الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
المدرسة الوطنية العليا للتكنولوجيات المتقدمة
Ecole Nationale Supérieure des Technologies Avancées

Département Génie Logistique et Transport

Mémoire de fin d'étude en vue de l'obtention du diplôme

D'INGÉNIEUR d'État

-Filière-

Ingénierie des transports

-Spécialité -

Ingénierie de la chaîne logistique et Ingénierie des transports

Thème

Application de livraison en utilisant des algorithmes d'optimisation et d'affectation

Réalisé par
BENBOUHADI Abdarrahman
BENOUADAH Mohamed Anis

Présenté et soutenu publiquement le 28/06/2025

Composition du Jury :

Président	M.TALBI Tayeb	Maitre-assistant A	ENSTA
Promoteur	M.ANNAD Oussama	Maître conférence B	ENSTA
Examinateur	M.ZIDANI Omar	Maitre-assistant A	ENSTA

2024/2025

Remerciements

Nous exprimons avant tout notre profonde reconnaissance à ***Allah, le Tout-Puissant,*** *pour nous avoir accordé la santé, la patience et la force nécessaires à l'accomplissement de ce travail.*

Nous adressons nos remerciements les plus sincères à ***Monsieur Annad oussama,*** *notre encadrant, pour la qualité de son accompagnement tout au long de ce projet. Sa disponibilité, son exigence scientifique, sa bienveillance et la richesse de ses conseils ont été déterminants dans l'aboutissement de ce travail. Nous lui sommes particulièrement reconnaissants pour sa confiance et son soutien constant.*

Nos remerciements vont également aux ***membres du jury,*** *qui nous honorent par leur présence et par l'intérêt porté à notre travail. Leurs remarques et suggestions sont pour nous une source précieuse d'enrichissement.*

Nous tenons à remercier chaleureusement ***l'ensemble des collaborateurs*** *pour leur accueil, leur coopération . Leur expertise et leur retour d'expérience ont grandement contribué à la pertinence et à la concrétisation de notre étude.*

Enfin, nous souhaitons exprimer notre gratitude à ***toutes les personnes,*** *de près ou de loin, qui ont participé à la réalisation de ce projet. Leur soutien moral, leurs encouragements et leurs conseils ont été pour nous une aide précieuse tout au long de ce parcours.*

Dédicace

*Une pensée particulière à **mon grand-père**, dont la sagesse, la mémoire et les valeurs continuent de m'inspirer chaque jour. Ce travail lui est aussi dédié, en hommage à tout ce qu'il représente pour moi.*
*À **ma chère grand-mère, qui nous a quittés**,*
je rends un hommage profond et sincère.

*Je tiens à exprimer ma profonde gratitude à **mes amis** qui ont été présents tout au long de ce parcours.*

*À **Dorin**,(Oasis luxury) mon binôme, pour son esprit d'équipe et sa constance.*
*À **Fouad**, **Dtippy Dz**, et à **tous les lhabat**, pour les moments de complicité et de motivation partagés.*

*À **toute ma promotion GLT**, je vous félicite pour votre courage, votre persévérance et votre réussite. Ce mémoire est aussi un peu le vôtre.*
*À **tous ceux que j'aime et qui m'aiment**, merci d'avoir été là, simplement.*
*À **monsieur Djabailia Iheb (Djagro)**, notre encadrant de stage, pour sa disponibilité, ses conseils précieux, sa pédagogie et son soutien tout au long de notre projet. Son accompagnement a joué un rôle important dans l'aboutissement de ce travail.*

*Et enfin, **à moi-même** :*
Pour avoir tenu bon malgré les doutes, les retards et les nuits blanches.
Pour ne pas avoir abandonné, même quand c'était difficile.
Pour avoir cru, avancé, et accompli ce travail.
Merci à moi.

Merci à vous tous.
BENOUADAH Mohamed Anis

Dédicace

Tout d'abord je commence par remercier Dieu qui nous guide dans notre vie vers le bon chemin.

Je dédie ce travail :
À ma chère mère qui était toujours à mes côtés, qui prenait soin de moi, qui s'inquiétait quand j'étais inquiet et qui se réjouissait quand j'étais heureux, merci maman.
A mon père qui m'a donné tout ce que j'avais besoin, qui m'a toujours poussé à être parmi les meilleur dans ce que je fais, merci papa.

A ma sœur qui m'a aidé dans mes études depuis mon enfance, qui m'a toujours motivé d'être le premier dans mes études, mon domaine, et m'a aidé à poursuivre ma patient.

A mes frères TAHIR et MOHAMMED que dieu lui accorde la réussite et le bonheur.
Une pensée toute particulière à **Mayar** et **Nassim**, qui occupent une place spéciale dans mon cœur.

*À **ma chère grand-mère, qui nous a quittés**,*
je rends un hommage profond et sincère.

Je dédie ce travail à mes professeurs, mon encadrant Mr ANNAD OUSSAMA, M^me^ MOULAI RATIBA, Mr AMICHI HOCINE
Une pensée particulière à mes amis de la promotion précédente à Mr CHERFAOUI YASSER , Mr BOUKHENNOUFA ANIS , Mr DJEBABLIA IHAB qui je dédie aussi ce travail, en souvenir des moments enrichissants que nous avons vécus ensemble.

Mon binôme ANIS BENOUADAH, et à toute personne m'avait aidé dans mon domaine et ma vie.
Je dédie ce travail à tous mes amis, chacun par son nom, à Hamza ,Mohammed,Jamel,Hichem,Abdellah,Walid,Fouad,Wassim,Oussama,Khairo,Wali,Abderrahmen, Zaki,Haithem,Louay,Bilel,islam.
À tous ceux que je n'ai pas pu nommer, que cela soit un oubli involontaire, je vous exprime ma sincère reconnaissance. Vos encouragements et votre amitié resteront gravés dans ma mémoire.

Merci à vous tous.
BENBOUHADI ABDERRAHMEN

ملخص
خلال من .المحلي السياق مع ومكيف متكامل تكنولوجي حل نشر خلال من جذريًا الجزائر في التوصيل مشهد بتحويل منصتنا تقوم .القطاع في للكفاءة جديدًا معيارًا نؤسس الحقيقي، الوقت في والتتبع المسارات، وتحسين الطلبات، لإدارة الذكي التنسيق
ويستفيد وأرباحهم، إنتاجيتهم من تزيد عالية أداء على التوصيل عمال يحصل :السلسلة في حلقة لكل ملموسة فوائد الابتكارات هذه تجلب لملاءمة خصيصًا المصمم المخصص، نهجنا يتميز .وموثوقة سريعة بخدمة العملاء يتمتع بينما واقتصادية، سلسة لوجستيك من التجار .الاستراتيجي الطعام توصيل قطاع في خاص بشكل الجزائرية، السوق خصوصيات
المتنوعة الاحتياجات بدقة يلبي مما مرنة، توصيل وخيارات الديناميكية، الموارد وإدارة الذكي، التخطيط مثل متقدمة ميزات المنصة تدمج الجزائر في بأكمله التوصيل قطاع وتطوير تحديث في الأمام إلى كبيرة خطوة المتكامل الحل هذا يمثل .سواء حد على والأفراد للمحترفين

الكلمات المفتاحية التوزيع ,التتبع المباشر, تحسين المسارات, منصة التوصيل,اللوجستيك الذكية الذكي

Résumé
Notre plateforme transforme radicalement le paysage de la livraison en Algérie en déployant une solution technologique complète et adaptée au contexte local. En synchronisant intelligemment la gestion des commandes, l'optimisation des itinéraires et le suivi en temps réel, nous établissons un nouveau standard d'efficacité dans le secteur.
Cette innovation apporte des bénéfices concrets à chaque maillon de la chaîne : les livreurs accèdent à des outils performants qui maximisent leur productivité et leurs revenus, les commerçants bénéficient d'une logistique fluide et économique, tandis que les clients profitent d'un service rapide et fiable. Notre approche sur-mesure, spécialement calibrée pour les particularités du marché algérien, se distingue particulièrement dans le segment stratégique de la livraison alimentaire.
La plateforme intègre des fonctionnalités avancées comme la planification intelligente, la gestion dynamique des ressources et des options de livraison flexibles, répondant ainsi avec précision aux besoins diversifiés des professionnels comme des particuliers. Cette solution complète représente une avancée majeure pour moderniser et professionnaliser l'ensemble du secteur de la livraison en Algérie.

