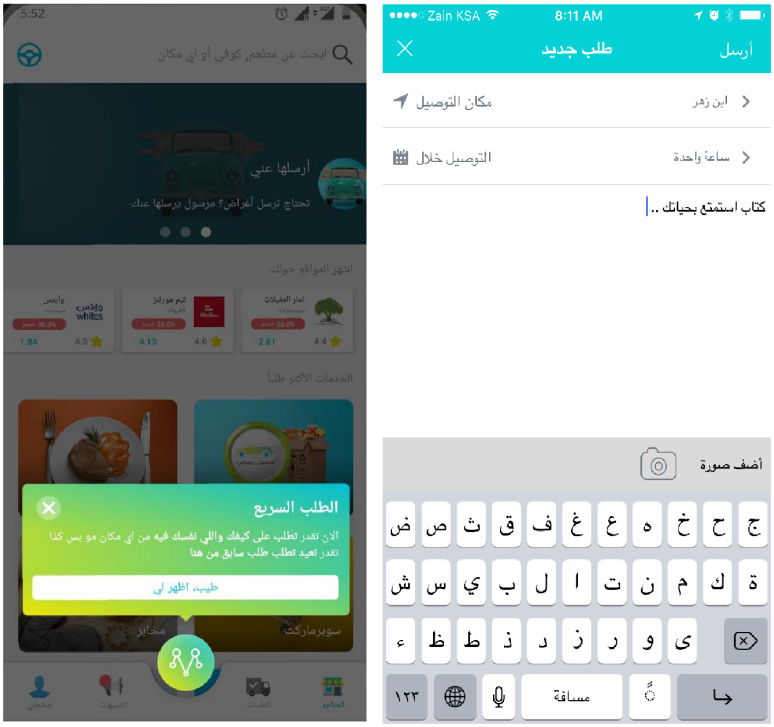

FIGURE 1.6 – Commande order anything Mrsool

- **Glovo** : application espagnole disponible en Europe et dans d'autres pays, offre la livraison de produits essentiels et de tous les articles partout grâce à l'option order anything [6].

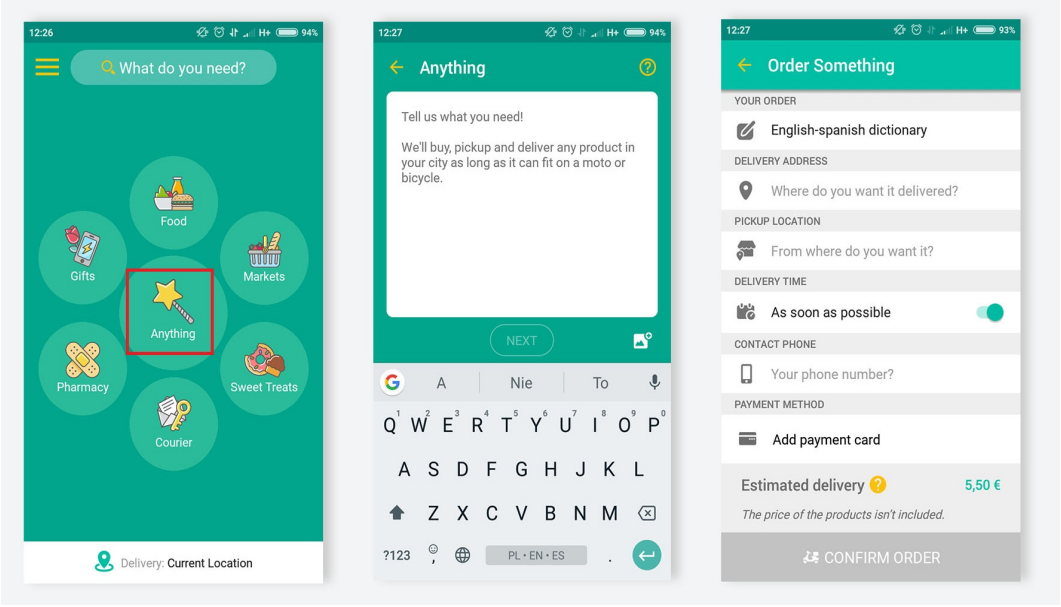

FIGURE 1.7 – Commande order anything Glovo

- **Uber** : application américaine de mise en contact d'utilisateurs avec des conducteurs réalisant des services de transport. Elle offre également une option de transport d'objet grâce à Uber Connect [7].

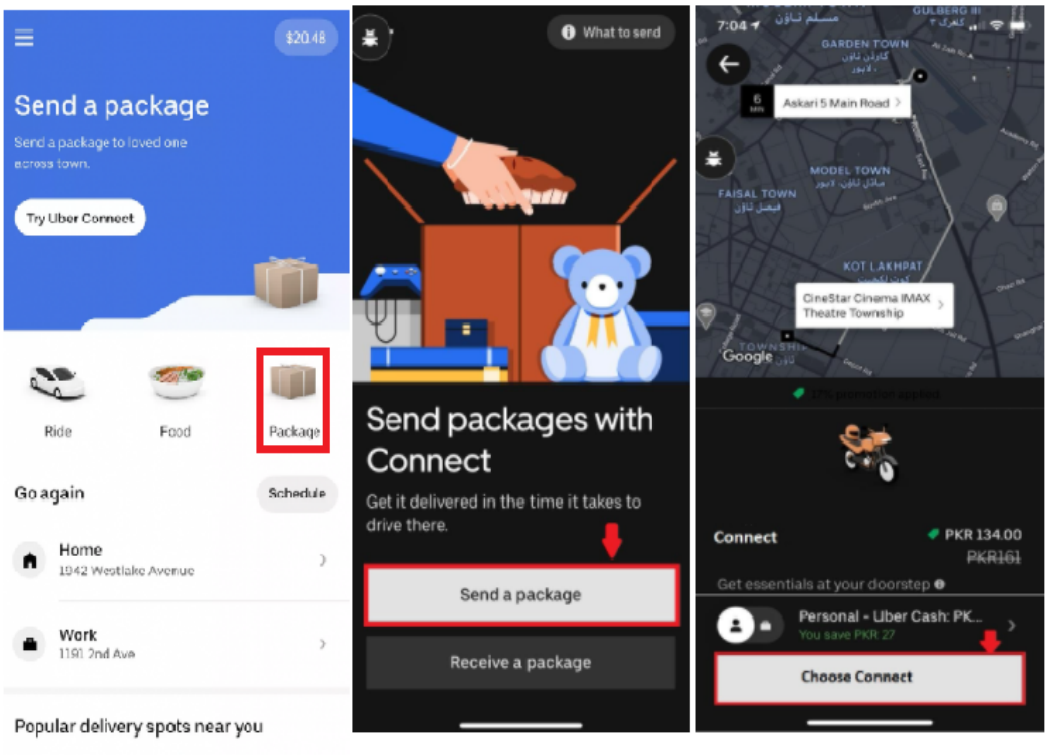

FIGURE 1.8 – Uber connect envoi d'un objet

1.4.2 Critique de l'existant

Dans cette section, nous étudions les avantages et inconvénients des applications cités précédemment, suivant leur classification :

1. Livraison de nourriture et de produits essentiels :

Application	Avantages	Inconvénients
Yassir Express	• Large choix de produits. • e-paiement disponible. • facile à utiliser.	• Indisponible dans certaines wilaya d'Algérie. • disponiblité des restaurants n'est pas en temps réel. • affectation des livreurs est lente.
Jumia food	• Différents modes de paiement disponibles. • suivi de la commande en temps réel. • ergonomique et facile à utiliser.	• Indisponible dans certaines wilaya d'Algérie. • liste réduite de produits à livrer. • contact et informations du livreur indisponibles.

Tableau 1.1 – Comparaison entre applications Yassir express et Jumia food

2. Livraison de produits divers :

Application	Avantages	Inconvénients
Yassir Market	• E-paiement disponible. • facile à utiliser. • livraison au niveau national.	• Suivi de la livraison indisponible. • évaluation de la livraison indisponible. • frais de livraison chers.

Jumia achat en ligne	• E-paiement disponible. • un large choix de produits. • évalution des produits est disponible.	• Frais de livraison chers. • réclamation difficile. • description des produits incomplete.

Tableau 1.2 – Comparaison entre applications Yassir market et Jumia achat en ligne

3. Livraison ouverte à toutes les commandes :

Cette catégorie d'applications résout le problème de manque de produits et de transport d'articles personnels en offrant une seule plateforme qui rassemble tous ces services. Cependant, ce type n'est pas encore disponible en Algérie.

1.5 Solution proposée

Après une étude et analyse des applications existantes, notre application doit prendre en considération les avantages et inconvénients constatés précédemment, également l'application vise à offrir une bonne qualité de services et une fluidité dans son utilisation. Notre solution consiste à développer une application mobile capable de :

- Prendre en charge les commandes de type livraison seule, et le type achat et livraison.
- Localiser les adresses de départ et de destination sur une carte géographique.
- Faciliter le contact entre l'utilisateur et le livreur.
- Evaluer le service des livreurs.

1.6 Méthode de conception

Une méthode d'analyse et de conception est un procédé qui a pour objectif de permettre de formaliser les étapes préliminaires du développement d'un système afin de rendre ce développement plus fidèle aux besoins du client [8].

1.6.1 Processus de développement

Un processus définit une séquence d'étapes, en partie ordonnées, qui concourent à l'obtention d'un système ou à l'évaluation d'un système existant.
L'objet d'un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles [31].

1.6.2 Processus unifié (UP)

Le processus unifié (UP) est une famille de méthodes de développement de logiciels orientés objets. Elle se caractérise par une démarche itérative et incrémentale, pilotée par les cas d'utilisation, et centrée sur l'architecture et les modèles UML. Elle définit un processus intégrant toutes les activités de conception et de réalisation au sein de cycles de développement composés d'une phase de création, d'une phase d'élaboration, d'une phase de construction et d'une phase de transition, comprenant chacune plusieurs itérations [30].

Le processus unifié se caractérise par les propriétés suivantes [26] :
- **Itératif et incrémental** : le projet est découpé en itérations de courte durée (environ 1 mois) qui aident à mieux suivre l'avancement global. A la fin de chaque itération, une partie exécutable du système final est produite, de façon incrémentale.
- **Centré sur l'architecture** : tout système complexe doit être décomposé en parties modulaires afin de garantir une maintenance et une évolution facilitées. Cette architecture (fonctionnelle, logique, matérielle, etc.) doit être modélisée en UML et pas seulement documentée en texte.
- **Piloté par les risques** : les risques majeurs du projet doivent être identifiés au plus tôt, mais surtout levés le plus rapidement possible. Les mesures à prendre dans ce cadre déterminent l'ordre des itérations.
- **Conduit par les cas d'utilisation** : le projet est mené en tenant compte des besoins et des exigences des utilisateurs. Les cas d'utilisation du futur système sont identifiés, décrits avec précision et priorisés.

Le processus unifié est défini par les phases du cycle de vie comme suit [26] :
- **La phase d'initialisation** : conduit à définir la vision du projet, sa portée, sa faisabilité, son business case, afin de pouvoir décider au mieux de sa poursuite ou de son arrêt.
- **La phase d'élaboration** : poursuit trois objectifs principaux en parallèle : Identifier et décrire la majeur partie des besoins des utilisateurs, construire l'architecture de base du système, et enlever les risques majeurs du projet.

- **La phase de construction** : consiste surtout 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.
- **La phase de transition** : permet de faire passer le système informatique des mains des développeurs à celles des utilisateurs finaux.

Le processus unifié propose les enchainements d'activités suivants :

- **Exigences** : présente le système du point de vue de l'utilisateur et recence 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, dynamique et statique) d'une architecture, à travers, un diagramme de classe et d'interaction, etc.
- **Implémentation** : implémentation des résulats de conception, intégration du système et des classes.
- **Test** : planification et mise en œuvre des tests.

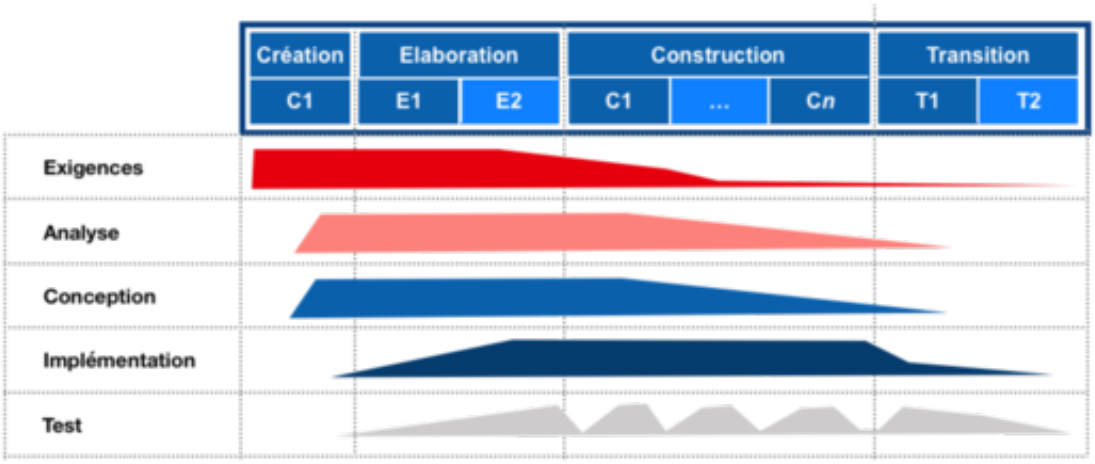

FIGURE 1.9 – Processus unifié

1.6.3 Langage de modélisation UML

Le processus unifié est un processus entièrement basé sur UML. Ce dernier se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue [31].

1.6.4 Types de diagrammes UML

UML définit 14 types de diagrammes divisés en deux catégories, les diagrammes structurels représentant une vue statique du système, et les diagrammes comportementaux concernant le comportement dynamique du système [31].

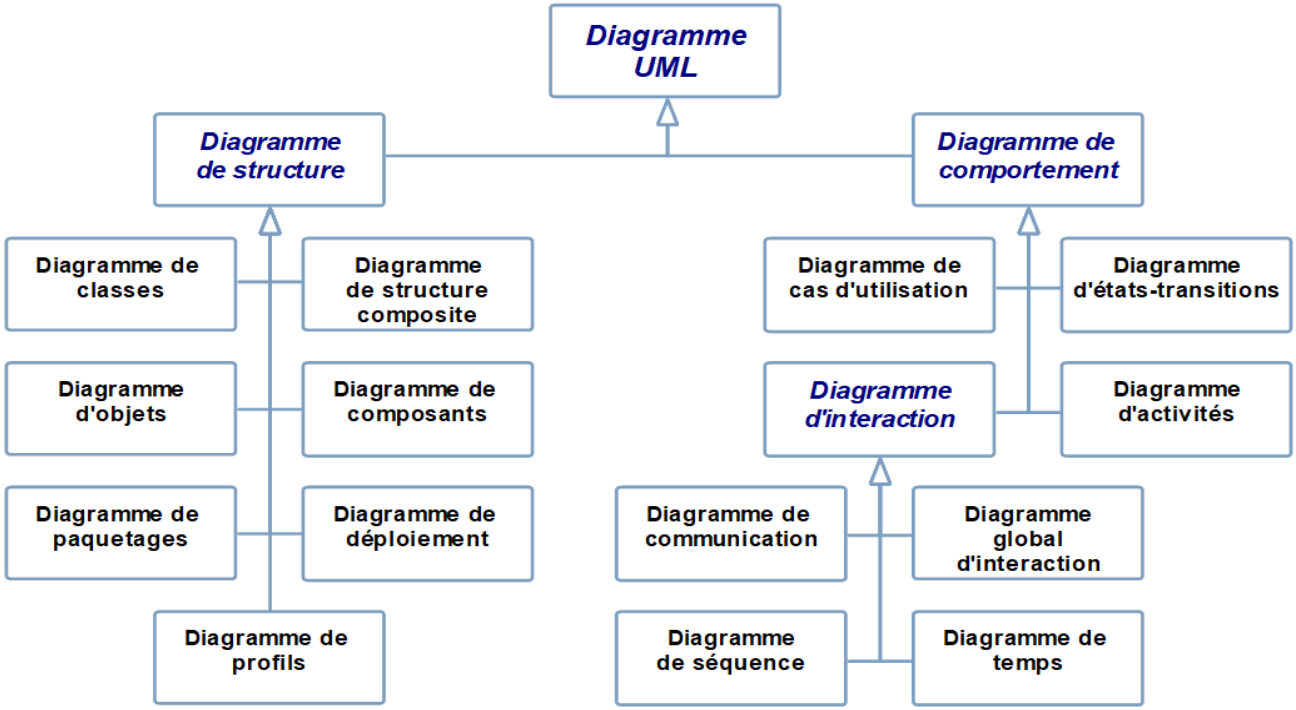

FIGURE 1.10 – Types de diagrammes UML

Dans notre projet nous allons utiliser seulement les diagrammes suivants dont la définition est définie dans [31] :

- **Diagramme de cas d'utilisation** : représente la structure des fonctionnalités nécessaires aux utilisateurs du système. Il est utilisé dans les deux étapes de capture des besoins fonctionnels et techniques.
- **Diagramme de séquence** : est un diagramme d'interaction, il représente les échanges de messages entre objets, dans le cadre d'un fonctionnement particulier du système. Ils servent ensuite à développer en analyse les scénarios d'utilisation du système.
- **Diagramme de classe** : a toujours été le plus important dans toutes les méthodes orientées objet. C'est également celui qui contient la plus grande gamme de notations

et de variantes centralise l'organisation des classes de conception, c'est lui qui se transforme le plus aisément en code.

1.7 Conclusion

Dans ce chapitre, nous avons décrit le contexte général du projet, ensuite nous avons étudié les applications similaires à notre projet pour mieux répondre aux besoins des utilisateurs.
Ainsi, nous avons défini le cadre du projet, ce qui nous permet d'entamer la prochaine étape qui consiste en la phase de spécification et analyse des besoins.

Chapitre 2

Spécification et Analyse des besoins

2.1 Introduction

Dans ce chapitre, nous allons identifier les fonctionnalités de l'application selon les exigences fonctionnelles et non fonctionnelles pour pouvoir entamer l'analyse des besoins par la description des acteurs et leurs interactions avec le système grâce au diagramme de contexte dynamique, les cas d'utilisation seront accompagnés d'une description détaillée et des diagrammes de séquence.

2.2 Spécification des besoins

La spécification des besoins nous donne une vision globale sur le projet, à travers l'identification des fonctionnalités principales à implémenter.

2.2.1 Besoins fonctionnels

Les besoins fonctionnels correspondent aux fonctionnalités que notre application doit offrir. Elles se résument en les fonctionnalités suivantes :

- Inscription à l'application : l'utilisation de l'application nécessite une inscription dans laquelle les informations de l'utilisateur vont être enregistrés.
- Gestion de comptes : les informations et l'état d'un compte utilisateur sont modifiables.
- Planification d'une livraison : l'application doit offrir la possibilité de créer une commande et de pouvoir déclencher ou retarder (activer ou suspendre) son lancement.

- Consultation des activités de livraison : les informations liées à une livraison peuvent être consultées.
- Gestion des commandes : l'utilisateur peut gérer l'état de la commande et son déroulement.

2.2.2 Besoins non fonctionnels

Les besoins non fonctionnels ont un aspect visible pour l'utilisateur, mais ne sont pas en rapport direct avec le comportement du système. Ce sont les contraintes techniques du système et les besoins qui le caractérisent. Les exigences qu'on doit satisfaire sont :

- L'ergonomie et l'adaptabilité des interfaces aux différents appareils mobiles : l'application doit offrir une interface conviviale, explicite et simple à utiliser.
- Connexion Internet : les services offerts par l'application sont accessibles via internet.
- La disponibilité du service : le service de livraison doit être disponible.
- Utilisabilité de l'application : l'application doit être facile à utiliser et offrir une bonne qualité d'expérience utilisateur.
- L'extensibilité de l'application afin de pouvoir introduire d'autres fonctionnalités.
- La sécurité : l'accès à l'application exige une authentification de l'utilisateur.
- Temps de réponse : l'application doit réagir en espace de quelques fractions de secondes.

2.3 Analyse des besoins

L'analyse des besoins permet à partir des besoins fonctionnels et non fonctionnels, de définir l'ensemble des cas d'utilisation du futur système respectant avec précision les exigences exprimés par le client.

2.3.1 Identification des acteurs

Un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, matériel ou autre système) qui interagissent directement avec le système étudié [31].
Notre application contient 3 acteurs principaux qui agissent directement avec le système :

- Client : représente l'utilisateur demandant la livraison d'une commande.
- Livreur : représente la personne s'occupant de la livraison et parfois l'achât d'une commande.
- Administrateur : représente la partie qui s'occupe de la gestion des comptes.

2.3.2 Modélisation du contexte

Le tableau suivant illustre les différents messages échangés entre les acteurs et le système.

ID Messages Acteur – Système	ID Messages Système – Acteur
M1 Demande d'authentification	M1' Interface Authentification
M2 Demande d'inscription	M2' Interface Inscription
M3 Planifier une livraison	M3' Interface de création de commande
M4 Gérer compte	M4' Interface de gestion de compte
M5 Consulter activités de livraison	M5' Interface détail des commandes
M6 Gérer les commandes	M6' Interface de gestion de commande

Tableau 2.1 – Messages échangés entre les acteurs et le système

Tous les messages (Système–Acteurs) identifiés précédemment peuvent être représentés de façon synthétique sur un diagramme appelé diagramme de contexte.
Le diagramme suivant représente les différents acteurs du système, ainsi que les messages échangés entre ces derniers.

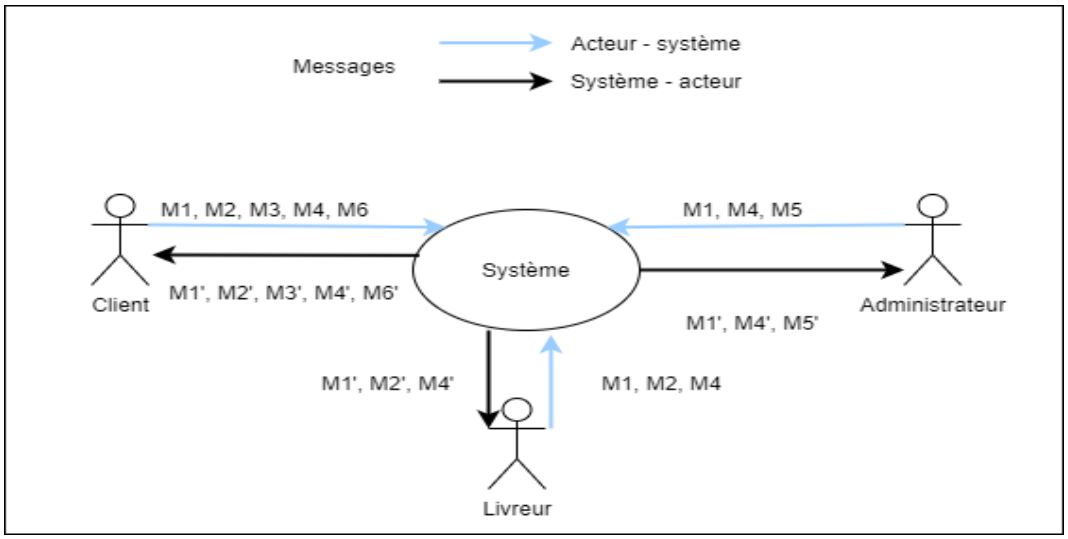

FIGURE 2.1 – Diagramme de contexte dynamique

2.3.3 Identification des cas d'utilisation

Un cas d'utilisation (use case) représente un ensemble de séquences d'actions réalisées par le système et produisant un résultat observable intéressant pour un acteur particulier [31].
Pour chaque acteur identifié précédemment on associe les cas d'utilisation qui lui correspondent.
Le tableau suivant liste les différents cas d'utilisation associés à notre système.

N°	Cas d'utilisation	Acteur
1	S'inscrire	Client, livreur
2	S'authentifier	Client, livreur, administrateur
3	Gérer son compte	Client, livreur
4	Gérer compte Client et Livreur (activer, suspendre, supprimer)	Administrateur
5	Consulter les activités de livraison	Administrateur
6	Créer une commande	Client
7	Accepter ou refuser un livreur	Client
8	Evaluer le service du livreur	Client
9	Modifier état de commande (activer, suspendre, supprimer)	Client
10	Recevoir les commandes client	Livreur
11	Accepter ou refuser commande client	Livreur
12	Confirmer la livraison de commande	Client, livreur
13	Réclamer une commande	Client, livreur
14	Recevoir réclamation	Administrateur
15	Consulter commande	Client, livreur

Tableau 2.2 – Cas d'utilisation associés au système

2.3.4 Description des cas d'utilisation

Dans cette partie nous allons présenter une description textuelle des cas d'utilisation les plus important.

- **Description du cas d'utilisation** "S'authentifier" :

Sommaire	
Titre	S'authentifier
Résumé	Authentifier les utilisateurs connectés
Acteurs	Client, livreur
Description des scénarios	
Pré conditions	Application accessible
Scénario	1. L'utilisateur saisi email ou numéro de téléphone et mot de passe puis valide 2. le système vérifie la saisie [**A1**] 3. le système vérifie la validité des données [**A2**] 4. le système renvoie l'interface correspondante
Enchainement d'erreurs	• [**A1**] : Erreur de saisi (champs vide ou numéro/email ne sont pas valides) • [**A2**] : Données erronées (mot de passe incorrect ou utilisateur non inscrit)
Post conditions	Application prête à fonctionner

Tableau 2.3 – Description du cas d'utilisation s'authentifier

- **Description du cas d'utilisation** "Créer une commande de type livraison seule" :

Sommaire	
Titre	Créer une commande option livraison seule
Résumé	Création d'une commande client de livraison
Acteurs	Client
Description des scénarios	
Pré conditions	Utilisateur authentifié
Scénario	1. L'utilisateur choisit le service de livraison seule 2. le système renvoie le formulaire correspondant 3. l'utilisateur introduit les adresses départ et destination [**A1**] 4. le système génère le prix de livraison selon la distance 5. l'utilisateur décrit le produit à livrer et valide [**A2**] 6. le système confirme la création de la commande
Enchainement d'erreurs	• [**A1**] : Adresse manquante ou aucune adresse sélectionnée • [**A2**] : Erreur de saisie (champs vides)
Post conditions	Commande créée

Tableau 2.4 – Description du cas d'utilisation livraison seule

- **Description du cas d'utilisation** "Créer une commande de type achât et livraison" :

Sommaire	
Titre	Créer une commande option achât et livraison
Résumé	Création d'une commande client de livraison
Acteurs	Client
Description des scénarios	
Pré conditions	Utilisateur authentifié
Scénario	1. L'utilisateur choisit le service achat et livraison 2. le système renvoie le formulaire correspondant 3. l'utilisateur introduit les adresses d'achat et destination [**A1**] 4. le système génère le prix de livraison selon la distance 5. l'utilisateur décrit le nom du produit, sa marque et sa quantité puis valide [**A2**] 6. le système confirme la création de la commande
Enchainement d'erreurs	• [**A1**] : Adresse manquante ou aucune adresse sélectionnée • [**A2**] : Erreur de saisie (champs vides)
Post conditions	Commande créée

Tableau 2.5 – Description du cas d'utilisation achât et livraison

2.3.5 Diagramme de cas d'utilisation par acteur

Dans cette section, il s'agit de résumer les cas d'utilisation par acteur, les figures suivantes correspondent aux diagrammes de cas d'utilisation respectifs aux acteurs.

a) Client

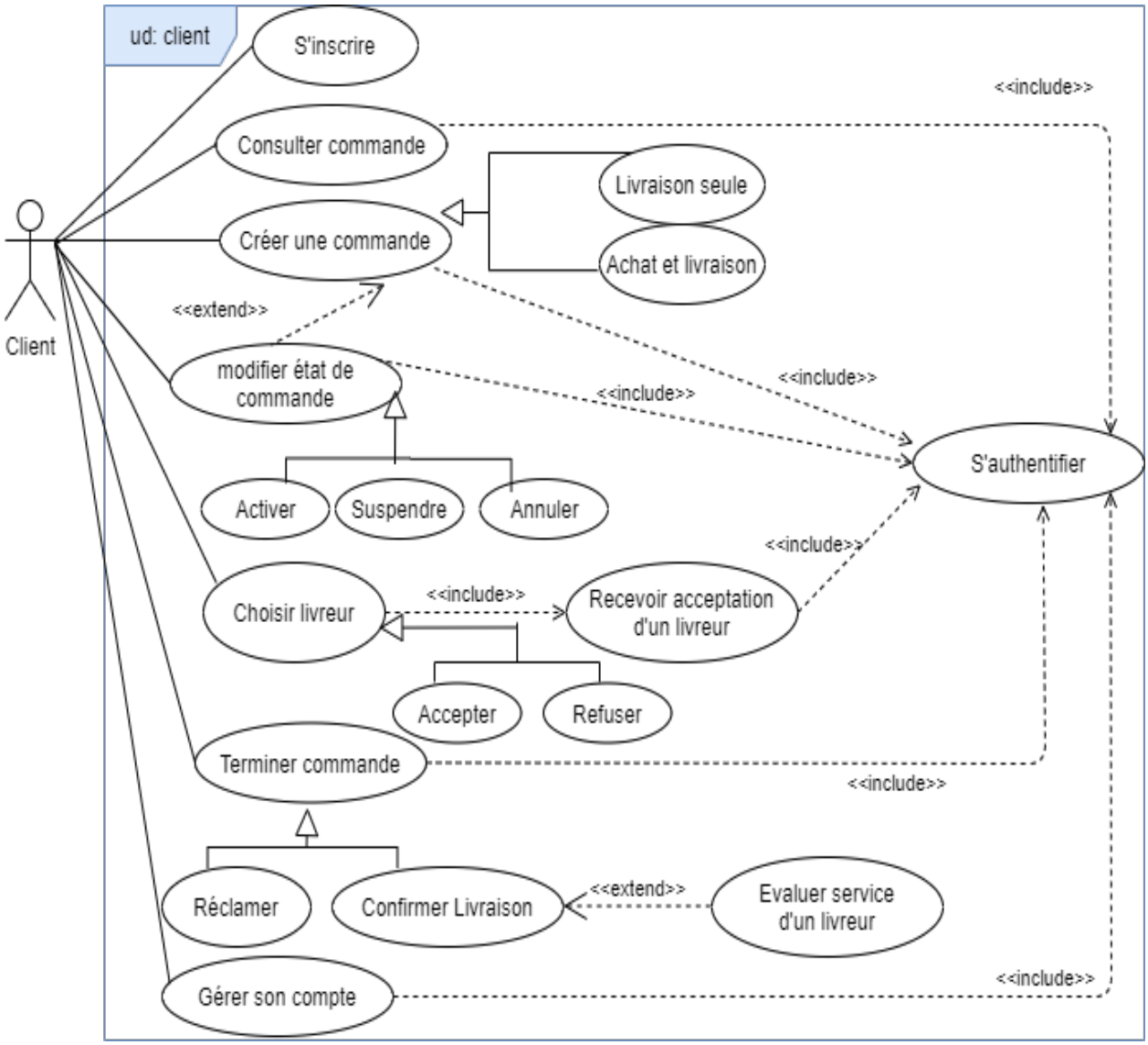

FIGURE 2.2 – Diagramme de cas d'utilisation client

b) Livreur

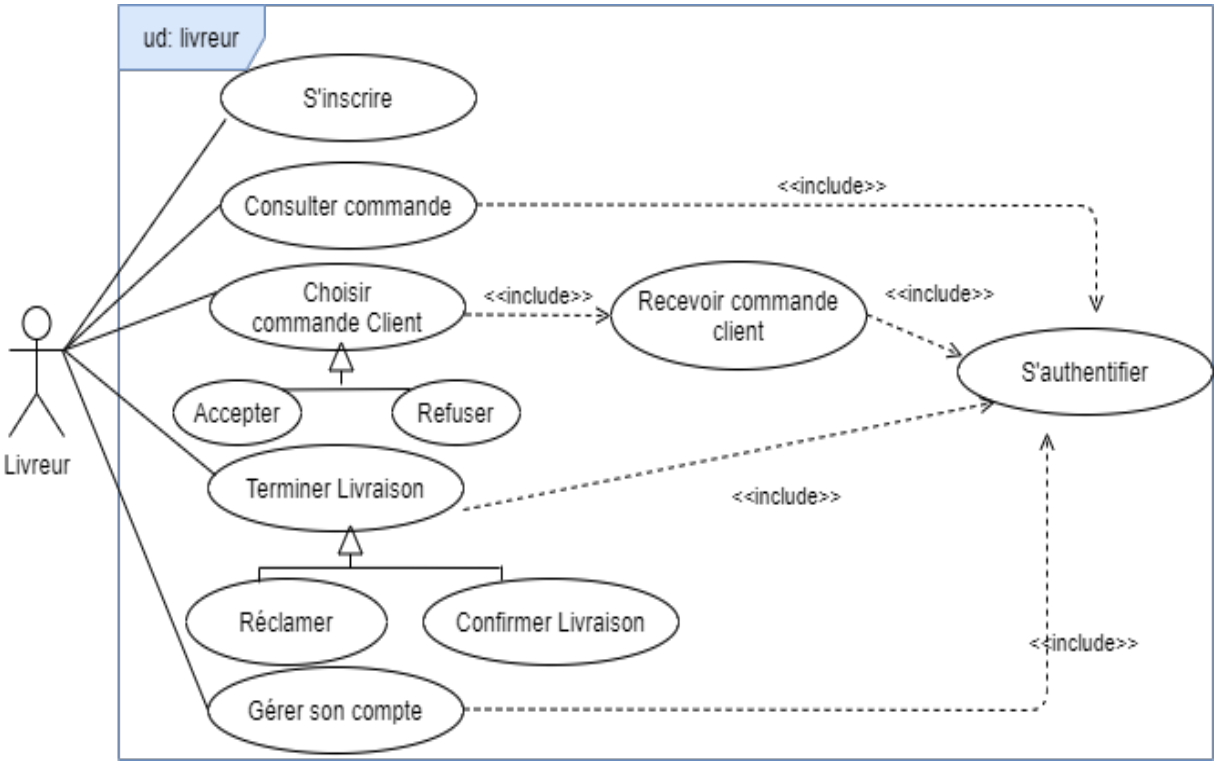

FIGURE 2.3 – Diagramme de cas d'utilisation livreur

c) Administrateur

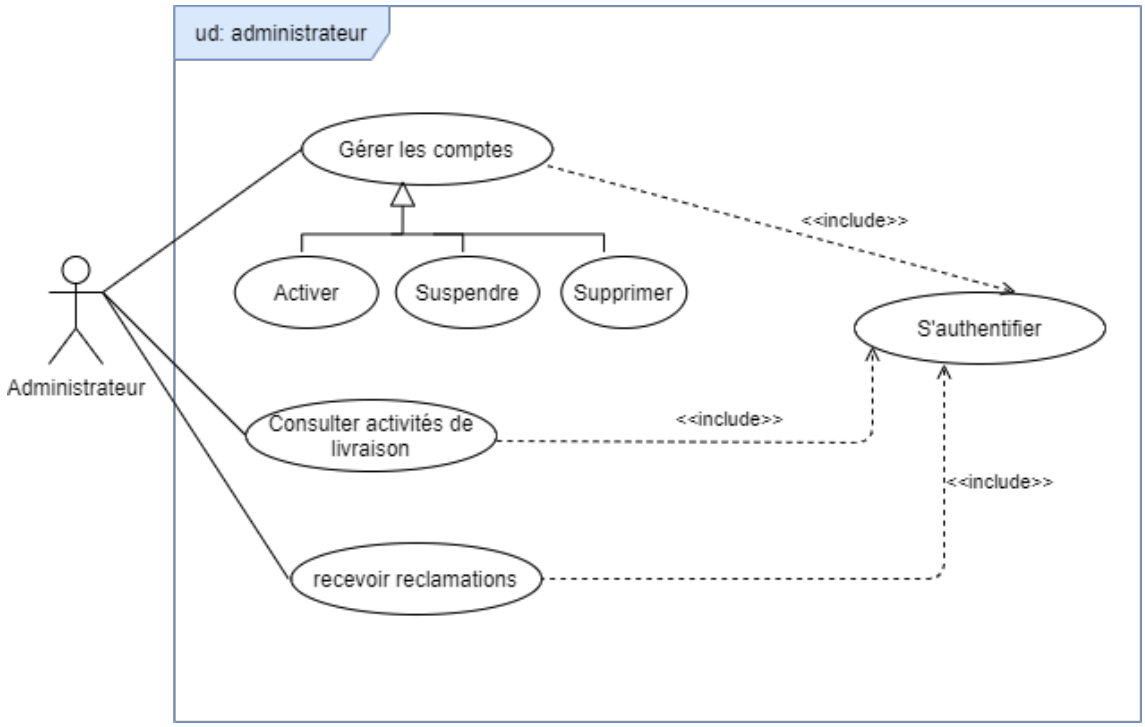

FIGURE 2.4 – Diagramme de cas d'utilisation administrateur

2.3.6 Diagramme de séquence système

La description textuelle ne permet pas de montrer les enchaînements des événements, le diagramme de séquence est nécessaire à cet effet pour compléter la description des cas d'utilisation précédents.