Mots clé : Logistique intelligente, Plateforme de livraison, Optimisation des itinéraires, Suivi en temps réel, Affectation intelligente

Abstract
Our platform is radically transforming the delivery landscape in Algeria by deploying a comprehensive technological solution tailored to the local context. By intelligently synchronizing order management, route optimization, and real-time tracking, we are establishing a new standard of efficiency in the sector.
This innovation delivers concrete benefits to every link in the chain: delivery personnel gain access to high-performance tools that maximize their productivity and earnings, merchants benefit from smooth and cost-effective logistics, while customers enjoy a fast and reliable service. Our customized approach, specifically calibrated for the particularities of the Algerian market, stands out particularly in the strategic segment of food delivery.
The platform incorporates advanced features such as intelligent planning, dynamic resource management, and flexible delivery options, thereby precisely meeting the diverse needs of both professionals and individuals. This comprehensive solution represents a major step forward in modernizing and professionalizing the entire delivery sector in Algeria.
Key words: Smart logistics, Delivery platform, Route optimization, Real-time tracking, Smart allocation

SOMMAIRE

Dédicace 3
LISTE DES TABLEAU 8
LISTE DES FIGURES 9
Liste des abréviations 10
Introduction générale 11
Chapitre 1 Contexte et genèse de l'application 14
1.1 Genèse de l'idée 15
1.2 Inspiration et extension du concept 15
1.3 Analyse des besoins du marché 15
1.4 Synthèse de l'opportunité 15
1.6 Vers une plateforme logistique universelle 16
1.7 Analyse de la concurrence et positionnement stratégique 16
1.7.1 Analyse concurrentielle du marché de la livraison urbaine 16
1.7.2 Proposition de valeur différenciante 17
1.7.3 Intégration multiservice et avantage stratégique 17
1.8 Flexibilité tarifaire et amélioration des revenus des livreurs 17
Chapitre 2 Concepts et modélisation de l'application 19
Introduction 20
2.1 Système d'affectation intelligente des livreurs 20
2.1.1 Présentation générale du concept 20
2.1.2Méthode de calcul 20
2.1.3 Mécanisme d'affectation 21
2.1.4 Politique de tarification de la livraison 21
2.2. Système intelligent d'affectation des livreurs 22
2.2.1 Objectif principal 22
2.2.2 Justification du choix de la méthode de scoring pondéré 23
2.2.3 Présentation des critères de décision 23
2.2.4Pondération par Jugement d'Experts 28
2.4 Introduction à la conception UML 37
2.4.1 Diagramme de cas d'utilisation 37

2.4.2 Diagrammes de classes 38
2.4.3 Diagrammes de séquence 39
2.5.1 Client 41
2.5.2 Livreur 43
2.5.3 Administrateur 45
2.6 Choix de l'architecture 47
2.6.1 Aspects fonctionnels de l'application 47
2.7 Aspects techniques complémentaires 48
2.7.1 Authentification des utilisateurs 48
2.7.2 Base de données 49
Chapitre 3 l'implémentation de la solution 52
Introduction 53
3.1 Choix technologiques et environnement de développement 53
3.1.1 Présentation de Flutter 53
3.1.2 Présentation de Dart 53
3.2 description 54
3.2.1 Présentation de l'application et fonctionnement du front-end 54
3 .3 Intégration des services Google API 60
3.3.1 Définition générale des API 60
3.3.2 Obtention des API Google 60
3.4 APIs et SDK Google intégrés 60
3.4.1 Google Maps SDK for Android 60
3.4.2 Google Directions API 61
3.4.3 Google Geocoding API 61
3.4.4 Google Places API 62
3.4.5 Google Roads API 63
3.4.6 Google Routes API 64
3.4.7 Google Geolocation API 64
Perspective ..... **Erreur ! Signet non défini.**
Conclusion Générale 76
Annexe 79
Bibliographie 82

LISTE DES TABLEAU

TABLEAU JUGEMENTS LINGUISTIQUES → NOMBRES GRIS POUR LE MEILLEUR CRITERE 31
JUGEMENTS LINGUISTIQUES → NOMBRES GRIS POUR LE PIRE CRITERE 31
MOYENNE PAR CRITERE 36
LIVREURS DISPONIBLES 72
DONNES NORMALISE..... 72

LISTE DES FIGURES

1. Figure 2.1 : Diagramme de cas d'utilisation global de l'application. 38
2. Figure 2.2 : Diagramme de classes modélisant les entités principales (Utilisateur, Commande, Livreur, etc.). 39
3. Figure 2.3 : Diagramme de séquence illustrant le processus de livraison.....40
4. Figure 2.4 : Diagramme de cas d'utilisation spécifique au Client. 42
5. Figure 2.5 : Diagramme de classes pour le Client. 42
6. Figure 2.6 : Diagramme de séquences pour le Client. 43
7. Figure 2.7 : Diagramme de cas d'utilisation spécifique au Livreur. 43
8. Figure 2.8 : Diagramme de classes pour le Livreur. 44
9. Figure 2.9 : Diagramme de séquences pour le Livreur. 44
10. Figure 2.10 : Diagramme de cas d'utilisation spécifique au l'administrateur. 45
11. Figure 2.11 : Diagramme de classes pour l'administrateur. 45
12. Figure 2.12 : Diagramme de séquences pour l'administrateur. 46
13. Figure 3.1 : Interface de démarrage de l'application (choix du rôle : Client ou Livreur). 55
14. Figure 3.2 : Interface de démarrage de l'application. 55
15. Figure 3.3 : Interface client affichant les pick up et destinations. 57
16. Figure 3.4 : Interface client avec une maps.....57
17. Figure 3.5 : Interface client avec détails des livreurs. 57
18. Figure 3.6 : Interface livreur avec liste des commandes entrantes. 58
19. Figure 3.7 : Interface livreur avec paramètres personnels (taux de réussite, ponctualité, etc.). 58
20. Figure 3.8 :Interface administrateur 59
21. Figure 4.1 : Fonctionnement du modèle DQN 71
22. Figure 4.2 : Pseudo code du modelé DQN 74

Liste des abréviations

Abréviation	Signification
API	Application Programming Interface
SDK	Software Development Kit
GPS	Global Positioning System
UI	User Interface
UX	User Experience
JSON	JavaScript Object Notation
HTTP	HyperText Transfer Protocol
AOT	Ahead-Of-Time (compilation)
JIT	Just-In-Time (compilation)
RGPD	Règlement Général sur la Protection des Données
CNIL	Commission Nationale de l'Informatique et des Libertés
IDE	Integrated Development Environment
CRUD	Create, Read, Update, Delete
MVC	Model – View – Controller
ETA	Estimated Time of Arrival
UI/UX	Interface Utilisateur / Expérience Utilisateur

Introduction générale

Dans un contexte où la connectivité et la rapidité sont devenues des exigences majeures, la logistique urbaine se confronte à des défis croissants en matière d'efficacité, de réactivité et de durabilité. La densité croissante des villes, la congestion routière, ainsi que l'essor du e-commerce et des services à la demande imposent de repenser les modèles traditionnels de livraison et de transport urbain.

La problématique à laquelle ce projet cherche à répondre est la suivante : comment concevoir une solution mobile intelligente, agile et multifonctionnelle, capable d'optimiser l'affectation des ressources humaines et matérielles dans un environnement urbain contraint, tout en assurant une qualité de service élevée pour des besoins variés tels que la livraison de repas, de colis ou le transport de personnes ?

C'est dans cette dynamique que s'inscrit ce projet de fin d'études, centré sur la conception d'une start-up innovante proposant une solution mobile dédiée à la livraison et au transport urbain. L'objectif principal du projet est de développer une application mobile multifonctionnelle offrant trois services complémentaires : la livraison de repas, la livraison de colis, et le transport de personnes. Ces services s'appuient principalement sur un réseau de motos et scooters, garantissant une grande rapidité d'exécution dans les environnements urbains denses et embouteillés.

L'originalité de la solution repose sur l'intégration d'un système intelligent d'affectation des commandes, basé sur un algorithme prenant en compte plusieurs critères pondérés : la position géographique, la disponibilité, le temps estimé de livraison, la capacité de charge et l'historique de performance des livreurs. Ce mécanisme vise à optimiser l'efficacité logistique tout en assurant une meilleure expérience utilisateur.

Ce mémoire est structuré en quatre chapitres principaux :

- Le premier chapitre revient sur la genèse du projet, les inspirations initiales, l'analyse des besoins du marché, ainsi qu'une étude critique des solutions concurrentes existantes. Il met en lumière les choix stratégiques et différenciateurs adoptés, notamment en matière de services, de logistique et de tarification.
- Le deuxième chapitre s'attarde sur la conception fonctionnelle et conceptuelle de l'application. Il présente les interfaces client et livreur, les principales fonctionnalités, les diagrammes UML (cas d'utilisation, classes, séquences), ainsi que le système d'affectation intelligent. La politique tarifaire adoptée y est également détaillée, de même que la méthode de pondération définie à partir d'un jugement d'experts du secteur de la livraison.
- Le troisième chapitre aborde l'implémentation technique du projet. Il décrit les choix technologiques (Flutter, Dart), l'architecture logicielle, et l'intégration des services de la Google Maps Platform (Maps SDK, Directions API, Roads API, etc.). Il détaille

également l'usage de Firebase pour la gestion de la base de données, de l'authentification et du suivi en temps réel.

- Enfin, le quatrième chapitre est consacré à l'intégration d'un agent intelligent basé sur un algorithme de type Deep Q-Network (DQN). Ce dernier vise à améliorer de manière autonome le processus d'affectation des commandes, en apprenant à prendre des décisions optimales dans des environnements urbains dynamiques et changeants.

Ce mémoire s'inscrit ainsi à la croisée du développement mobile, de la logistique intelligente et de l'intelligence artificielle appliquée, avec une ambition : proposer une solution agile, évolutive et adaptée aux enjeux actuels de la mobilité urbaine.

Chapitre 1 Contexte et genèse de l'application

1.1 Genèse de l’idée

Ce travail s'inscrit dans le cadre d'une réflexion sur l'optimisation des systèmes de livraison dans les plateformes numériques. Une analyse empirique des services offerts par deux plateformes de référence, Jumia Food et Fast Delivery, a mis en évidence certaines limites récurrentes dans l'affectation des commandes aux livreurs.

Ces limites se traduisent notamment par des retards dans la livraison dus à un dispatching sous-optimal, qui ne tient pas toujours compte de paramètres essentiels tels que la position géographique des livreurs, la densité des commandes ou encore leur disponibilité effective.

Face à ces constats, il apparaît nécessaire d’envisager une approche plus dynamique et adaptative pour la répartition des missions de livraison. L’utilisation d’un système d’affectation en temps réel s’impose alors comme une solution pertinente, permettant de prendre en compte plusieurs variables contextuelles et d’optimiser l’allocation des ressources logistiques.

1.2 Inspiration et extension du concept

En parallèle, une autre observation est venue enrichir le concept initial : pendant les heures de pointe, de nombreuses personnes préfèrent se faire transporter par moto ou scooter pour éviter les embouteillages. Ce mode de transport est rapide, flexible et particulièrement adapté aux zones urbaines encombrées. Cette tendance, de plus en plus marquée dans plusieurs grandes villes, Ce constat a conduit à l’élargissement du concept initial par l’intégration du transport de personnes en moto comme une fonctionnalité supplémentaire à la plateforme.

1.3 Analyse des besoins du marché

Enfin, un autre segment de besoin est ressorti avec force : la livraison urgente de colis pour les petits e-commerçants. Sur plusieurs groupes et réseaux sociaux tels que Facebook, Telegram, WhatsApp ou Viber, on constate une forte demande de livraison express de la part de vendeurs qui n’ont pas le temps ou les moyens logistiques pour se rendre eux-mêmes dans les agences de livraison. Ils passent souvent des réservations manuelles de livreurs pour s’assurer que leurs colis soient livrés rapidement, ce qui montre un manque d’organisation centralisée.

1.4 Synthèse de l’opportunité

Face à ces constats le projet consiste en une application mobile regroupant trois services complémentaires, avec pour fonctionnalités la centralisation et l’optimisation de ces servicesLivraison de repas (type Jumia Food),

- Livraison express de colis pour les particuliers et e-commerçants,
- Transport de personnes par moto, particulièrement en heures de pointe.

La plateforme proposera un système intelligent de gestion des affectations, assurant une expérience fluide pour les clients, une meilleure rentabilité pour les livreurs, et une organisation

optimale des courses.

1.5 Élargissement des services et fonctionnalités clés

En plus des services de livraison classiques et du transport de personnes, la plateforme prévoit d'intégrer une fonction innovante de réservation directe de livreur. Cette option permettra à un utilisateur de réserver un livreur pour une mission personnalisée, en dehors des commandes proposées dans l'application. Par exemple, un client pourra envoyer un livreur effectuer un achat spécifique ou livrer un objet d'un point A à un point B, selon ses propres besoins.
Afin de mieux structurer l'offre, la plateforme intégrera également :

- Des restaurants partenaires, pour passer des commandes de repas en toute simplicité.
- Une mini-supérette virtuelle, permettant de commander des produits de première nécessité.
- Une pharmacie, dédiée aux médicaments ne nécessitant pas d'ordonnance, pour répondre aux besoins urgents de santé.

1.6 Vers une plateforme logistique universelle

Grâce à cette flexibilité et à cette diversité de services, la plateforme ambitionne de devenir un outil logistique universel, capable de répondre à toutes les formes de livraisons possibles, dans toutes les conditions, que ce soit pour :

- des repas chauds,
- des courses de dernière minute,
- des médicaments,
- des colis express,
- ou même des courses personnalisées hors catalogue.