- **Description du cas d'utilisation** "S'authentifier" :

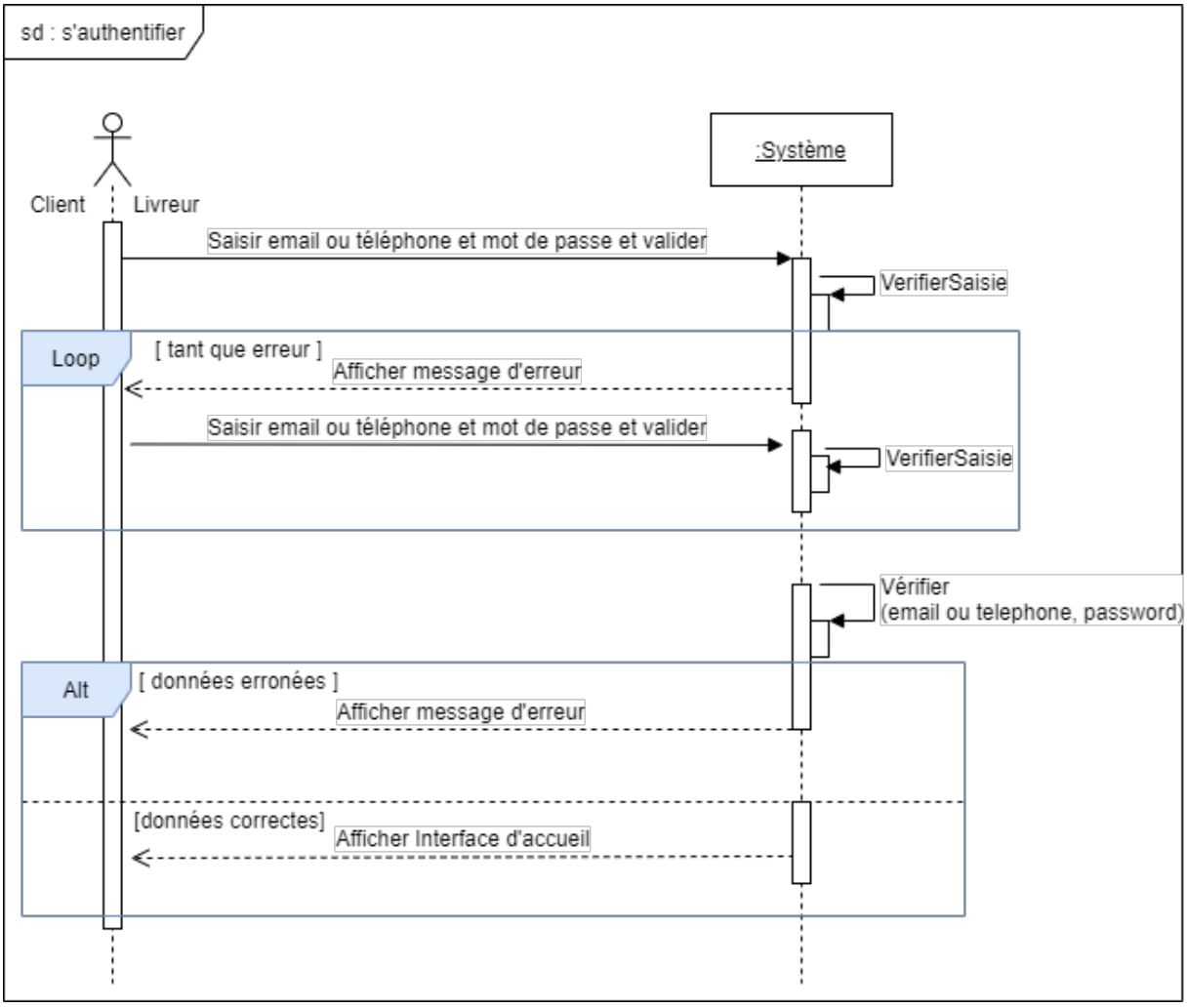

FIGURE 2.5 – Diagramme de séquence système "s'authentifier"

- **Description du cas d'utilisation** "Commande de livraison seule" :

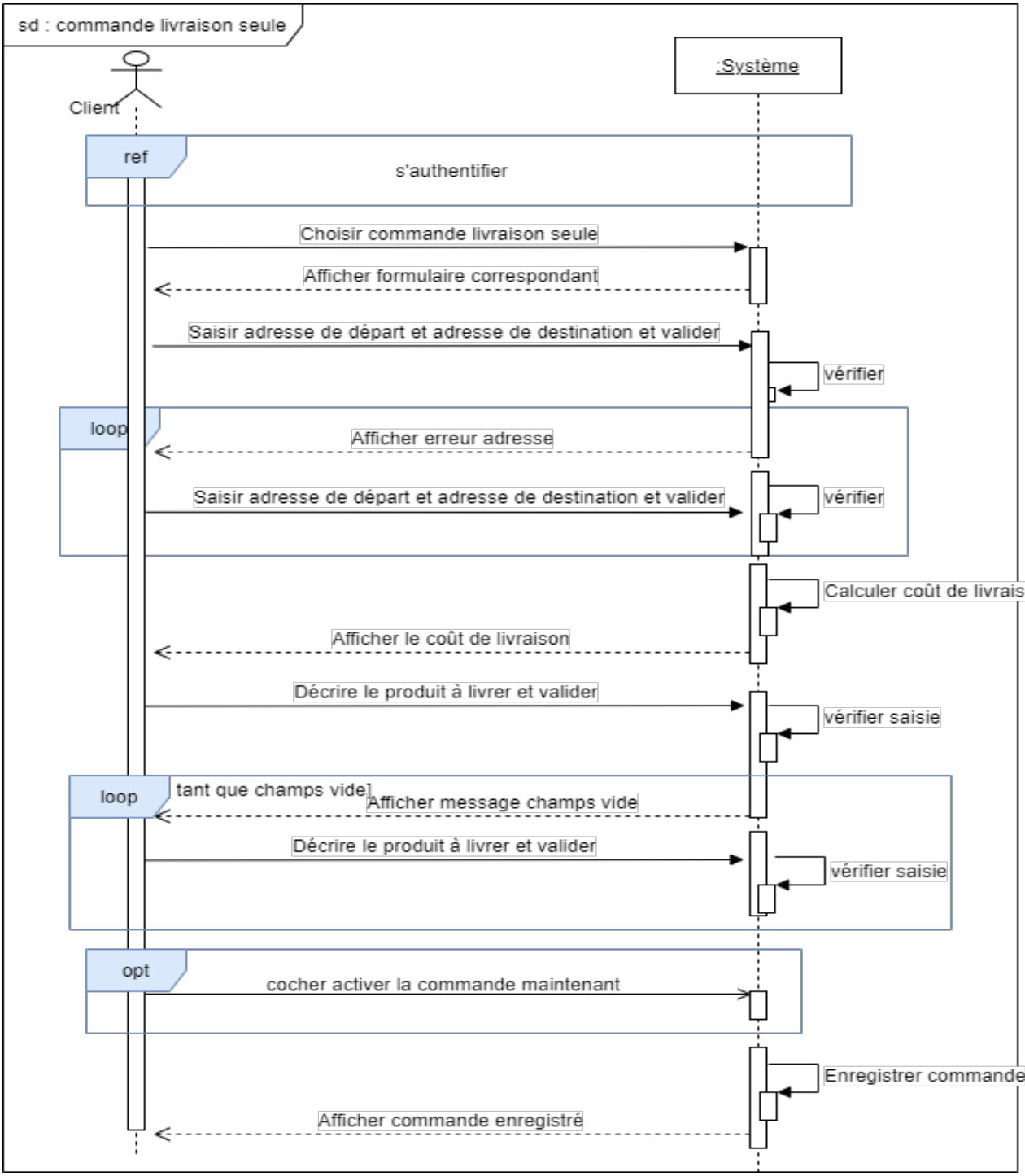

FIGURE 2.6 – Diagramme de séquence système "commande de livraison seule"

- **Description du cas d'utilisation** "Commande d'achât et livraison" :

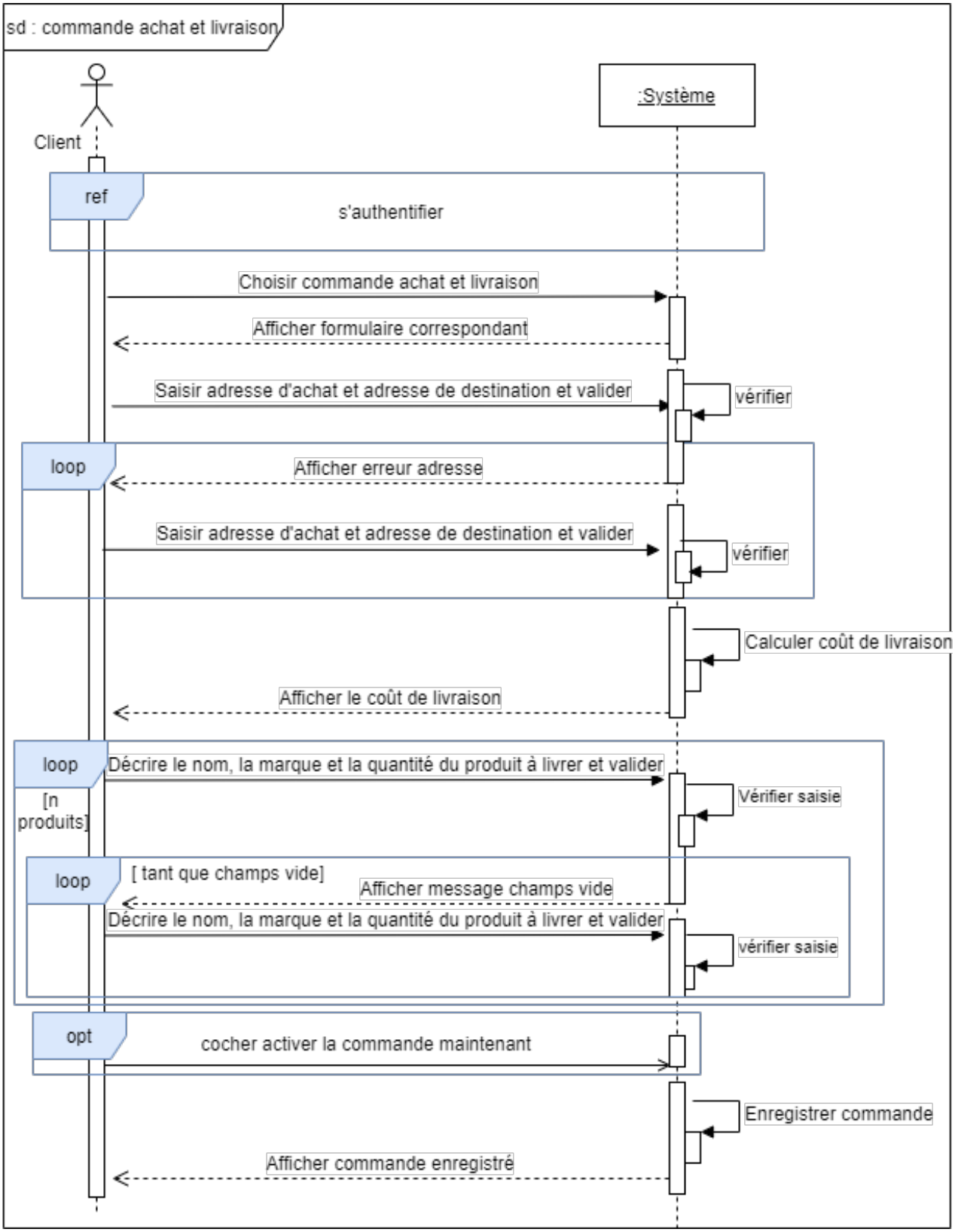

FIGURE 2.7 – Diagramme de séquence système "commande d'achât et livraison"

Des diagrammes de séquence relatifs à d'autres cas d'utilisation sont présentés dans l'annexe A.

2.4 Conclusion

A l'issu de cette étape, nous avons pu exprimer et décrire les objectifs et les besoins de notre système, ainsi que les exprimer sous forme de diagramme de cas d'utilisation et diagramme de séquence.

Dans le chapitre suivant, nous entamons la phase de conception dans laquelle nous décrirons d'une manière détaillée ces besoins.

Chapitre 3

Conception

3.1 Introduction

Dans ce chapitre nous allons établir, à partir des diagrammes de séquence système décrits dans le chapitre précédent, les diagrammes de séquence d'interaction dans lesquels le système est remplacé par les objets qui interviennent pour réaliser les différents cas d'utilisation. Par la suite, nous exploitons les objets entités pour la réalisation du diagramme de classe. Enfin, nous allons définir les bases de données non relationnelles et justifier le choix de cette solution dans notre projet.

3.2 Diagramme de séquence d'interaction

Pour chaque diagramme de séquence système, nous établirons le diagramme de séquence d'interaction correspondant.
Les objets composant le système sont de 3 types :

a) L'objet interface (dialogue) : représente l'interface entre l'acteur et le système.

b) l'objet contrôle : représente un traitement du système déclenché par un acteur.

c) l'objet entité : représente des objets décrits dans le cas d'utilisation.

- **Description du cas d'utilisation** "S'authentifier" :

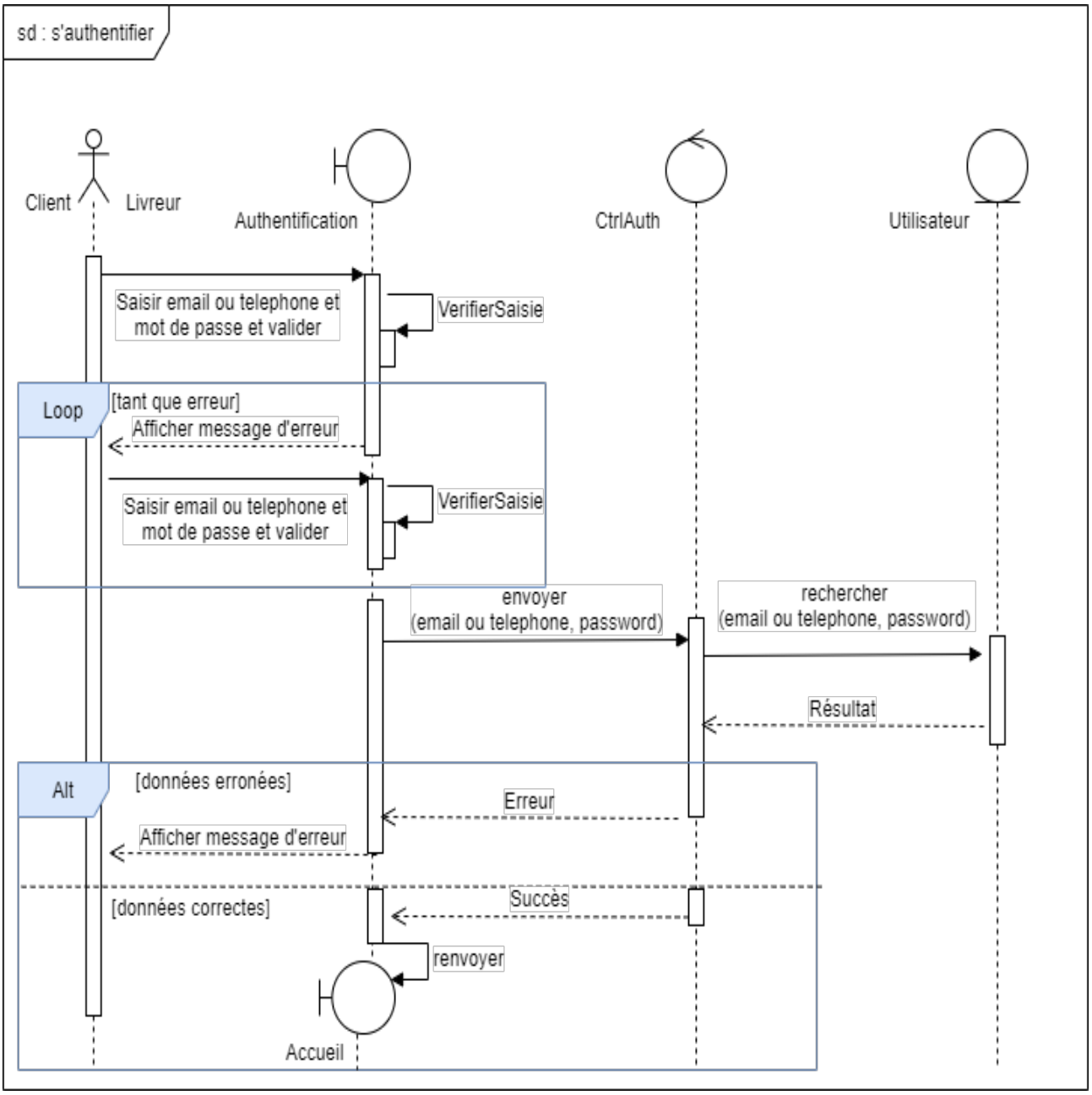

FIGURE 3.1 – Diagramme de séquence d'interaction "s'authentifier"

- **Description du cas d'utilisation** "commande de type livraison" :

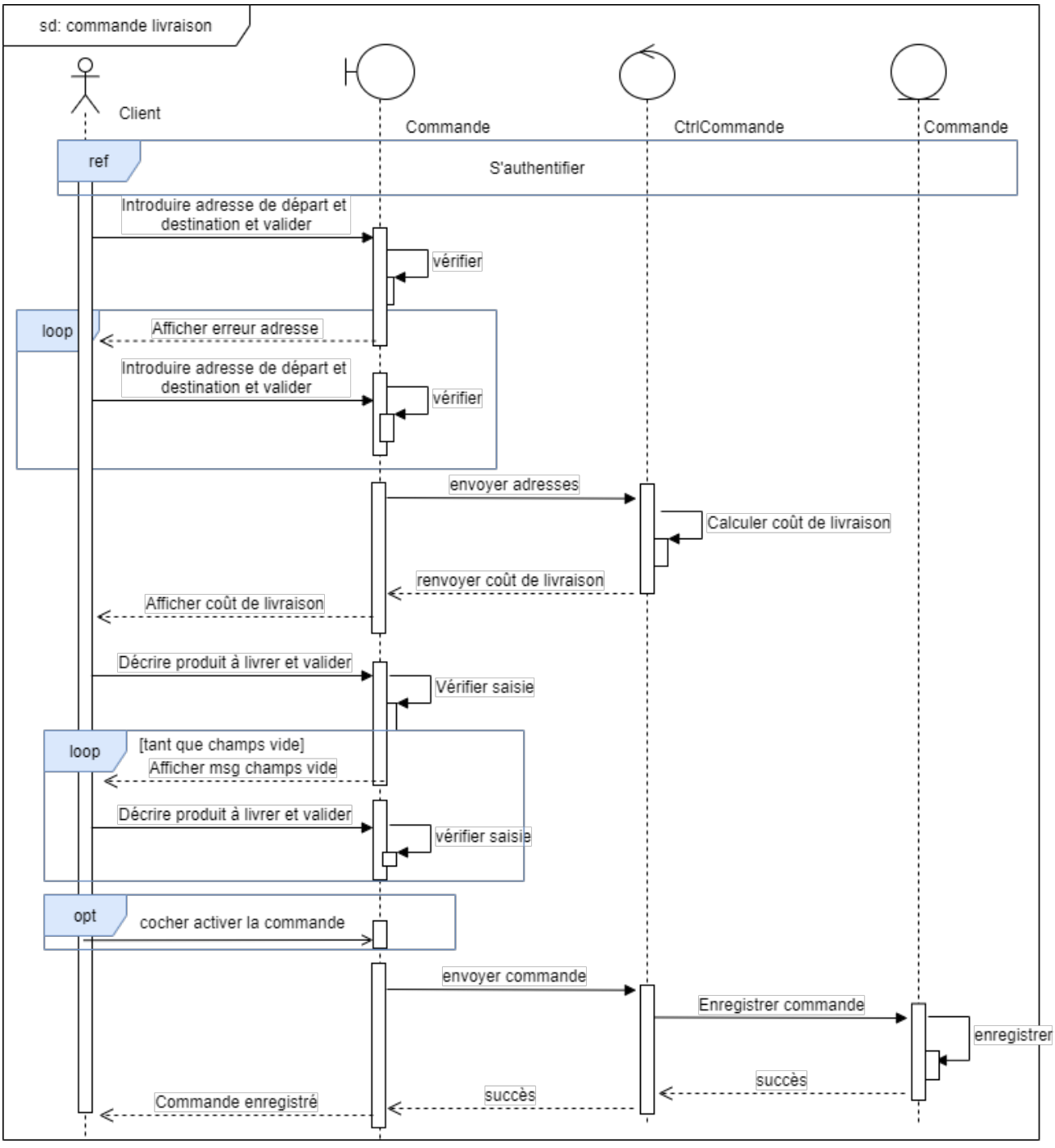

FIGURE 3.2 – Diagramme de séquence d'interaction "commande de livraison seule"

- **Description du cas d'utilisation** "commande de type achât et livraison" :

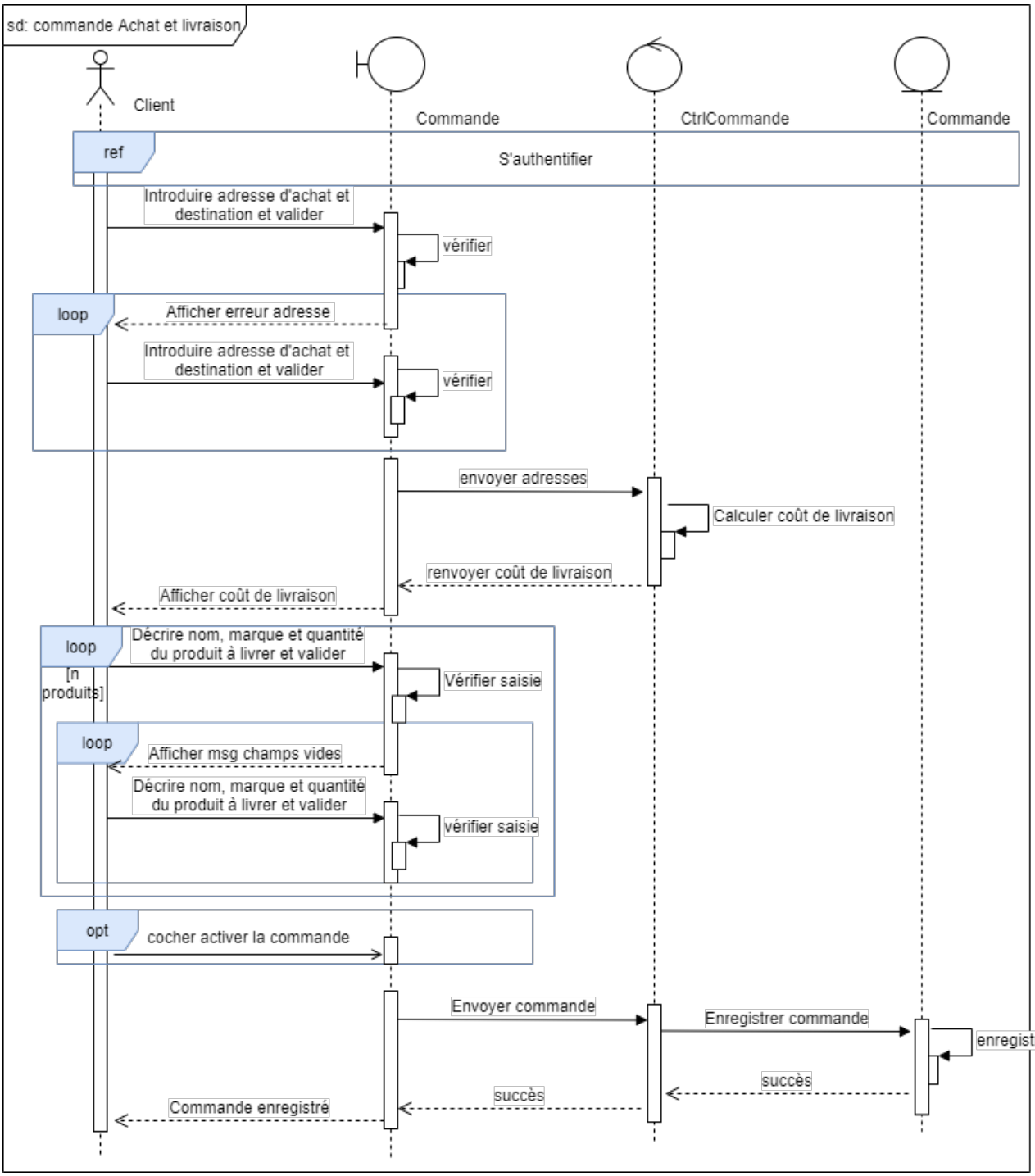

FIGURE 3.3 – Diagramme de séquence d'interaction "commande achât et livraison"

Des diagrammes de séquence relatifs à d'autres cas d'utilisation sont présentés dans l'annexe B.

3.3 Diagramme de classe de conception

Le diagramme de classes montre la structure interne du système. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. Il s'agit d'une vue statique, car on ne tient pas compte du facteur temporel dans le comportement du système.

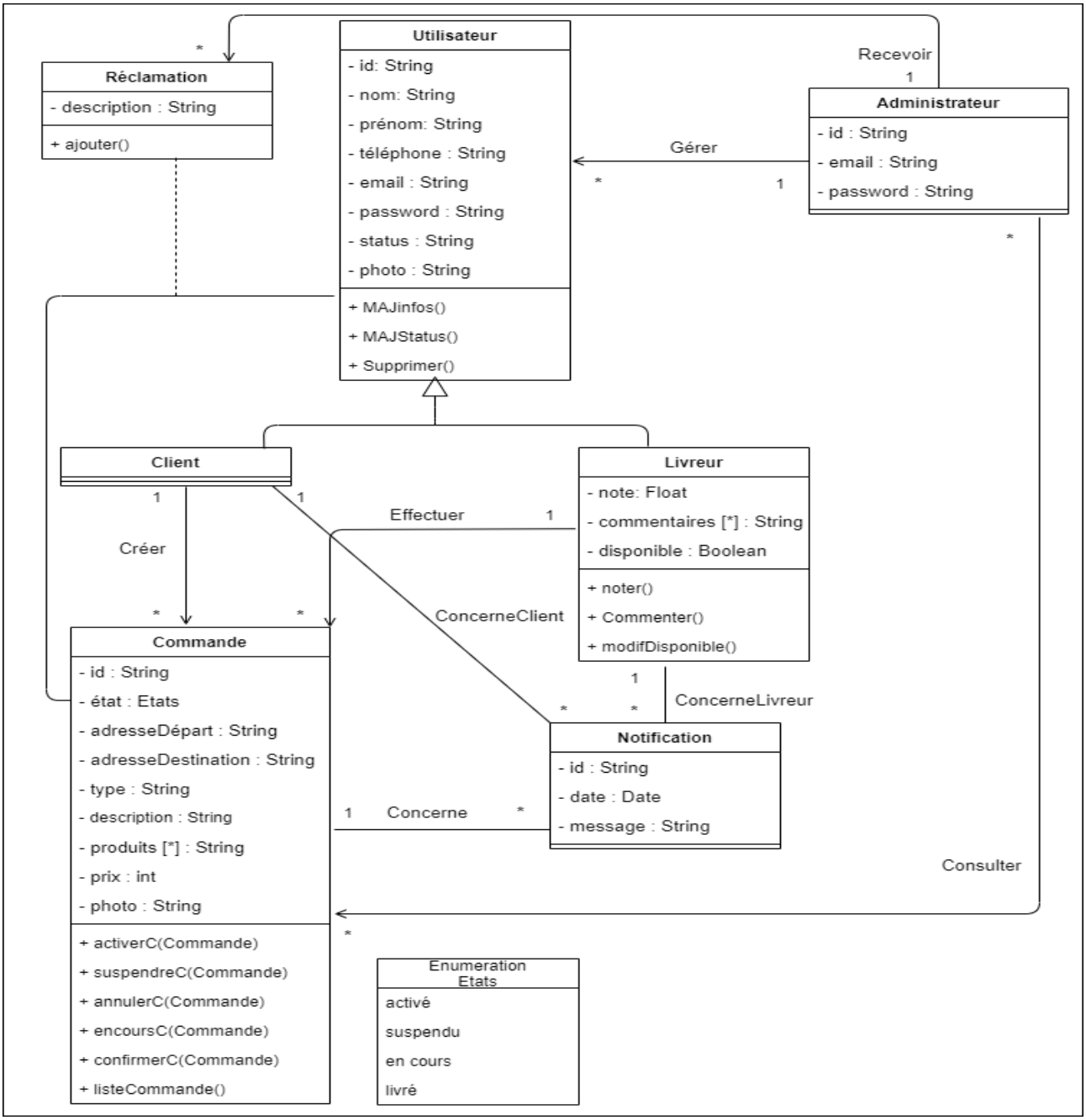

FIGURE 3.4 – Diagramme de classes de conception

3.4 Dictionnaire des données

Le tableau suivant décrit les types de données des attributs de chaque classe collectée dans le diagramme de classe.

Classe	Attribut	Désignation	Type
Utilisateur	id	identifiant	String
	nom	nom de l'utilisateur	String
	prénom	prénom de l'utilisateur	String
	téléphone	téléphone de l'utilisateur	String
	email	email de l'utilisateur	String
	password	mot de passe de l'utilisateur	String
Commande	id	identifiant	String
	etat	état de la commande	Enumeration
	adresseDepart	adresse de départ	String
	adresseDestination	adresse de destination	String
	type	type de commande livraison seule ou achat et livraison	String
	description	description de la commande	String
	produits	liste des produits à acheter	String
	prix	prix de livraison	int
Livreur	note	note du livreur	Float
	commentaires	commentaires sur le livreur	String[]
	disponible	disponibilité	Boolean
Adminstrateur	id	identifiant	String
	email	email de l'admin	String
	password	mot de passe	String
Notification	id	identifiant	String
	date	date de création	Date

	message	contenu de la notification	String
Réclamation	description	description de la récalamtion	String

Tableau 3.1 – Dictionnaire des données du diagramme de classe

3.5 Base de données NoSql

Le NoSql (**Not Only SQL**) regroupe de nombreuses bases de données, qui se caractérisent par une logique de représentation de données non relationnelle sans schéma logique défini a priori [27].
Les bases de données NoSql sont une nouvelle approche de stockage et de gestion des données développée par de grandes sociétés comme Google, Yahoo, etc., car les bases de données relationnelles existantes n'étaient pas en mesure de faire face aux exigences croissantes en matière de traitement des données [27]. La figure suivante regroupe les systèmes de gestion de bases de données avec leur classification [32].

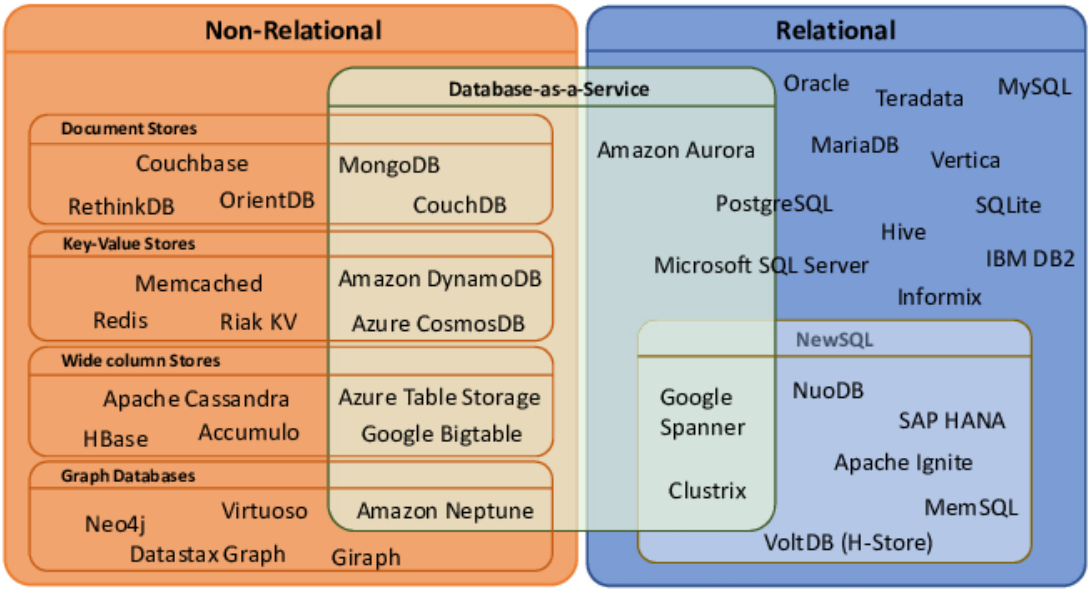

FIGURE 3.5 – Bases de données relationnelles et non relationnelles

L'intérêt d'utiliser une base de données NoSql dans notre projet est de faciliter le stockage des données semi structurées et la conception d'un schéma de base de données flexible pouvant être facilement modifié sans interruption de service.

3.5.1 Types des bases de données NoSql

Selon [29], il existe quatre types de bases de données non relationnelles, comme présenté dans la figure précédente.

- Base de données orientée clé-valeur **:** représente les données sous forme de clé/valeur, les valeurs peuvent être de simples chaînes de caractères ou des objets sérialisés complexes.
- Base de données orientée colonne **:** ressemble aux bases de données relationnelles, mais avec un nombre de colonnes dynamique.
- Base de données orientée document **:** est une variante du type clé/valeur, où la valeur est un document de type xml ou json.
- Base de données orientée graphe **:** basée sur la théorie des graphes, conçue pour les données dont les relations sont représentées comme graphes, et ayant des éléments interconnectés, avec un nombre indéterminé de relations entre elles.