L'ensemble sera géré de manière intelligente, avec une affectation dynamique des livreurs, pour garantir une satisfaction client maximale, tout en optimisant les ressources humaines et logistiques de la plateforme.

1.7 Analyse de la concurrence et positionnement stratégique

1.7.1 Analyse concurrentielle du marché de la livraison urbaine

Le marché de la livraison urbaine est déjà occupé par des plateformes comme Jumia Food et Fast Delivery, qui dominent la livraison de repas. Toutefois, des limites importantes ont été constatées sur le terrain, notamment dans l'organisation logistique et l'attribution des commandes. Jumia se tourne désormais vers une offre logistique plus large, s'éloignant du secteur alimentaire. D'autres acteurs, comme Yassir Food ou Wajeez Food, restent peu structurés et peinent à offrir une expérience compétitive, ce qui les écarte pour l'instant de la concurrence directe.

1.7.2 Proposition de valeur différenciante

La plateforme se distingue par sa capacité à intégrer des commandes issues de canaux variés : téléphone, réseaux sociaux ou messages directs, en plus des commandes via l'application. Cela permet aux restaurateurs de centraliser la gestion des livraisons et de réserver un livreur en temps réel. Concernant les colis, le système inversé proposé — collecte au domicile du client au lieu du dépôt en agence — représente un gain de temps significatif. Cette approche centrée sur la flexibilité améliore nettement l'expérience utilisateur par rapport aux standards actuels du marché.

1.7.3 Intégration multiservice et avantage stratégique

Contrairement à des plateformes comme InDrive, qui se concentrent sur les voitures et négligent les deux-roues, la solution proposée mise sur la polyvalence. Elle combine livraison de repas, de colis et transport de personnes sur une seule interface. Ce modèle permet aux livreurs d'alterner les missions selon la demande, améliorant leur rentabilité et la réactivité du service. En heures de pointe, cette flexibilité et l'usage de scooters assurent une meilleure efficacité dans les zones urbaines denses, renforçant l'intérêt stratégique de la plateforme.

1.8 Flexibilité tarifaire et amélioration des revenus des livreurs

En parallèle, la plateforme intègre un système d'affectation intelligent qui regroupe les commandes ayant des points de départ proches ou des destinations alignées. Cette logique de mutualisation permet de réduire les trajets à vide, d'augmenter la rentabilité des livreurs par tournée, et d'améliorer les délais de livraison. Ce fonctionnement collaboratif bénéficie à l'ensemble de l'écosystème : clients, commerçants et livreurs en tirent chacun un avantage direct.

Conclusion :

Ce premier chapitre a permis de poser les bases du projet en mettant en lumière les constats de terrain, les besoins non satisfaits et les opportunités à saisir sur le marché de la livraison urbaine. Issu d'une expérience personnelle et d'une observation directe des lacunes des plateformes existantes, le projet vise à proposer une solution innovante, flexible et adaptée aux réalités locales.

En centralisant trois services clés (livraison de repas, transport de personnes par deux-roues, et livraison express de colis) sur une seule et même application mobile, la plateforme ambitionne de redéfinir les standards de la logistique urbaine. Elle se distingue non seulement par son système d'affectation intelligent et sa capacité à intégrer des commandes manuelles, mais aussi par une politique tarifaire claire et une gestion optimisée des tournées.

Ces éléments différenciateurs, combinés à une stratégie d'adaptation aux besoins des commerçants et des livreurs, positionnent la plateforme comme une durable face aux acteurs traditionnels du marché. Le chapitre suivant détaillera les aspects techniques et technologiques ayant permis la concrétisation de cette vision, notamment l'architecture de l'application et l'intégration des API Google.

Chapitre 2 Concepts et modélisation de l’application

Introduction

Dans ce chapitre, nous abordons les éléments théoriques essentiels ayant guidé la conception de notre application avant son implémentation technique. Deux axes majeurs y sont développés : le système intelligent d'affectation des commandes et la modélisation UML de l'application.

Le premier axe est consacré à l'élaboration du système d'affectation intelligente des livreurs, cœur fonctionnel de notre plateforme. Ce système repose sur une méthode de décision multicritère pondérée, visant à optimiser en temps réel la répartition des missions de livraison selon des critères tels que la proximité, le temps estimé de trajet, la charge du livreur et son historique de performance. Ce mécanisme a été conçu pour répondre aux exigences de réactivité, d'équité et de performance opérationnelle dans un contexte de logistique urbaine dense.

Le second axe présente la modélisation UML de la solution, qui constitue une étape clé de la phase de conception. Grâce à l'Unified Modeling Language (UML), nous avons structuré les interactions entre les acteurs du système, les entités principales et les scénarios d'utilisation. Cette modélisation a permis d'assurer une vision claire et partagée de l'architecture de l'application, de guider le développement technique à venir, et de garantir la cohérence fonctionnelle de l'ensemble de la solution

2.1 Système d'affectation intelligente des livreurs

2.1.1 Présentation générale du concept

L'application développée dans le cadre de ce projet repose sur un système intelligent de gestion des livraisons. Elle intègre un algorithme d'affectation des commandes aux livreurs, fondé sur une logique multicritère pondérée. Cet algorithme vise à attribuer automatiquement chaque commande au livreur le plus optimal, en tenant compte d'un ensemble de critères déterminants, évalués en temps réel.

Les principaux critères pris en compte dans le processus décisionnel sont les suivants :

- La proximité géographique entre le livreur et le point de collecte,
- La capacité de charge disponible du livreur (en nombre ou volume de colis),
- Le temps estimé de livraison, calculé via la distance ou l'API Google Directions,
- L'historique de performance du livreur (ponctualité, taux de réussite, évaluations).

2.1.2Méthode de calcul

La méthode GBWM (Grey Best-Worst Method) est une approche multicritère d'aide à la décision qui combine la méthode Best-Worst (BWM) avec la théorie des nombres gris, afin de traiter l'incertitude et la variabilité des jugements d'experts.

2.1.3 Mécanisme d'affectation

Dans le cadre de notre application de livraison, nous envisageons la mise en place d'un mécanisme intelligent d'affectation des commandes. L'objectif principal de ce système est d'optimiser la distribution des livraisons entre les livreurs disponibles en temps réel, tout en minimisant les délais, réduisant les trajets à vide, et en maximisant l'efficacité logistique.

Plutôt que d'affecter les commandes de façon statique (un livreur = une commande = un trajet), ce système s'appuie sur une logique d'affectation conditionnelle et contextuelle, en fonction :

- De l'état de progression des trajets en cours,
- Des points de départ et d'arrivée des commandes,
- Des fenêtres temporelles de préparation des colis,
- De la charge actuelle de chaque livreur.

.

Ce système nécessitera l'utilisation combinée de :

- Données GPS en temps réel (via les API de géolocalisation),
- Estimations précises de temps de trajet (Google Routes API),
- Un moteur d'optimisation dynamique, potentiellement basé sur des heuristiques ou de l'intelligence artificielle (algorithmes gloutons, recherche tabou, ou apprentissage par renforcement à long terme).

Objectives du systeme

- Réduction des distances parcourues
- Diminution du temps d'attente des clients
- Réduction de la consommation de carburant
- Maximisation du taux de remplissage des tournées
- Amélioration du service client par une meilleure ponctualité

2.1.4 Politique de tarification de la livraison

Afin de proposer une solution à la fois transparente, juste et optimisée économiquement, notre application adopte une politique de tarification simple et compétitive, adaptée au marché local.

2.1.4.1Tarification des livraisons

Le tarif appliqué aux clients se décompose comme suit :

- Frais de prise en charge (pick-up) : 180 DA fixes.

- Frais de distance : 20 DA par kilomètre parcouru, calculé selon le véritable itinéraire routier et non une estimation en "vol d'oiseau".

Ce modèle offre une plus grande précision tarifaire, contrairement à :

Jumia : qui calcule les trajets en ligne droite (vol d'oiseau), ce qui peut fausser la réalité du trajet et entraîner des pertes pour les livreurs.

Fast Delivery : applique un tarif unique fixe de 250 DA, quel que soit le trajet, ce qui est désavantageux pour les trajets courts ou très longs.

Yassir-FOOD : utilise également un calcul en vol d'oiseau, ce qui peut créer des écarts tarifaires importants avec la réalité du terrain.

2.1.4.2Politique de commission sur les transactions

Le modèle économique de notre plateforme repose sur un partage équitable des revenus, réparti entre l'application, le commerçant et le livreur.

Commission sur les commerçants :
Une commission de 10 % est prélevée sur chaque commande traitée via l'application, ce qui reste raisonnable comparé aux standards du marché.

Commission sur les livreurs :
Une commission de 5 % est appliquée sur les gains générés par les livreurs, dans le but de soutenir les frais de fonctionnement de la plateforme tout en respectant leur rentabilité.

À titre de comparaison :

De nombreuses applications concurrentes prélèvent jusqu'à 20 % de commission globale, sur le commerçant, ce qui réduit considérablement leur marge.

Notre approche vise à encourager les partenariats durables et à fidéliser les utilisateurs professionnels (commerçants et livreurs) en garantissant un meilleur équilibre économique.

2.2. Système intelligent d'affectation des livreurs

2.2.1 Objectif principal

L'objectif de ce système d'affectation intelligente est de maximiser l'efficacité opérationnelle dans le cadre des livraisons à la demande. Concrètement, il s'agit d'attribuer automatiquement chaque nouvelle commande à un livreur jugé optimal, en fonction de critères déterminés dynamiquement et en temps réel.

Ce système vise à:

- Réduire les délais de livraison.
- Améliorer l'équité entre les livreurs.
- Optimiser les trajets et la consommation de carburant.
- Renforcer la satisfaction des clients finaux.
- Diminuer les cas d'échec de livraison ou de retards.

2.2.2 Justification du choix de la méthode de scoring pondéré

Afin de garantir une affectation optimale des commandes de livraison, nous avons opté pour une méthode de décision multicritère fondée sur un **sy**stème de scoring pondéré. Ce choix s'explique par la nécessité de prendre en compte plusieurs facteurs à la fois (temps, distance, capacité, performance...) dans un environnement dynamique et en temps réel.

Contrairement à une règle de décision unique ou un ordre de priorité fixe, la méthode du scoring pondéré permet d'évaluer chaque livreur selon plusieurs critères pondérés et de calculer un score global pour chaque candidature. Cela rend le processus d'affectation plus juste, objectif et adaptable aux contextes réels de la logistique urbaine.

Cette méthode est aussi simple à implémenter techniquement et parfaitement compatible avec les structures de données de Firestore, tout en étant suffisamment flexible pour intégrer de nouveaux critères à l'avenir.

2.2.3 Présentation des critères de décision

L'algorithme s'appuie sur un ensemble de **critères clés** permettant de déterminer quel livreur est le plus adapté à chaque commande. Voici une description détaillée de chaque critère utilisé

2.2.3.1 Proximité géographique

Définition :
La proximité géographique représente la distance entre le livreur et le point de collecte de la commande (point de pick-up). Ce critère est déterminant dans le système d'attribution car un livreur situé plus près permet une prise en charge plus rapide, réduit les délais d'attente pour le client et optimise l'efficacité de la tournée. Ce facteur influence directement la satisfaction client, la consommation de carburant, et la gestion globale du temps.

Fonctionnement :
Deux méthodes sont disponibles pour calculer cette distance :

- La formule de Haversine permet une estimation rapide de la distance à vol d'oiseau entre deux points géographiques (latitude/longitude). Cette méthode est utile pour un premier filtrage approximatif des livreurs.

- L’API Directions de Google Maps, quant à elle, a été privilégiée dans ce projet pour le calcul final. Contrairement à la Haversine, cette API prend en compte les conditions réelles de circulation, les routes disponibles, les sens de circulation, les barrages éventuels ou zones inaccessibles, offrant ainsi une précision bien supérieure pour les trajets routiers.

Justification du choix de l’API Directions :

Bien que la formule de Haversine soit légère et rapide, elle ne reflète pas la réalité du terrain. Dans une application de livraison, où les délais et la précision sont cruciaux, il est indispensable de considérer les données de trafic en temps réel et l'accessibilité routière réelle. L’API Directions répond parfaitement à ces exigences, en offrant une distance routière exacte, intégrable directement dans le scoring pondéré. Elle permet aussi de récupérer le temps estimé de trajet, utilisé dans un autre critère de sélection. C’est donc une solution complète et fiable pour la logistique urbaine dynamique.

2.2.3.2 Capacité de charge

Définition :

La capacité de charge correspond à la quantité maximale de colis ou au volume total qu’un livreur peut transporter en une seule tournée. Elle dépend de plusieurs facteurs tels que le moyen de transport utilisé (sac à dos, scooter, voiture, triporteur), la politique de sécurité (éviter les surcharges) et les conditions de livraison spécifiques (fragilité, taille, température, etc.). Ce critère est crucial pour assurer un bon déroulement des livraisons sans incidents ni retards, tout en maintenant un niveau de qualité de service satisfaisant.

Fonctionnement :

Chaque profil de livreur enregistré dans la base de données (ici Firestore) contient un champ maxCapacity, qui représente sa capacité totale, et un champ currentLoad, qui indique le nombre actuel de colis qu’il transporte ou qu’il a déjà acceptés.

Pourquoi c’est important :

En respectant la capacité de chaque livreur, on évite :

- Des retards de livraison dus à une surcharge.
- Des colis endommagés ou refusés.
- Une baisse de motivation ou de performance des livreurs.
- Des situations de service non conforme.

Ce critère contribue à une répartition efficace et durable du travail, garantissant la fiabilité du système logistique.

2.2.3.3 Temps estimé de livraison

Définition :
Le temps estimé de livraison représente la durée totale prévisionnelle nécessaire pour qu'un livreur effectue une mission, depuis sa position actuelle jusqu'au point de collecte (pick-up), puis du point de collecte jusqu'au point de livraison (drop-off). Il est un indicateur clé de performance logistique, car il permet :

- De réduire les délais de livraison,
- D'optimiser les tournées,
- Et de garantir la satisfaction des clients.

Ce critère est essentiel dans le système d'affectation, car il permet de privilégier le livreur le plus rapide pour une mission donnée, ce qui contribue à une meilleure réactivité globale.

Fonctionnement :
Le calcul du temps estimé s'effectue en deux étapes :

1. Temps de trajet entre le livreur et le point de collecte.
2. Temps de trajet entre le point de collecte et le point de livraison.

Pour obtenir ces durées de manière fiable, l'API Google Directions est utilisée. Cette API a été choisie plutôt que la formule de Haversine, car :

- Elle tient compte du trafic routier en temps réel.
- Elle offre des estimations fiables et adaptées selon le mode de transport (voiture, moto, vélo…).
- Elle fournit des informations détaillées comme la durée exacte, la distance, et le tracé optimal.