3.5.2 Les bases de données SQL vs NoSql

Le tableau suivant décrit les différences entre les bases de données relationnelles et non relationnelles selon les propriétés relatives à la structure, l'évolutivité et l'interrogation des données, etc [9].

	BDD Sql	BDD NoSql
Schéma	Schéma défini au départ	Schéma dynamique
Langage d'interrogation	Sql	Chaque plateforme définit son propre langage
Evolutivité	Verticale (ajouter plus de puissance CPU, RAM à une machine existante)	Horizontale (ajouter de nouvelles machines pour augumenter les performances)
Structure	Table	- Paires clé/valeur - document - graphe - magasin à colonnes larges

Propriété suivie	ACID (atomicité, cohérence, isolation et durabilité)	Théorème de CAP (cohérence, disponibilité et tolérance de partition)

Tableau 3.2 – Différences entre Sql et NoSql

3.5.3 Structure de la base de données

Nous avons opté pour une base de données NoSql orientée document avec le format json. Les règles de transformation du relationnel vers un document json sont regroupées dans le tableau suivant [10] :

Relationnel	Document json
La relation est **un-à-un** ou **un-à-plusieurs**	Stocker les données associées en tant qu'**objets imbriqués**
la relation est **plusieurs-à-un** ou **plusieurs-à-plusieurs**	stocker les données associées sous forme de **documents séparés**
les lectures de données sont principalement des **champs parents**	stockez les enfants dans des **documents séparés**
les lectures de données sont principalement des **champs parent + enfant**	stocker les enfants en tant qu'**objets imbriqués**
les écritures de données sont principalement **parent ou enfant** (pas les deux)	stocker les enfants dans des **documents séparés**
les écritures de données sont principalement **parent et enfant** (les deux)	stocker les enfants en tant qu'**objets imbriqués**

Tableau 3.3 – Règles de transformation relationnel en NoSql json

La représentation du schéma de la base en arborescence json est comme suit :

FIGURE 3.6 – Structure de la base de données

3.6 Conclusion

Dans ce chapitre, nous avons présenté les diagrammes de séquence d'interaction relatifs à quelques cas d'utilisation de notre système. Ensuite nous avons élaboré le diagramme de classe de notre système qui illustre d'une manière globale la structure des éléments constituant la base de données associée à notre application. Ainsi, le choix et la structure d'une base de données NoSql ont été discuté à la fin du chapitre.
Dans le chapitre suivant nous allons aborder la partie réalisation de notre projet, en se basant sur les mécanismes et les solutions déterminés dans la phase de conception.

Chapitre 4

Réalisation

4.1 Introduction

Dans ce chapitre, la réalisation du projet va être présentée d'abord par les outils de développement ainsi que les choix d'implémentations de la base de données et des services invoqués. Suivi de l'architecture logicielle qui sert à organiser le code source du projet. Ensuite, une présentation des scénarios d'exécution des cas d'utilisation sera illustrée par l'enchaînement des interfaces graphiques de l'application.

4.2 Langages et framework de développement

La mise en œuvre du projet a nécessité l'utilisation de plusieurs langages de développement orientés objet, notre choix s'est porté sur le langage java et xml pour développer une application mobile Android pour le client et le livreur. De plus, le Framework angular est exploité pour réaliser l'espace de l'administrateur.

- **Java** : est un langage de programmation orienté objet, permettant le développement des applications mobiles natives, web et desktop [11].
- **Xml** : est un langage de balisage qui définit un ensemble de règles pour le codage des documents dans un format lisible à la fois par l'homme et par la machine [12].
- **Angular** : est un Framework open source basé sur TypeScript pour créer des applications Web mobiles et de bureau [13].

4.3 Environnements de développement

- **Android studio** : Android Studio est un environnement de développement pour développer des applications mobiles Android [14].
- **Visual Studio Code** : Visual Studio Code est un éditeur de code redéfini et optimisé pour la création et le débogage d'applications Web et cloud modernes [15].

4.4 Plateformes de services web

Un service web est une application accessible via un protocole Web standard (HTTP ou HTTPS) conçue pour communiquer avec d'autres applications.

- **Mapbox** : est une plateforme qui offre des services de cartographie et de localisation faciles à intégrer aux applications mobiles, web et desktop [16].
- **Geoapify** : est une plateforme de localisation qui possède de nombreux services tel que le convertisseur de coordonnées géographique en adresse [17].
- **Firebase** : est une plateforme qui regroupe un ensemble de services pour n'importe quel type d'application (Android, iOS, Javascript, Node.js, Java, etc.). Il propose d'héberger en NoSql et en temps réel des bases de données, du contenu, de l'authentification sociale (Google, Facebook, Twitter et GitHub) et des notifications [18].

4.5 Implémentation de la base de données

Dans cette section, nous allons montrer comment ajouter le service Firebase à notre projet et définir la base de données non relationnelle choisie ainsi que les services utilisés.

4.5.1 Firebase intégration et services utilisés

Firebase regroupe un ensemble de services web dans la même plateforme, ces services sont exploitables en passant par les étapes suivantes :

- Créer un projet dans la plateforme Firebase https ://console.firebase.google.com/

- Ajouter une application (Android, web) au projet crée précédemment.

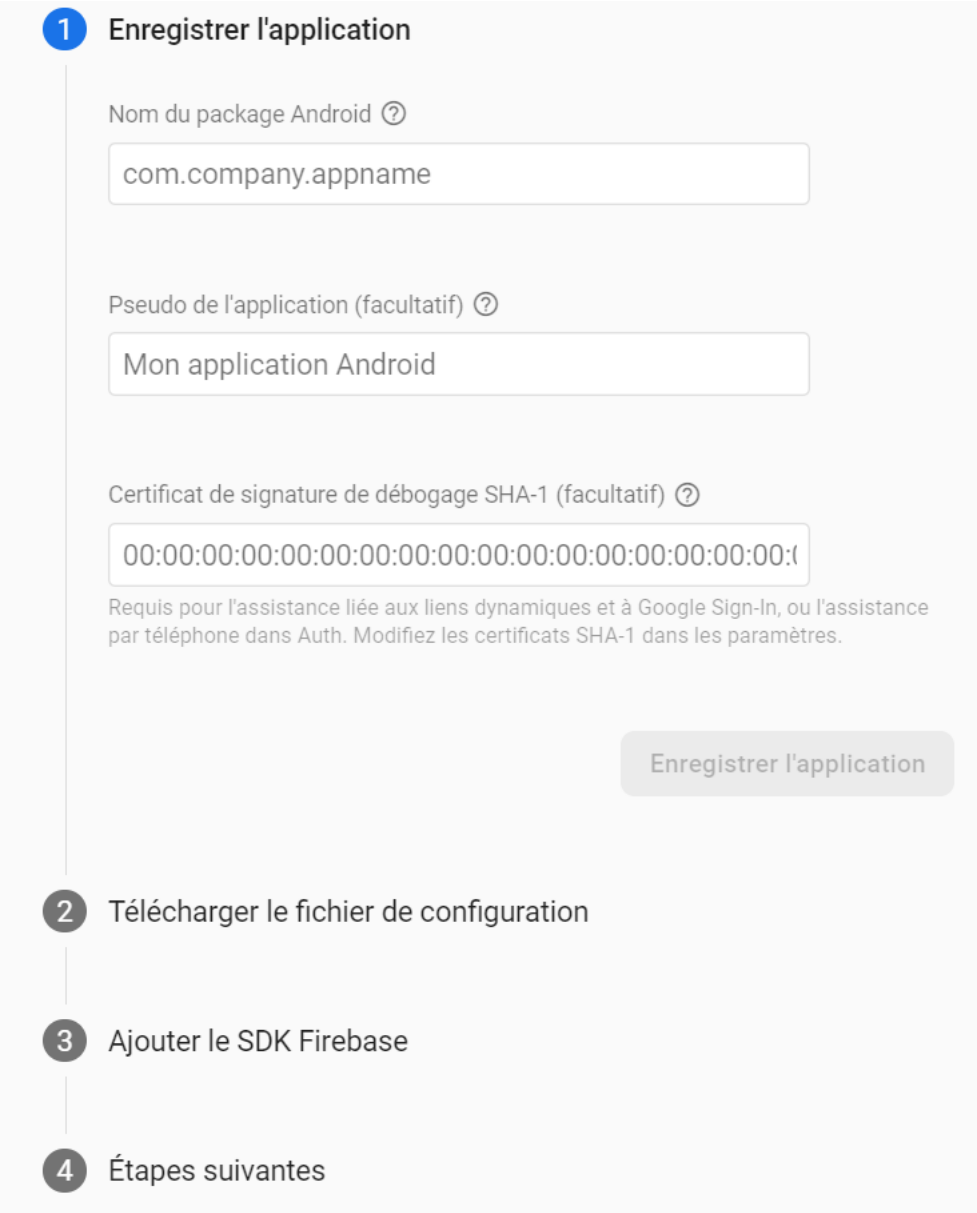

FIGURE 4.1 – Ajouter une application au projet

Nous pouvons à présent utiliser les services d'authentification, base de données, notification etc.

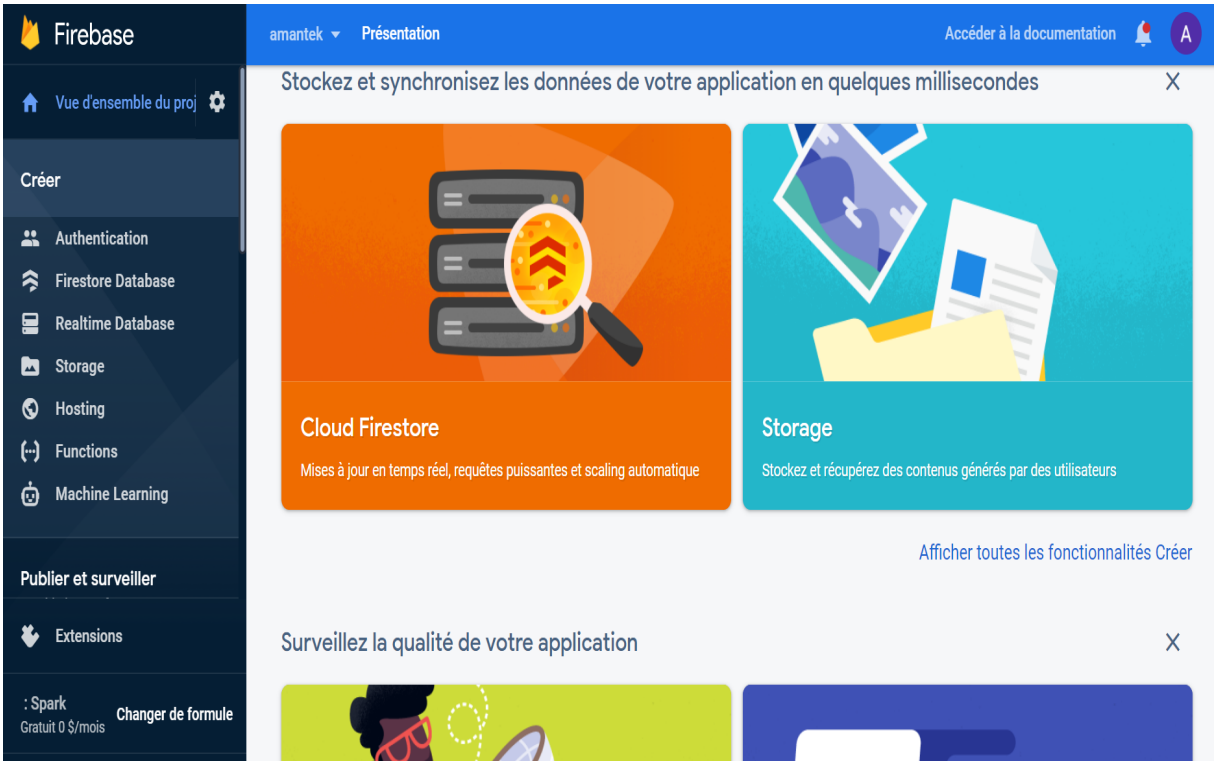

FIGURE 4.2 – Vue d'ensemble du projet firebase

Nous avons utilisé les services d'authentification, de base de données et de notifications :

a) **Authentification :**

Firebase Authentication fournit des services backend, des SDK faciles à utiliser et des bibliothèques d'interface utilisateur prêtes à l'emploi pour authentifier les utilisateurs auprès de l'application. Il prend en charge l'authentification à l'aide de mots de passe, de numéros de téléphone, de fournisseurs d'identité fédérés populaires tels que Google, Facebook et Twitter, etc [19].

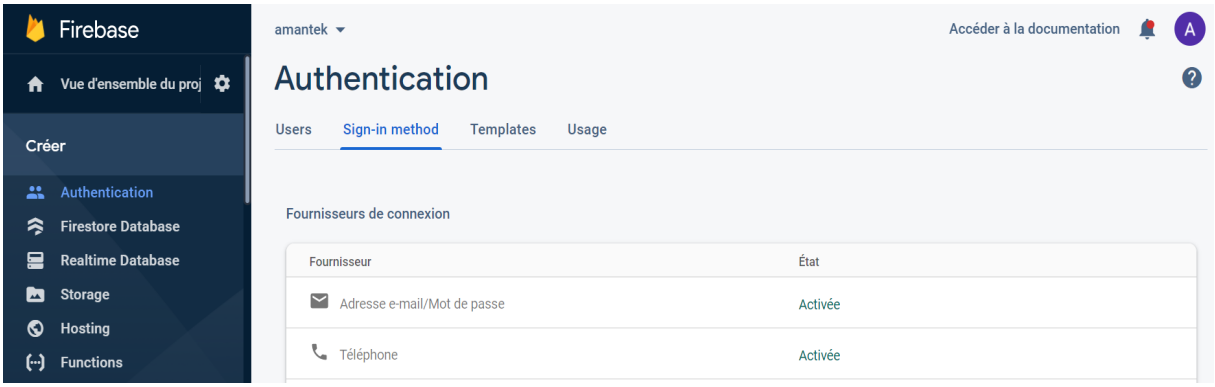

FIGURE 4.3 – Méthodes d'authentification activées

b) **Realtime database :**

La base de données Firebase Realtime est une base de données hébergée dans le cloud. Les données sont stockées au format JSON et synchronisées en temps réel avec chaque client connecté. Lorsque des applications multiplateformes sont créées avec les SDK (iOS, Android et JavaScript) de Firebase, tous les clients partagent une instance de base de données en temps réel et reçoivent automatiquement des mises à jour avec les données les plus récentes [20].

FIGURE 4.4 – Base de données Realtime database

c) **Cloud messaging :**

Firebase cloud messaging (FCM) est une solution de messagerie multi-plateforme qui permet de livrer des messages de manière fiable et gratuite. Il permet d'envoyer des messages de notification aux utilisateurs de l'application [21].

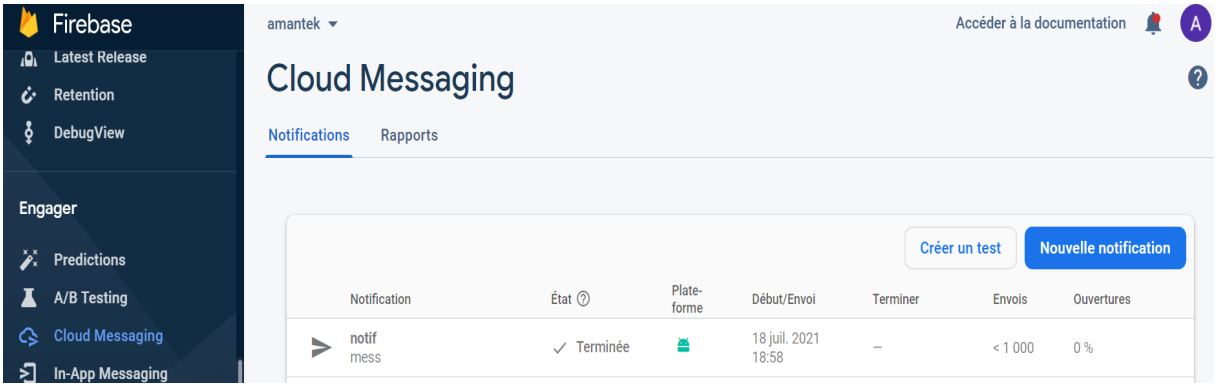

FIGURE 4.5 – Notifications et cloud messaging

d) **Geofire :**

GeoFire est une bibliothèque open source qui permet de stocker et d'interroger un ensemble d'éléments en fonction de leur emplacement géographique, GeoFire stocke

les données dans son propre format et son propre emplacement dans la base de données Firebase [22].

e) **Cloud storage :**

Cloud Storage est conçu pour stocker et diffuser du contenu généré par les utilisateurs, comme des photos ou des vidéos [23].

4.5.2 Architecture de l'application

L'architecture d'une application client/serveur traditionnelle implique le développement des deux parties frontend et backend ce qui ralenti la productivité des développeurs. Firebase, permet un effort minimal sur le backend car l'exploitation des services proposés est facilitée par les SDK fournis [24].

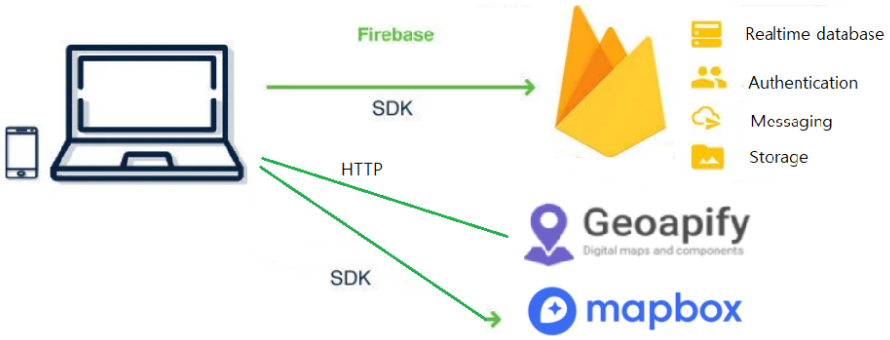

FIGURE 4.6 – Architecture globale de l'application

L'interaction entre l'application et l'api GeoApify se fait par une requête HTTP et une réponse json, quant à Mapbox, il offre un sdk facilitant l'intégration des cartes géographiques dans le projet.