2.2.3.4 Historique de performance

Définition :
L'historique de performance est un indicateur composite qui évalue la qualité, la fiabilité et la régularité d'un livreur sur la base de ses missions précédentes. Il regroupe plusieurs métriques comportementales et opérationnelles qui reflètent son sérieux, son efficacité et son professionnalisme.

Ce critère a pour objectif :

De favoriser les livreurs fiables dans le processus d'affectation,

De garantir une meilleure expérience client,

D'encourager les comportements vertueux (ponctualité, rigueur, professionnalisme).

Fonctionnement :
L'évaluation de l'historique de performance repose sur un calcul de score global à partir de plusieurs sous-critères pondérés. Voici les principaux éléments pris en compte :

- Taux de réussite des livraisons (successRate)

Nombre de commandes livrées / Nombre total de commandes acceptées
Ex. : 92 commandes réussies sur 100 = 92 %

- Taux de ponctualité (onTimeRate**)**

  Pourcentage des livraisons effectuées dans le temps estimé.
  Cela reflète la capacité du livreur à respecter les délais annoncés.

- Évaluation moyenne des clients (averageRating)

  Note moyenne attribuée par les clients après la livraison (sur 5 étoiles).
  Elle reflète la qualité du service perçue : politesse, communication, présentation…

- Temps de réponse moyen (responseTime)

  Temps entre l'envoi d'une commande au livreur et son acceptation.
  Plus ce temps est court, plus le livreur est réactif et engagé.

Méthode d'intégration dans l'algorithme
Chaque sous-critère est converti en score numérique normalisé (entre 0 et 1) et multiplié par un poids pondéré défini selon son importance (par exemple : 40 % pour le taux de réussite, 30 % pour la ponctualité, 20 % pour la note client, 10 % pour le temps de réponse). Le total forme un score global de performance :

$$PerformanceScore = (0.4 \times successRate) + (0.3 \times onTimeRate) + (0.2 \times averageRating) + \big(0.1 \times (1 - normalizedResponseTime)\big)$$

Ce score est ensuite combiné avec les autres critères (proximité, capacité, etc.) dans le calcul final du score pondéré de chaque livreur.

2.2.3.5 Équilibrage de la charge (Fairness)

Définition

L'équilibrage de la charge consiste à répartir les opportunités de livraison de manière équitable entre les livreurs actifs. Il vise à éviter qu'un petit groupe de livreurs – souvent les plus performants – ne reçoive systématiquement toutes les commandes, au détriment des autres. Ce critère permet ainsi de maintenir un bon niveau de motivation, de prévenir la surcharge, et de garantir un traitement juste au sein du réseau de distribution.

Il s'agit donc d'un facteur social et organisationnel essentiel à la durabilité du système de livraison.

Fonctionnement

Pour mesurer l'équilibre de la charge de travail, le système prend en compte plusieurs indicateurs clés :

- Nombre de livraisons déjà effectuées dans la journée (deliveriesToday)
  → Plus ce nombre est élevé, plus la priorité du livreur diminue, sauf cas exceptionnel.
- Temps cumulé d'activité (activeTimeToday)
  → Total des minutes/heures où le livreur a été en statut actif (connecté, en mission, en déplacement).
- Temps d'attente depuis la dernière affectation (waitingTime)
  → Permet de prioriser les livreurs en attente depuis longtemps, favorisant une répartition équitable.
- Score de charge pondéré (loadScore)
  → Calculé comme une combinaison des éléments ci-dessus selon des poids définis.

Exemple simplifié de score de charge :

$$\text{loadScore} = (0.5 \times \text{deliveriesToday}) + (0.3 \times \text{activeTimeToday}) - (0.2 \times \text{waitingTime})$$

Ce score peut être **inversé ou normalisé** pour favoriser les livreurs les moins sollicités (score faible = priorité haute).

Intégration dans l'algorithme

Le score de charge est intégré dans la formule globale de pondération avec les autres critères (proximité, performance, capacité, etc.). Un livreur avec un excellent score global mais déjà surchargé verra son score total ajusté à la baisse, au profit d'un autre livreur ayant un profil légèrement inférieur mais moins sollicité.

Avantages de ce critère

- Assure une répartition juste du travail, évitant les déséquilibres.
- Réduit la fatigue et les erreurs dues à la surcharge.
- Maintient la motivation et la fidélité des livreurs.
- Améliore la performance collective du système de livraison.

2.2.4Pondération par Jugement d'Experts

Description générale

La pondération par jugement d'expert est une méthode qualitative permettant d'attribuer des poids relatifs aux différents critères d'un système décisionnel, sur la base de l'expérience, de l'intuition métier et des observations terrain des professionnels du secteur. C'est une méthode souvent utilisée lorsque les données quantitatives ne sont pas encore disponibles ou lorsque le modèle repose fortement sur la connaissance métier.

2.2.4.1Étapes de mise en œuvre dans le cadre du projet

A . Identification des experts

Les experts sélectionnés pour la pondération des critères sont issus de divers segments du secteur de la livraison urbaine. Le choix s'est fondé sur leur expérience directe dans la gestion ou l'exécution des opérations logistiques. L'échantillon comprend 5 profils : un responsable régional d'une plateforme de livraison de repas (Jumia Food), un coordinateur logistique d'une entreprise de livraison rapide (Fast Delivery), un opérateur terrain de Yassir Food, un ancien gérant de restaurant à service rapide, ainsi qu'un livreur indépendant cumulant plus de 5 ans d'expérience.

Bien que limité en nombre, cet échantillon vise à refléter une diversité fonctionnelle (stratégique, opérationnelle et exécutive). Les données ont été collectées via des entretiens semi-directifs, au cours desquels chaque expert a été invité à pondérer quatre critères clés sur une échelle de 1 à 5. Les pondérations finales ont été obtenues par normalisation des moyennes.

Cette approche s'inspire des méthodes de jugement d'experts utilisées en décision multicritère (telles que la méthode SMART), particulièrement appropriées lorsque les données quantitatives sont insuffisantes ou indisponibles.

b. Organisation d'ateliers de discussion

Des entretiens semi-directifs ou des réunions collectives ont été organisés pour comprendre les priorités opérationnelles :

- Quelle importance accorder à la rapidité vs la fiabilité ?

- Quelle place donner à la performance passée vs la situation actuelle ?
- Comment éviter la surcharge ou les affectations injustes ?

c. Méthode de calcul GBWM

La méthode Best-Worst (BWM) est une approche d'aide à la décision multicritère qui permet de déterminer les poids des critères en demandant aux experts d'identifier le critère "meilleur" (le plus important) et le critère "pire" (le moins important). Ensuite, une série de comparaisons est réalisée entre ces extrêmes et l'ensemble des autres critères, ce qui génère deux vecteurs de préférences. Ces informations sont utilisées pour résoudre un problème d'optimisation linéaire, ce qui permet d'obtenir des poids cohérents et robustes pour chaque critère

d. Identification des critères de décision

On commence par lister les critères qui seront utilisés pour évaluer les alternatives Chaque expert identifie, selon lui :

- Le meilleur critère (B) : celui qui est le plus important ou le plus favorable.
- Le pire critère (W) : celui qui est le moins important.

e. Comparaison directe entre le meilleur et le pire critère

Pour la méthode GBWM, il est possible de se limiter à une seule comparaison : celle entre le meilleur critère et le pire critère, sans évaluer tous les autres.

L'expert répond simplement à la question :
« Dans quelle mesure le meilleur critère est-il plus important que le pire ? »

Par exemple, s'il estime que le meilleur critère est 7 fois plus important que le pire, on exprime cette préférence par le rapport suivant :

$$W_{best} / W_{worst} = 7$$

Cette approche permet de réduire considérablement le nombre de jugements nécessaires, tout en conservant une base logique pour l'établissement des pondérations. Par la suite, des hypothèses d'estimation ou des modèles d'interpolation peuvent être utilisés pour déterminer les poids des critères intermédiaires.

f. Avantages de la méthode GBWM

- Réduction du nombre de comparaisons :Contrairement à d'autres méthodes multicritères comme l'AHP (Analytic Hierarchy Process), GBWM ne nécessite pas de comparer tous les critères entre eux. Seules les comparaisons entre le meilleur et le pire, sont requises.
- Cohérence accrue des jugements :Le modèle repose sur une optimisation qui minimise l'incohérence dans les jugements fournis. Cela permet d'obtenir des pondérations plus stables et fiables, même en présence d'experts multiples
- Prise en compte de plusieurs experts :GBWM est conçu pour agréger les jugements de plusieurs experts, en les combinant pour produire un résultat collectif.[26]

g. Exemple de calcul

Étape 1 : Définir la liste des critères

Les critères utilisés pour évaluer l'adéquation d'un livreur sont :

- Proximité géographique
- Capacité de charge
- Historique de performance
- Temps estimé de livraison

Étape 2 : Choix du meilleur et du pire critère (par consensus ou majorité)

D'après les notes moyennes :

- Meilleur critère (Best) : Temps estimé de livraison
- Pire critère (Worst) : Capacité de charge

Étape 3 : Jugements experts (synthèse intuitive)

Les experts comparent

- Best vs autres critères : à quel point le temps de livraison est plus important que chaque autre critère.
- Autres critères vs Worst : à quel point chaque critère est plus important que la capacité de charge.

Tableau jugements linguistiques → nombres gris pour le meilleur critère

Comparaison	Jugement linguistique	Nombres gris (approximatifs)
Temps livraison vs Proximité	Légèrement à modérément plus important	[1, 2]
Temps livraison vs Performance	Modérément plus important	[2, 3]
Temps livraison vs Capacité	Très fortement plus important	[5, 6]

jugements linguistiques → nombres gris pour le pire critère

Comparaison	Jugement linguistique	Nombres gris (approximatifs)
Proximité vs Capacité	Fortement plus important	[4, 5]
Performance vs Capacité	Modérément plus important	[2, 3]
Temps livraison vs Capacité	Très fortement plus important	[5, 6]

Étape 4 : Résolution du modèle GBWM

Afin de résoudre ce problème d'optimisation basé sur la méthode GBWM, nous avons utilisé le modèle mathématique suivant, qui permet de traduire les jugements des experts sous forme de contraintes linéaires. Ce modèle cherche à déterminer les poids w_j des critères tout en minimisant une variable d'incohérence ξ, représentant l'écart maximal entre les jugements exprimés et les rapports effectifs entre les poids.

h. modèle mathématique

Minimiser ξ
Sous contraintes

$$(w_Best \,/\, w_j) \;-\; a_Bj \;|\; \leq\; \xi \; pour \; tout \; j$$
$$(w_j \,/\, w_Worst) \;-\; a_jW \;|\; \leq\; \xi \; pour \; tout \; j$$
$$Somme \; des \; w_i = \; 1 \; et \; w_i \geq \; 0$$

Notation
• w_j : le poids du critère j qu'on cherche à déterminer
• w_Best : le poids du meilleur critère (le plus important selon les experts)
• w_Worst : le poids du pire critère (le moins important)
• a_Bj ∈ [a_Bj^L, a_Bj^U] : le jugement gris du meilleur critère par rapport au critère j
• a_jW ∈ [a_jW^L, a_jW^U] : le jugement gris du critère j par rapport au pire critère

• ξ : la variable d'incohérence qu'on cherche à minimiser

Formulation du modèle GBWM (optimisation linéaire)

L'objectif est de trouver les poids *w_j* qui respectent les comparaisons grises fournies par les experts, tout en minimisant l'incohérence ξ. Le modèle se formule ainsi :

1. Contraintes du meilleur critère :

$$\left| \left(\frac{w_{Best}}{w_j} \right) - a_{Bj} \right| \leq \xi \quad pour\ tout\ j$$

Cela signifie :

Le rapport entre le poids du meilleur critère et celui du critère j doit être aussi proche que possible de l'évaluation grise a_Bj.

2. Contraintes du pire critère :

$$\left| \left(\frac{w_j}{w_{Worst}} \right) - a_{jW} \right| \leq \xi \quad pour\ tout\ j$$

Cela signifie :

Le rapport entre le poids de j et celui du pire critère doit être cohérent avec le jugement a_jW.

3. Normalisation :

$$\sum w_i = 1 \quad et \quad w_i \geq 0$$

On as :

- *Temps de livraison* = meilleur critère
- *Capacité de charge* = pire critère

Alors, par exemple :

- Si *a_Bj*, *Proximité* = [1, 2] alors $\frac{w_{Best}}{w_j} \in [1, 2]$
- Si *a_jW*, *Proximité* = [4, 5] alors $\frac{w_j}{w_{Worst}} \in [4, 5]$

Afin d'obtenir les résultats de la méthode GBWM, nous avons utilisé un script Python, car les calculs mathématiques réalisés manuellement sont à la fois chronophages et moins précis. L'automatisation permet ainsi de gagner du temps et d'assurer une plus grande fiabilité dans le traitement des données.

Résultats d'analyse des poids des critères

Critère	**Symbole**	**Poids**
Temps estimé de livraison	w_1	0.4615
Proximité géographique	w_2	0.3077
Historique de performance	w_3	0.1538
Capacité de charge	w_4	0.0769
Incohérence	ξ	0.0

Interprétation

Les résultats du modèle GBWM montrent que le critère le plus important est le temps estimé de livraison, représentant 46,15 % de l'importance globale. Cela signifie que la rapidité est la priorité principale dans le choix d'un livreur. Il est suivi par la proximité géographique avec 30,77 %, soulignant l'importance de la localisation du livreur pour garantir une livraison rapide.

L'historique de performance obtient 15,38 %, ce qui indique que la fiabilité passée du livreur est prise en compte, mais reste secondaire. Enfin, la capacité de charge, avec seulement 7,69 %, est jugée comme le critère le moins déterminant dans ce contexte, probablement en raison de la nature légère des colis.

L'incohérence étant de 0,0, cela confirme que les jugements des experts étaient parfaitement cohérents.

i. scoring pondéré

La méthode de scoring pondéré consiste à affecter à chaque critère un coefficient d'importance ,Chaque critère est associé à un coefficient de pondération (ou poids relatif) représentant son importance dans la décision globale. Par exemple, la proximité peut être jugée plus déterminante que la note de performance dans certains contextes, et inversement dans d'autres. L'algorithme calcule alors pour chaque livreur un score global pondéré, obtenu selon la formule :

$$Score = \sum(i = 1)^n (w_i \times c_i)$$

Où :

w_i est le poids affecté au critère

c_i est la valeur mesurée de ce critère pour le livreur considéré.