4.6 Architecture logicielle

4.6.1 Présentation de MVVM

MVVM est un patron de conception, il vise à réaliser plus de découplage entre l'interface utilisateur et la couche intermédiaire, représentée par les éléments du VueModèle. La communication entre ces deux parties se fait exclusivement à travers des notifications de liaison-de-données bidirectionnelles [28].

Mvvm correspond au triptyque Model-View-ViewModel. Ce sont les trois éléments sur lesquels est basé le pattern :

- Le modèle de données (ou model) correspond aux différentes entités métier utilisées par l'application.

- la vue (ou view) correspond à la représentation qui est faite de ces données.

- le modèle de la vue (ou viewModel) correspond à une représentation abstraite de la vue. Il va aussi manipuler le modèle de données.

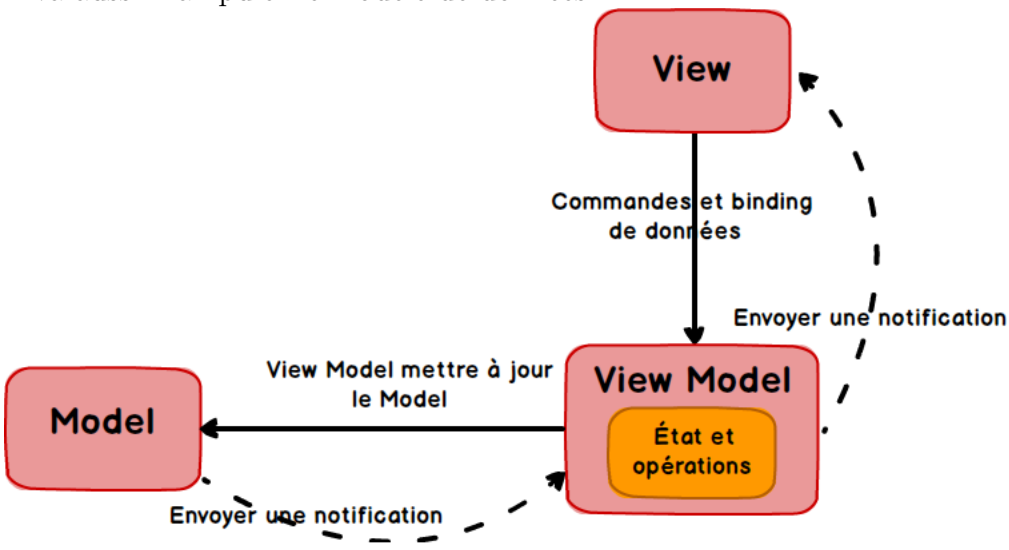

FIGURE 4.7 – Architecture mvvm

4.6.2 Comparaison entre MVC et MVVM

Le modèle MVC (model-view-controller) : une architecture logicielle est une façon d'organiser une interface graphique d'un programme. Elle consiste à distinguer trois entités distinctes qui sont, le modèle, la vue et le contrôleur ayant chacun un rôle précis dans l'interface [25].

MVC	MVVM
La plus ancienne architecture d'application Android.	Modèle d'architecture reconnu par l'industrie pour les applications.
La logique métier est mélangée avec Ul.	La logique métier est découplée de Ul
Les entrées utilisateur sont gérées par le contrôleur.	La vue prend l'entrée de l'utilisateur et agit comme le point d'entrée de l'application.
Prise en charge limitée des tests unitaires.	La testabilité unitaire est la plus élevée dans cette architecture.

La vue est liée avec le modèle.	La vue est indépendante du modèle.

Tableau 4.1 – Comparaison entre MVC et MVVM

4.7 Présentation des interfaces de l'application

Cette section présente les interfaces homme-machine de l'ensemble du projet et des fonctionnalités principales liées à chaque acteur du système. L'application Android déstinée au client est nommée "amantek" et celle du livreur est appelée "amantek livreur".

4.7.1 Interfaces principales du client

a) Interface connexion/inscription

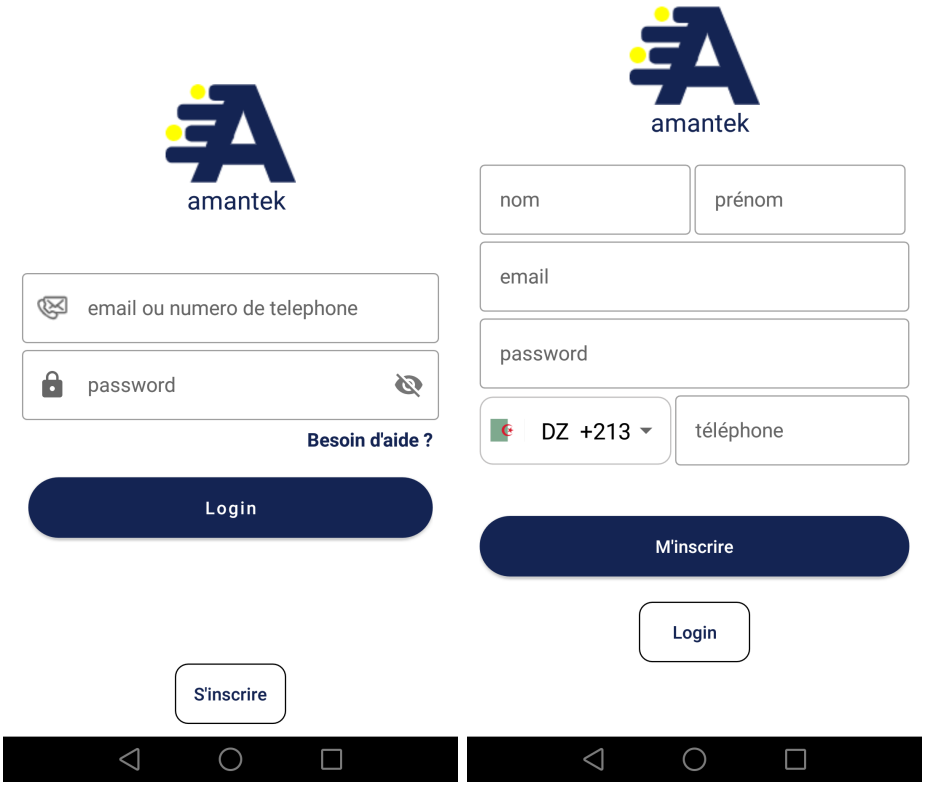

FIGURE 4.8 – Connexion/inscription

b) Interface commande et details commande

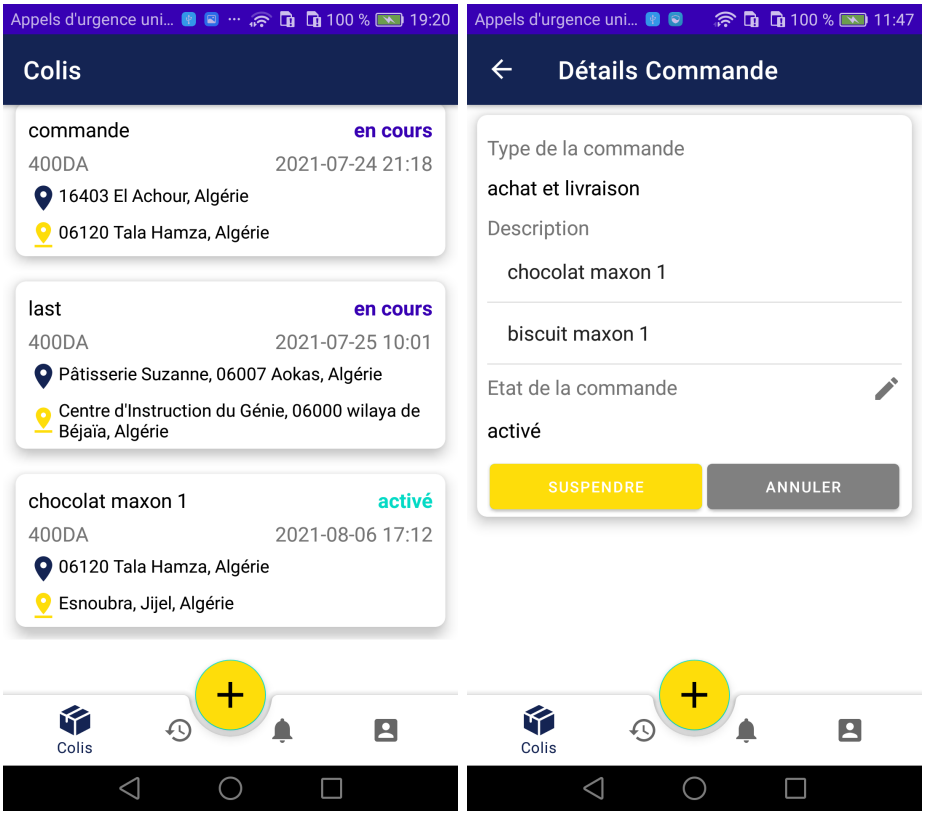

FIGURE 4.9 – Liste des commandes et détails

c) Interfaces créer commande de livraison

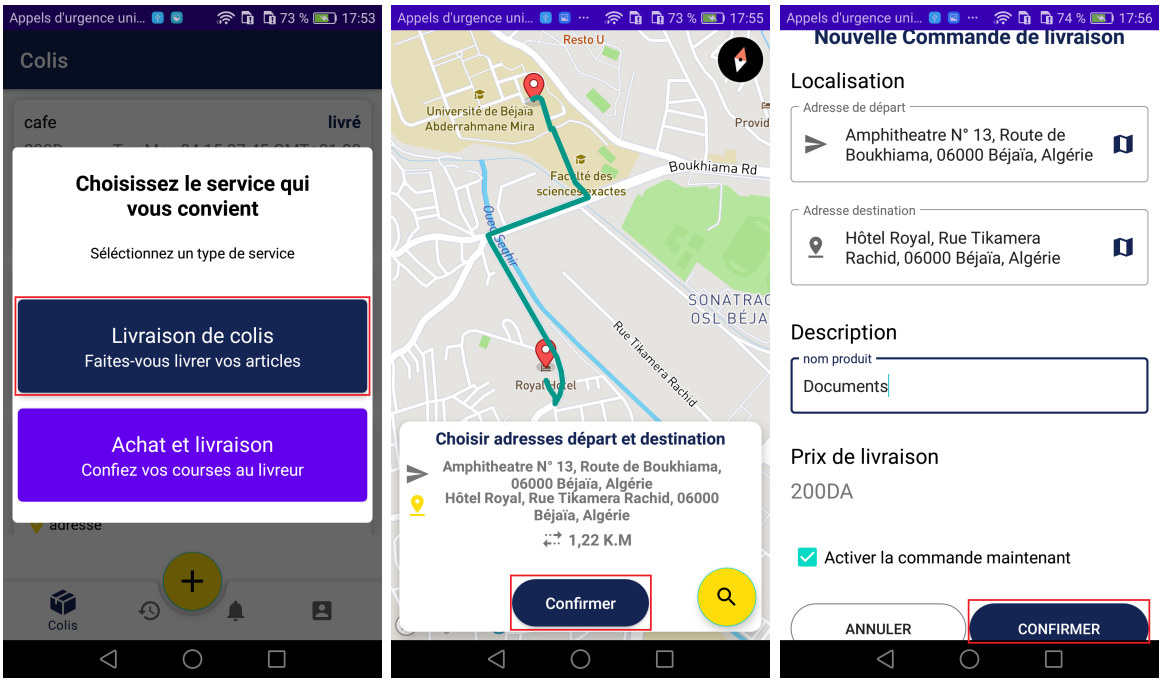

FIGURE 4.10 – Créer commande de livraison

d) Intefaces créer commande d'achat et livraison

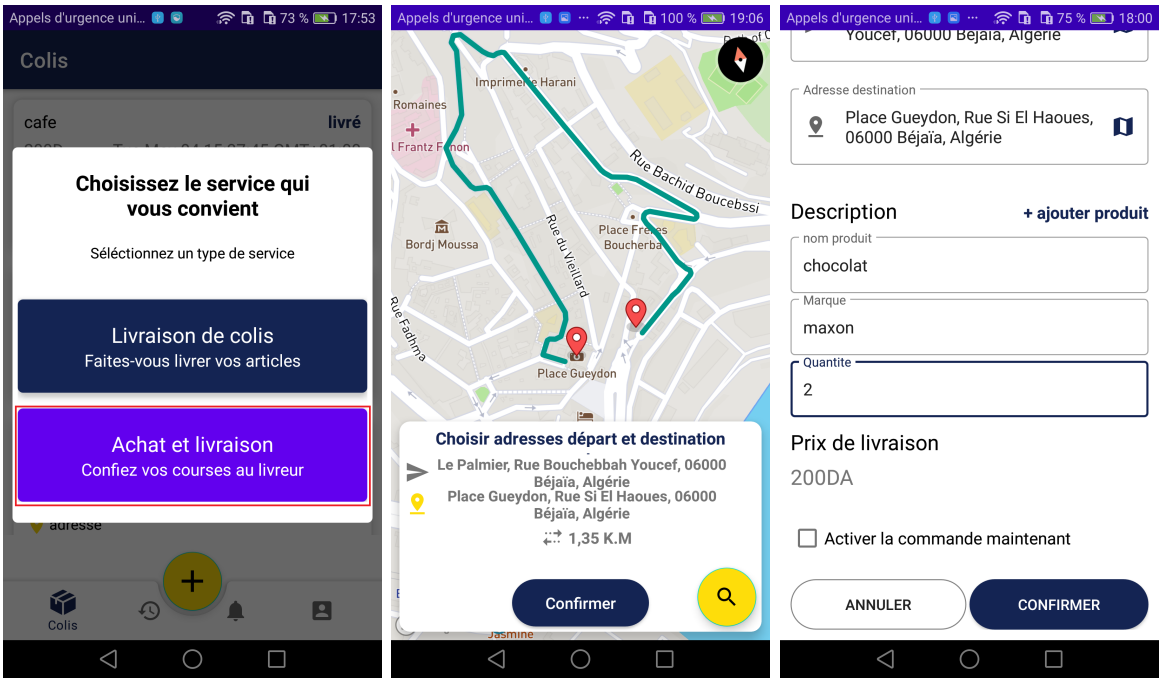

FIGURE 4.11 – Créer commande d'achat et livraison

e) Interface compte

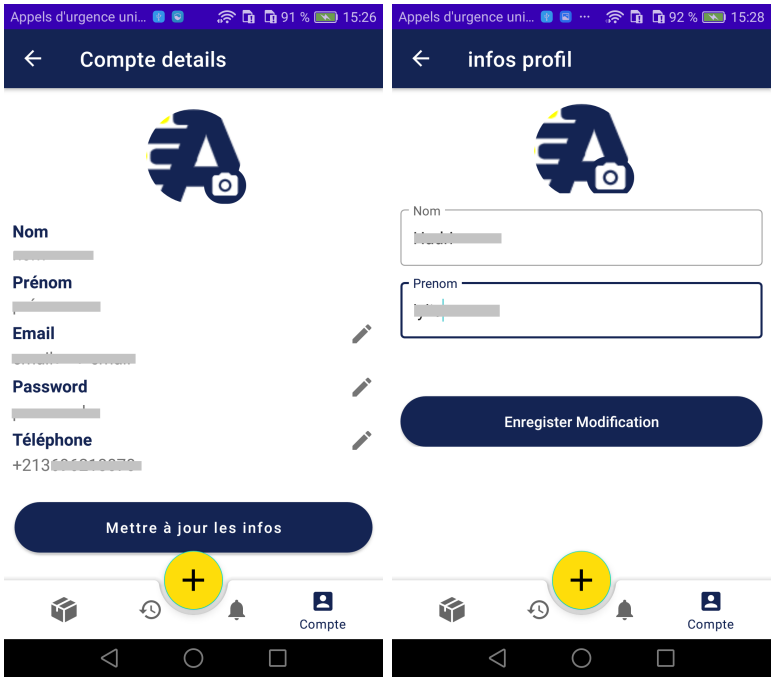

FIGURE 4.12 – Gérer son compte

f) Interface recevoir acceptation livreur

FIGURE 4.13 – Notification acceptation livreur

j) Interface confirmer reception commande et noter service

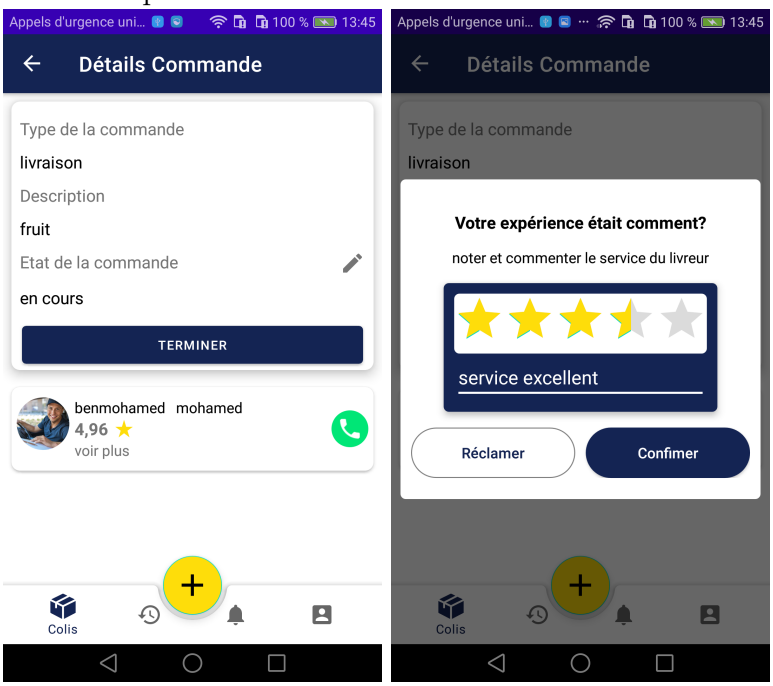

FIGURE 4.14 – Confirmer et noter livraison

4.7.2 Interfaces principales du livreur

a) Interface recevoir commande

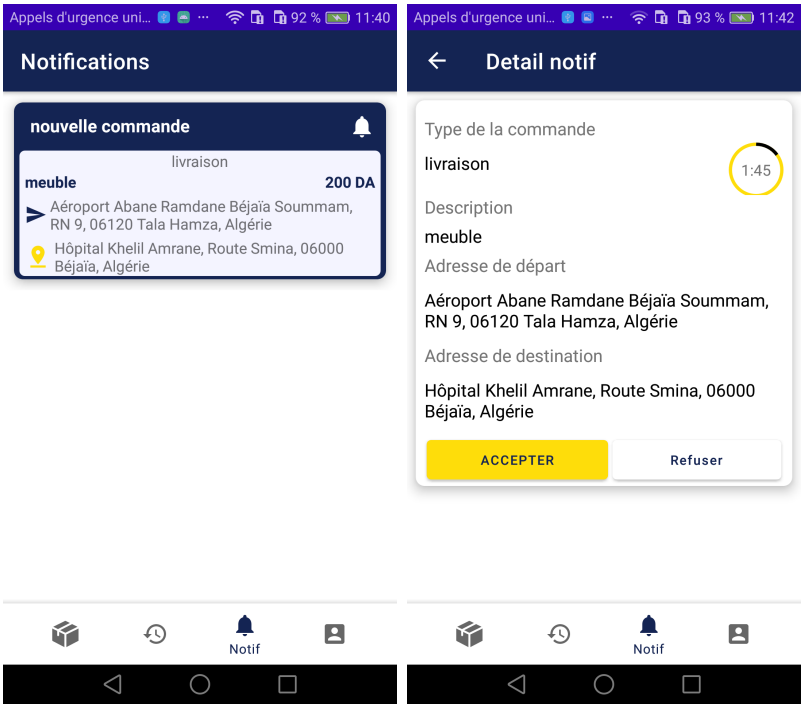

FIGURE 4.15 – Notification nouvelle commande

b) Interface contacter client

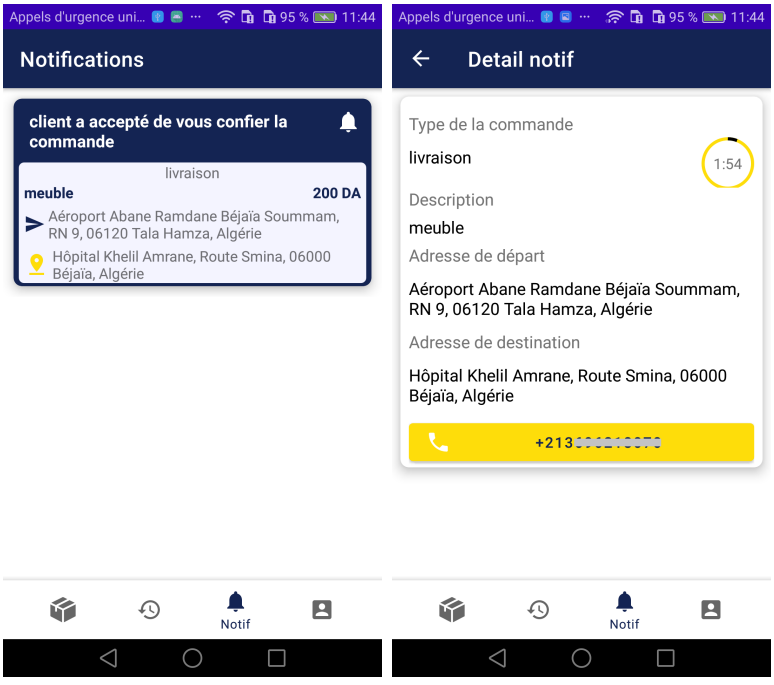

FIGURE 4.16 – Notification contacter client

4.7.3 Interfaces principales de l'administrateur

a) Interface s'authentifier administrateur

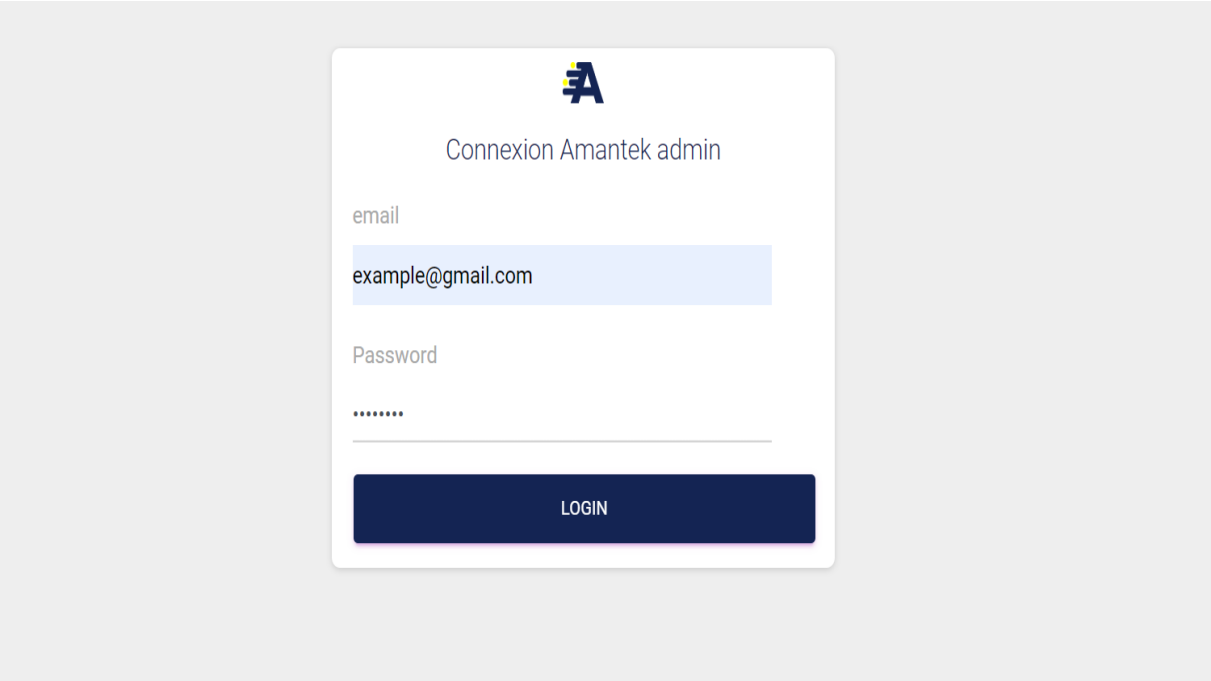

FIGURE 4.17 – Interface de connexion

b) Interface gestion des livreurs et des clients

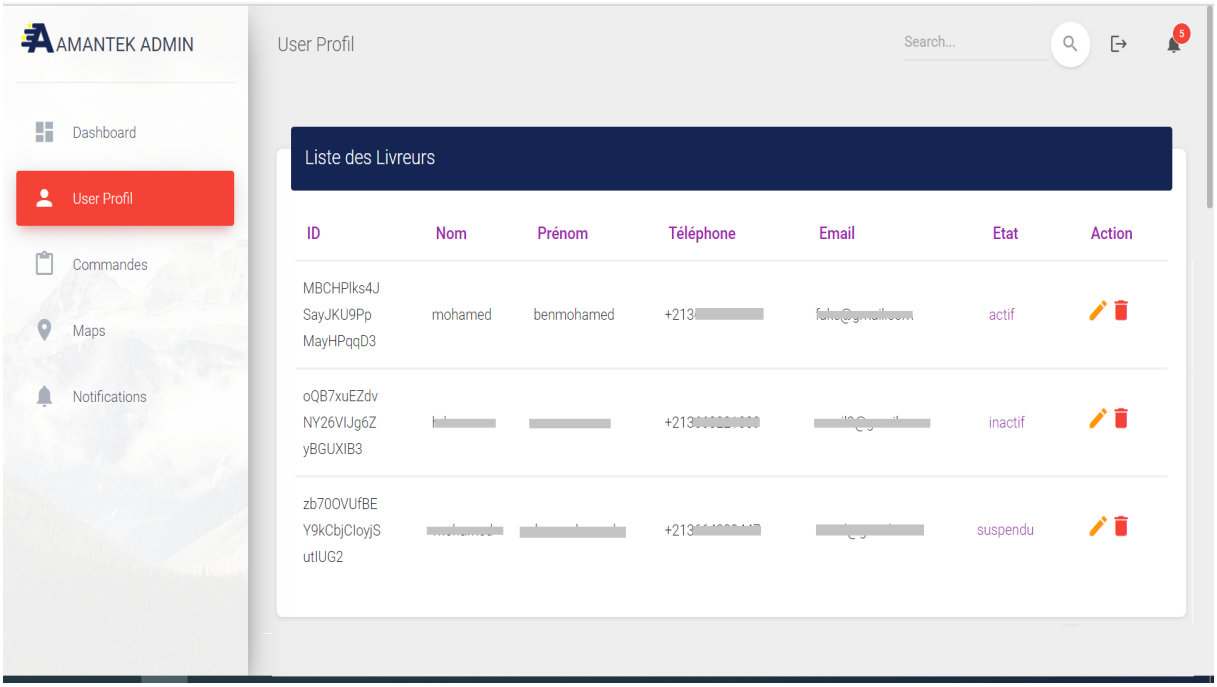

FIGURE 4.18 – Interface gestion des livreurs

c) Intreface consulter activitées de livraison

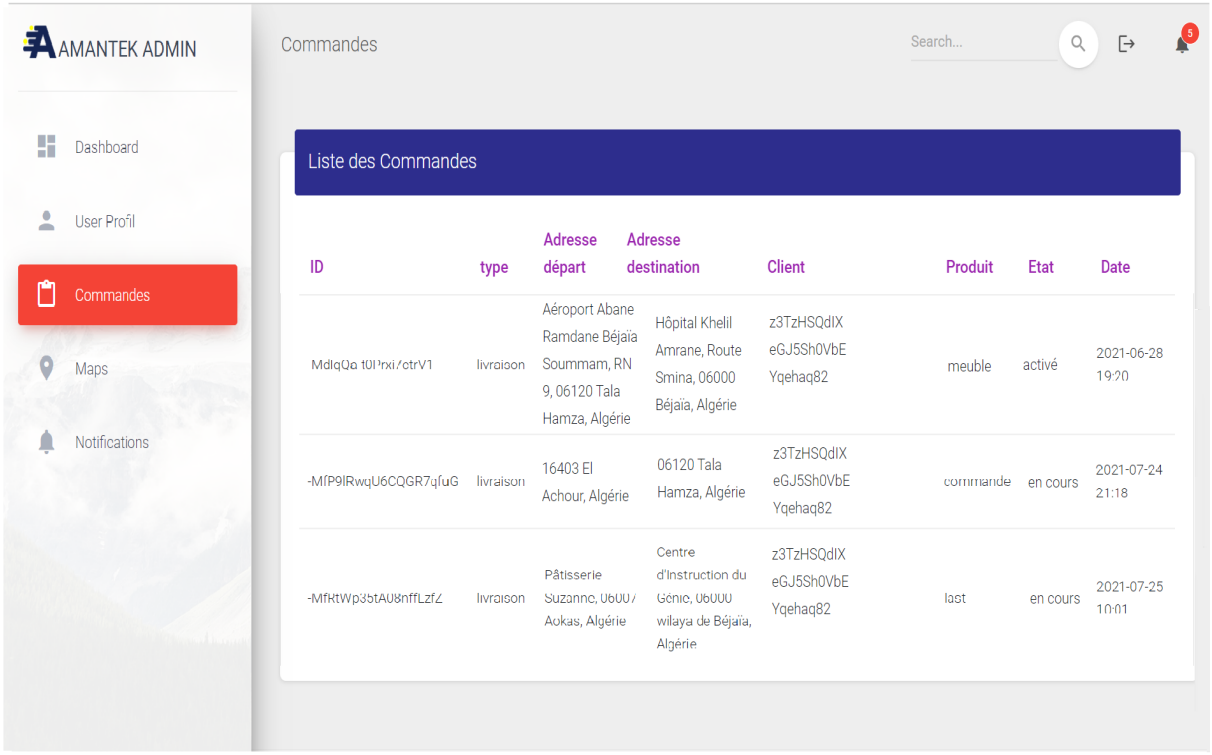

FIGURE 4.19 – Interface consulter commandes

4.8 Conclusion

Dans ce dernier chapitre, nous avons présenté la partie réalisation en décrivant l'ensemble des langages de programmation utilisés, l'environnement de développement ainsi que l'architecture logicielle adopté. Enfin, les fonctionnalités de base de l'application sont mises en valeur avec quelques interfaces.

Conclusion générale et perspectives

Dans ce document, nous avons étudié, conçu et réalisé un système qui permet aux utilisateurs de passer une commande de livraison et d'entrer facilement en contact avec un livreur. L'élaboration de ce projet a suivi plusieurs étapes, nous avons dans un premier temps exposé l'entreprise d'accueil et l'étude de l'existant qui nous a permis d'avoir un aperçu des solutions disponible sur le marché, ensuite la méthode de conception a défini les phases préliminaires du développement d'un système afin de le rendre plus fidèle aux besoins du clients, par la suite nous avons identifié les besoins à modéliser sous forme de diagrammes de cas d'utilisation, diagrammes de séquences, diagramme de classe et terminer par la réalisation du projet en développant notre application mobile sous la plateforme Android.

La réalisation du projet nous a permis d'enrichir nos connaissances en java, angular et firebase etc., ainsi que les outils du développement mobile sous Android. Egalement, ce travail fut bénéfique car il nous a donné un aperçu sur la vie professionnelle, pour mieux s'organiser dans notre travail, afin d'accomplir les tâches qui nous sont confiées dans les meilleures conditions et dans les plus brefs délais, malgré la crise sanitaire relative à la Covid-19.

Des perspectives d'amélioration de notre application restent toutefois indispensables. Nous envisageons d'ajouter de nouvelles fonctionnalités à notre application en y intégrant les points suivants :

- Le développement de commandes vocales permettra une utilisation par un plus large public notamment les personnes handicapées.
- Le suivi de la commande en temps réel est un excellent moyen pour rassurer le client de la progression de la commande.
- La messagerie en temps réel va faciliter aux utilisateurs la communication avec les livreurs.
- La recherche de commande pour les livreurs permettra au client d'avoir plus de chance que la commande soit satisfaite rapidement, et le livreur pourra définir des filtres de recherche selon la distance, la localisation, etc.

Bibliographie

[1] https://express.yassir.io/. consulté le : 22/04/2021.

[2] https://food.jumia.dz/. consulté le : 22/04/2021.

[3] https://market.yassir.com/. consulté le : 15/05/2021.

[4] https://www.jumia.dz/. consulté le : 15/05/2021.

[5] https://mrsool.co/. consulté le : 15/05/2021.

[6] https://glovoapp.com/. consulté le : 17/05/2021.

[7] https://www.uber.com/fr/fr/. consulté le : 18/08/2021.

[8] https://fr.wikipedia.org/wiki/M%C3%A9thodes_d%27analyse_et_de_conception. consulté le : 17/05/2021.

[9] http://infodecisionnel.com/data-management/big-data/comparaison-relationnel-vs-nosql/. consulté le 25/07/2021.

[10] https://www.slideshare.net/Dataversity/slides-nosql-data-modeling-using-json-documents-a-practical-approach. consulté le : 30/10/2021.

[11] https://docs.oracle.com/javase/1.5.0/docs/guide/. consulté le : 30/07/2021.

[12] https://www.w3.org/TR/REC-xml/. consulté le : 30/07/2021.

[13] https://angular.io/. consulté le : 30/07/2021.

[14] https://developer.android.com/studio. consulté le : 30/07/2021.

[15] https://code.visualstudio.com/. consulté le : 30/07/2021.

[16] `https://www.mapbox.com/`. consulté le : 30/07/2021.

[17] `https://www.geoapify.com/`. consulté le : 30/07/2021.

[18] `https://fr.wikipedia.org/wiki/Firebase`. consulté le : 30/07/2021.

[19] `https://firebase.google.com/docs/auth`. consulté le : 16/08/2021.

[20] `https://firebase.google.com/docs/database`. consulté le : 16/08/2021.

[21] `https://firebase.google.com/docs/cloud-messaging`. consulté le : 16/08/2021.

[22] `https://firebaseopensource.com/projects/firebase/geofire-android/`. consulté le : 12/09/2021.

[23] `https://firebase.google.com/docs/storage`. consulté le : 12/09/2021.

[24] `https://www.hammermarketing.com/what-is-firebase-and-how-does-it-work/`. consulté le : 30/09/2021.

[25] `https://www.irif.fr/~carton/Enseignement/InterfacesGraphiques/Cours/Swing/mvc.html`. consulté le : 21/09/2021.

[26] Pascal André and Alain Vailly. Développement de logiciel avec uml2 et ocl ; cours et exercices corrigés, collection technosup. *Editions Ellipses*, 2013.

[27] STÉPHANE CROZAT. Conception de bases de données. Bases de données non-relationnelle. utc Formation. 4 septembre 2017.

[28] Aymen Daoudi. Étude des patrons architecturaux de type mvc dans les applications android. 2018.

[29] Lionel Heinrich. *Architecture NoSQL et réponse au théorème CAP*. PhD thesis, Haute école de gestion de Genève, 2012.

[30] Ivar Jacobson, Grady Booch, and James Rumbaugh. *Le processus unifié de développement logiciel*. Eyrolles, 2000.

[31] Pascal Roques and Franck Vallée. *UML 2 en action : de l'analyse des besoins à la conception*. Editions Eyrolles, 2011.

[32] G Dumindu Samaraweera and J Morris Chang. Security and privacy implications on database systems in big data era : A survey. *IEEE Transactions on Knowledge and Data Engineering*, 33(1) :239–258, 2019.

Annexes

A Diagrammes de séquences

A.1 Diagramme de séquence du cas d'utilisation "s'inscrire"

La figure suivante représente le diagramme de séquence système du cas d'utilisation "s'inscrire" :

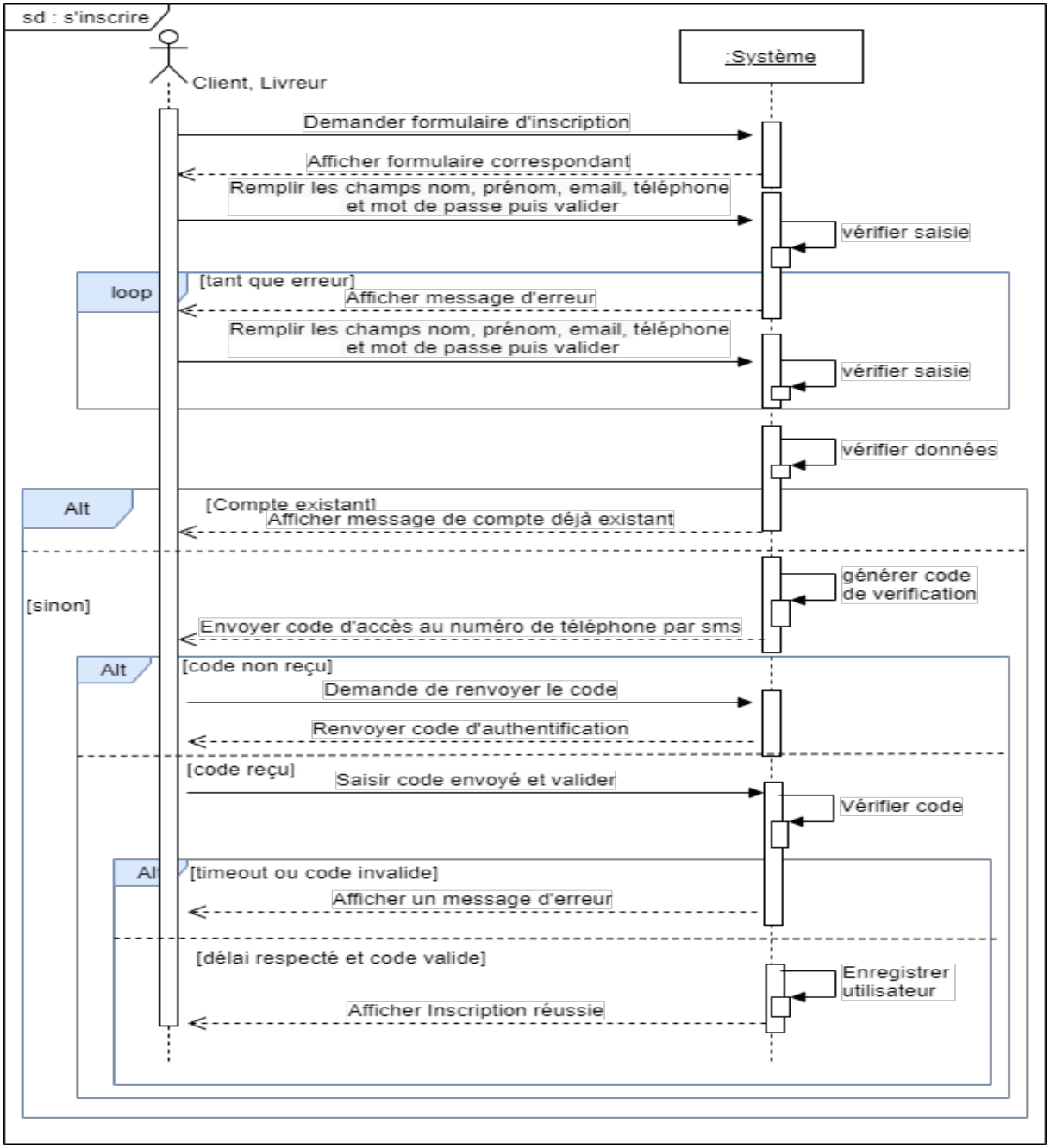

FIGURE 20 – Diagramme de séquence système "s'inscrire"

A.2 Diagramme de séquence du cas d'utilisation "recevoir commande client"

La figure suivante représente le diagramme de séquence système du cas d'utilisation "recevoir commande client" :

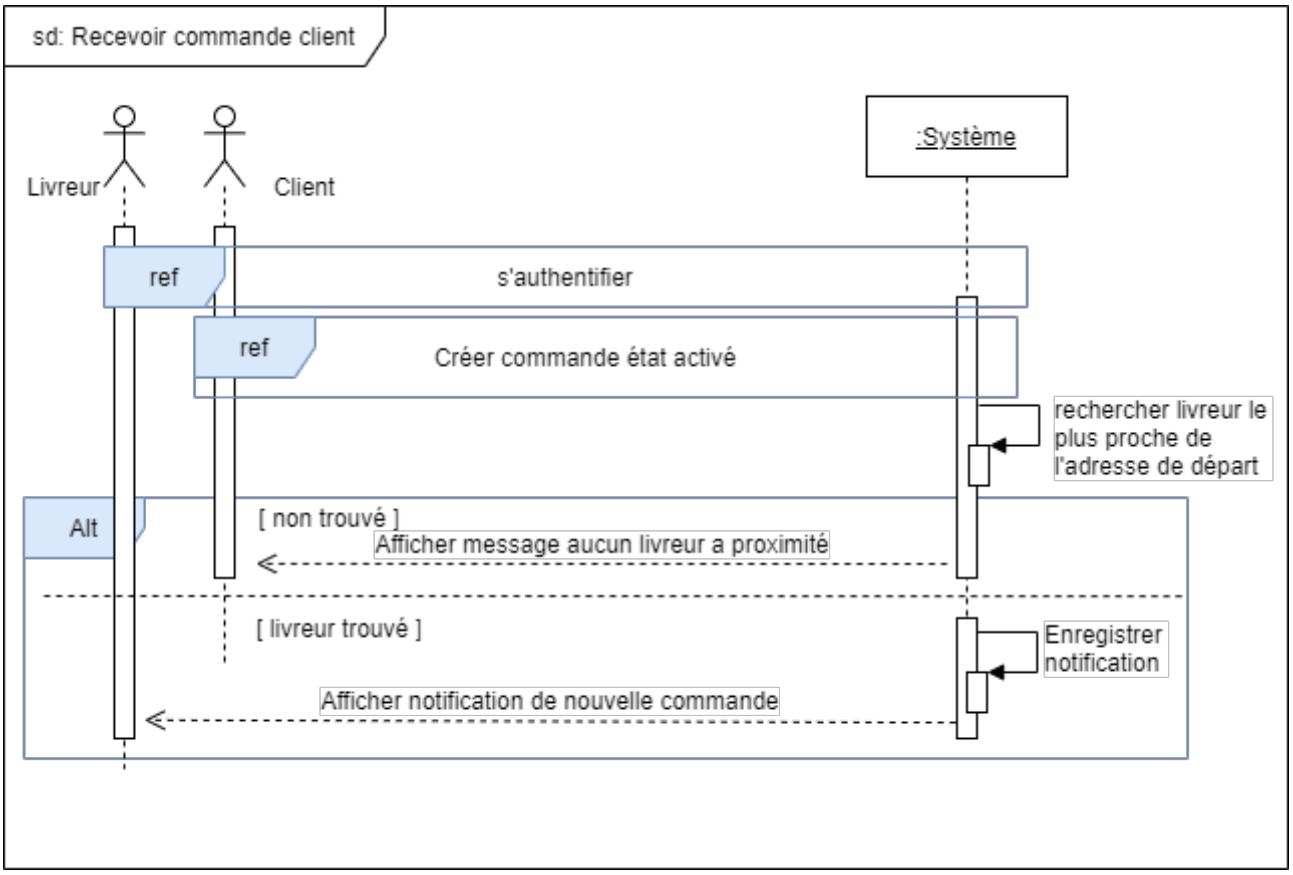

FIGURE 21 – Diagramme de séquence "recevoir commande client"

B Diagrammes d'interaction

B.1 Diagramme d'interaction "s'inscrire"

La figure suivante représente le diagramme de séquence d'interaction du cas d'utilisation "s'inscrire" :

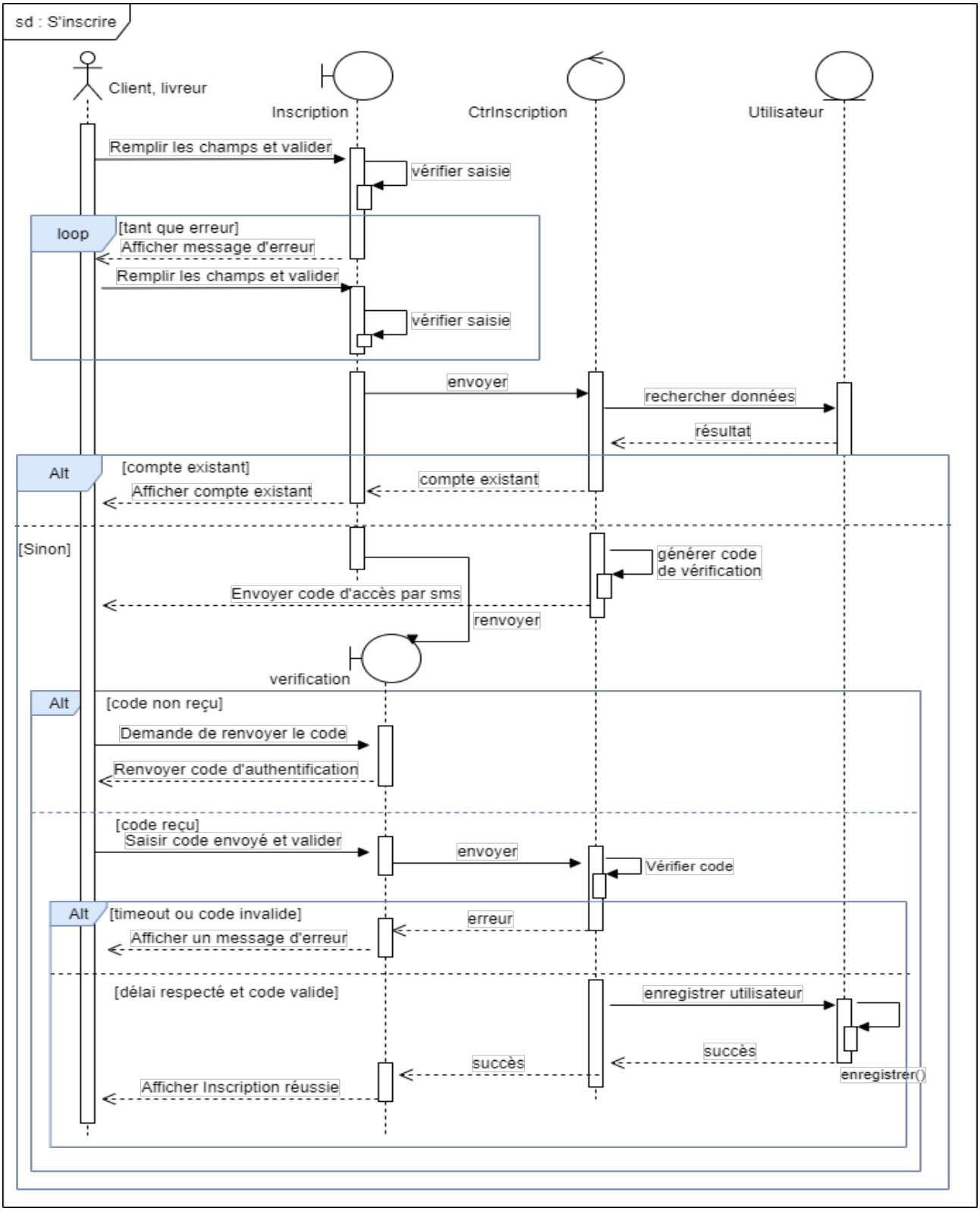

FIGURE 22 – Diagramme d'interaction "s'inscrire"

B.2 Diagramme d'interaction "recevoir commande client"

La figure suivante représente le diagramme de séquence d'interaction du cas d'utilisation "recevoir commande client" :

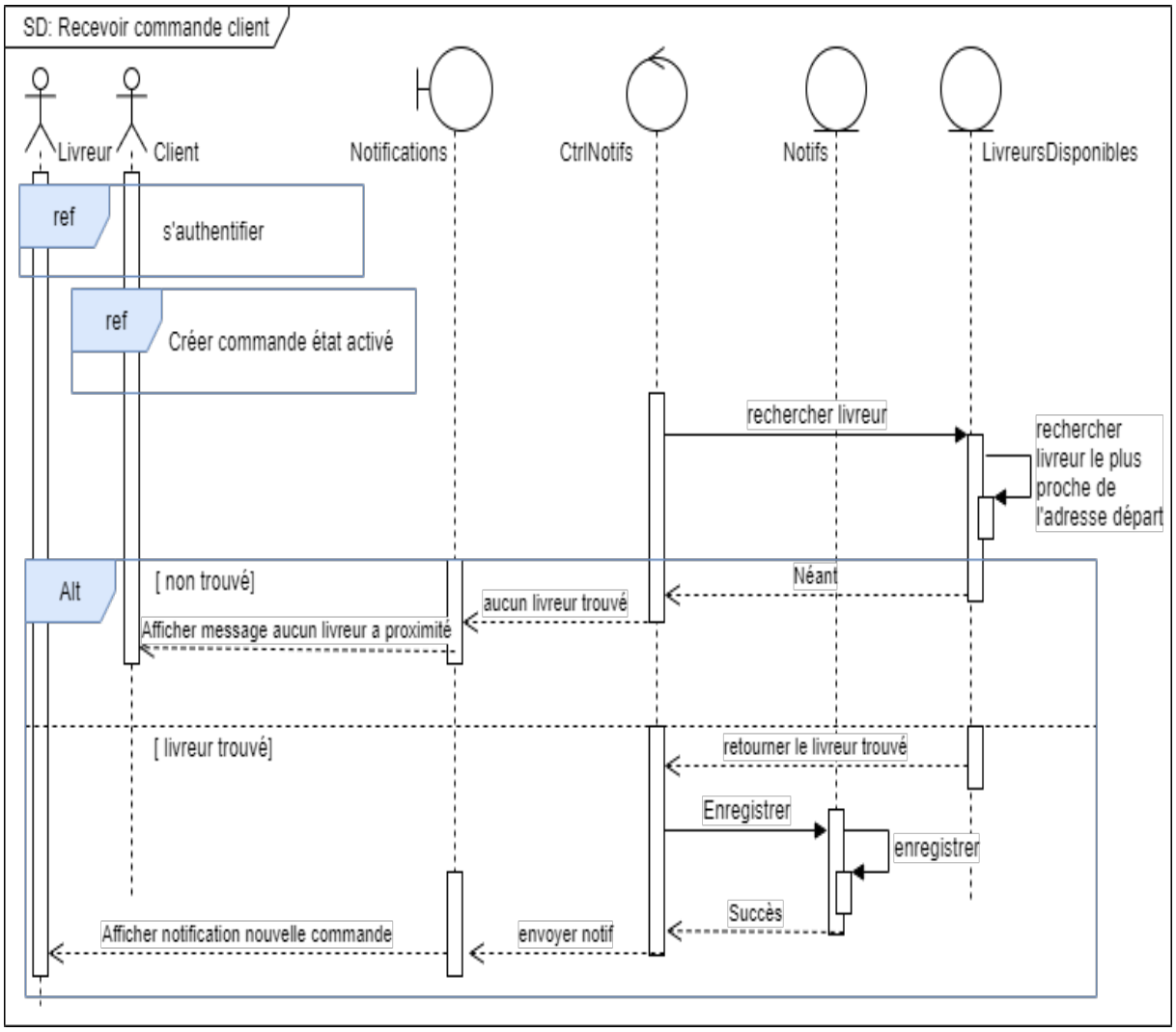

FIGURE 23 – Diagramme d'interaction "recevoir commande client"

Résumé

Ce travail a été réalisé dans le cadre d'un projet de fin d'étude de Master en Génie logiciel, portant sur la livraison. Pour ce faire, nous avons conçu et réalisé un système qui prend en charge la création de commandes de livraison, de mettre en contact le client et un livreur rapidement.

Pour mettre en œuvre notre solution, nous avons utilisé un processus de développement appelé Processus Unifié (UP), qui se base sur UML comme langage de modélisation conçu pour fournir une méthode normalisée pour la conception et la construction des documents nécessaires au bon développement.

Cette conception est mise en œuvre sous l'environnement de développement Android Studio, en langage java ainsi que Visual Studio code avec angular comme framework, Firebase pour l'implémentation de la base de données et des services offerts par la plateforme, sans oublier les services web pour la localisation tel que Mapbox et Geoapify.

Mots clés : UP, UML, application mobile, java, firebase, nosql, angular, android studio, realtime database, cloud messaging.

Abstract

This work was carried out as part of an end-of-study project of a Master in Software Engineering, focusing on delivery. To do this, we have designed and produced a system that supports the creation of delivery orders, to put in contact the customer and a delivery person quickly.

For the implementation of our solution, we used a development process called Unified Process (UP), which is based on UML as a modeling language designed to provide a standardized method for designing and building the documents necessary for proper development.

This conception is implemented under the Android Studio development environment, in java language as well as Visual Studio code with Angular as framework, Firebase for the implementation of the database and the services offered by the platform, not to mention the web services for localization such as Mapbox and Geoapify.

Keywords : UP, UML, mobile app, java, firebase, nosql, angular, android studio, realtime database, cloud messaging.