Ainsi, chaque livreur se voit attribuer un score personnalisé, calculé dynamiquement en fonction de la commande à traiter et de la situation actuelle (localisation, charge en cours, antécédents, etc.). L'ordre est alors affecté au livreur ayant obtenu le score le plus élevé.

Cette approche pondérée et dynamique garantit une répartition plus juste, plus efficace et plus contextuelle des commandes, en prenant en compte à la fois les préférences opérationnelles et les réalités du terrain. Ce système permet également d'éviter les déséquilibres d'affectation, en valorisant les livreurs disponibles, performants et proches du point de départ.

Pour évaluer les livreurs de manière objective, chaque critère est converti en score numérique (sur une échelle de 1 à 5, ou 0 à 1) afin de faciliter la comparaison et l'analyse multicritère.

A. Proximité géographique

Principe de scoring : Plus la distance est courte, plus le score est élevé.
Barème proposé :

Distance	Score
< 1 km	5
1 – 3 km	4
3 – 5 km	3
> 5 km	1

B. Capacité de charge

Principe de scoring:

Gradué possible : Si le système prend en compte la capacité restante, un score sur 5 peut être envisagé.

C. Temps estimé de livraison

Principe de scoring : Moins le temps est élevé, plus le score est haut.

Barème proposé:

$$N_{IJ} = \frac{x_{jmax} - x_{ij}}{x_{jmax} - x_{jmin}} \times 5$$

Element dans la formule	Signification
N_{IJ}	Valeur normalisée pour le livreur i sur le critère j
x_{ij}	Valeur réelle du livreur i pour le critère j
x_{jmax}	Valeur maximale parmi tous les livreurs pour le critère j
x_{jmin}	Valeur minimale parmi tous les livreurs pour le critère j

[28]

D. Historique de performance

Sous-critères utilisés :

- Taux de succès des livraisons (% livraisons sans incident)
- Ponctualité (% livraisons à l'heure)
- Note moyenne des clients (1 à 5)

- Principe de scoring : Les sous-critères sont combinés pour produire une note unique (souvent sur 5).

Exemple de calcul

A. Calculer le Score Total pour Chaque Livreur

Pour chaque livreur qui a passé le filtre initial, vous multipliez le score de chaque critère par son poids respectif, puis vous additionnez tous ces résultats.

La formule est la suivante :

$$Score\ Total\ du\ Livreur = i = 1\sum n(Score\ du\ Crite\grave{}re\ i \times Poids\ du\ Crite\grave{}re\ i)$$

On obtient

Moyenne par critère

Livreur	Proximité (5 = très proche)	Capacité (5 = grande)	Performance (5 = fiable)	Temps estimé (5 = très rapide)
A	4	3	4	5
B	2	5	3	3
C	5	2	5	4

Livreur A :
Temps estimé de livraison : 5 × 0.4615 = 2.3075
Proximité géographique : 4 × 0.3077 = 1.2308
Historique de performance : 4 × 0.1538 = 0.6152
Capacité de charge : 3 × 0.0769 = 0.2307
Score total Livreur A = 2.3075 + 1.2308 + 0.6152 + 0.2307 = 4.3842

Livreur B :
Temps estimé de livraison : 3 × 0.4615 = 1.3845
Proximité géographique : 2 × 0.3077 = 0.6154
Historique de performance : 3 × 0.1538 = 0.4614
Capacité de charge : 5 × 0.0769 = 0.3845
Score total Livreur B = 1.3845 + 0.6154 + 0.4614 + 0.3845 = 2.8458

Livreur C :
Temps estimé de livraison : 4 × 0.4615 = 1.8460
Proximité géographique : 5 × 0.3077 = 1.5385
Historique de performance : 5 × 0.1538 = 0.7690
Capacité de charge : 2 × 0.0769 = 0.1538
Score total Livreur C = 1.8460 + 1.5385 + 0.7690 + 0.1538 = 4.3073

B. Interprétation

Le Livreur A est le meilleur choix selon les pondérations et les évaluations, avec un score total de 3.7688, légèrement supérieur à celui des autres livreurs. Il se distingue particulièrement par une excellente performance en temps estimé de livraison (note 5), qui est le critère le plus fortement pondéré. Cela montre qu'il est susceptible d'effectuer la livraison plus rapidement, ce qui en fait la meilleure option globale dans ce cas.

2.4 Introduction à la conception UML

La conception de notre plateforme repose sur une approche structurée et modulaire pour garantir une gestion optimale des données, des utilisateurs et des interactions dans le système. L'UML (Unified Modeling Language) est un outil central dans cette démarche, permettant de modéliser les processus, d'identifier les composants essentiels et de faciliter la communication entre les développeurs, les parties prenantes et les utilisateurs finaux.
L'utilisation d'UML permet de clarifier les fonctionnalités clés de notre application et de définir de manière précise la manière dont ces fonctionnalités interagiront au sein du système. Les diagrammes UML choisis pour ce projet incluent les principaux types suivants :

- Diagrammes de cas d'utilisation : pour décrire les interactions entre les utilisateurs et le système.
- Diagrammes de classes : pour modéliser la structure statique du système en termes de classes, d'attributs et de relations.
- Diagrammes de séquence : pour visualiser l'ordre des interactions entre les différents composants pendant l'exécution des processus clés.

L'objectif est de proposer une solution claire et compréhensible qui puisse être facilement modifiable et extensible en cas d'ajout de nouvelles fonctionnalités ou d'évolution des besoins des utilisateurs.

2.4.1 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation présenté ci-dessous illustre les principales interactions entre les différents acteurs de la plateforme de livraison et les fonctionnalités offertes par le système. Quatre types d'acteurs sont identifiés :

- Le Client, qui utilise l'application pour passer une commande, réserver un livreur, et suivre la livraison en temps réel.
- Le Livreur, qui reçoit automatiquement des commandes via un système intelligent d'affectation.
- L'Administrateur, chargé de la gestion globale et du bon fonctionnement de la plateforme.
- Le Commerçant, qui peut publier et gérer ses produits ou services à livrer, ainsi que le suivi de ses commandes.

L'application propose plusieurs fonctionnalités essentielles pour assurer une expérience complète aux différents utilisateurs. Le client peut passer une commande, qu'il s'agisse d'un repas ou d'un colis, en sélectionnant le produit ou en indiquant les détails de l'envoi via l'interface dédiée. Il lui est également possible de réserver un livreur pour des livraisons ponctuelles, à une date et un lieu déterminés, notamment dans le cadre d'envois personnalisés entre particuliers. Une fois la commande validée, le système met en œuvre un algorithme d'affectation intelligente qui attribue automatiquement la livraison au livreur le plus optimal, en fonction de critères tels que la disponibilité, la proximité géographique ou encore la capacité de charge. Par ailleurs, le client bénéficie d'un service de suivi en temps réel, qui lui permet de localiser précisément son livreur tout au long du trajet. Du côté administratif, un espace est

réservé à l'administrateur, lequel dispose d'outils de gestion des utilisateurs (clients, livreurs, commerçants), de supervision du bon fonctionnement de la plateforme, et de traitement des éventuels litiges. Enfin, les commerçants peuvent gérer leur catalogue de produits ou de services (par exemple les menus dans le cas d'un restaurant), suivre les commandes en cours et interagir avec les autres acteurs via leur interface dédiée.

Ce diagramme permet de visualiser rapidement les interactions essentielles du système, en posant les bases fonctionnelles pour le développement de l'application. Il joue un rôle clé dans la conception du système, en orientant les choix techniques et l'architecture logicielle à mettre en œuvre

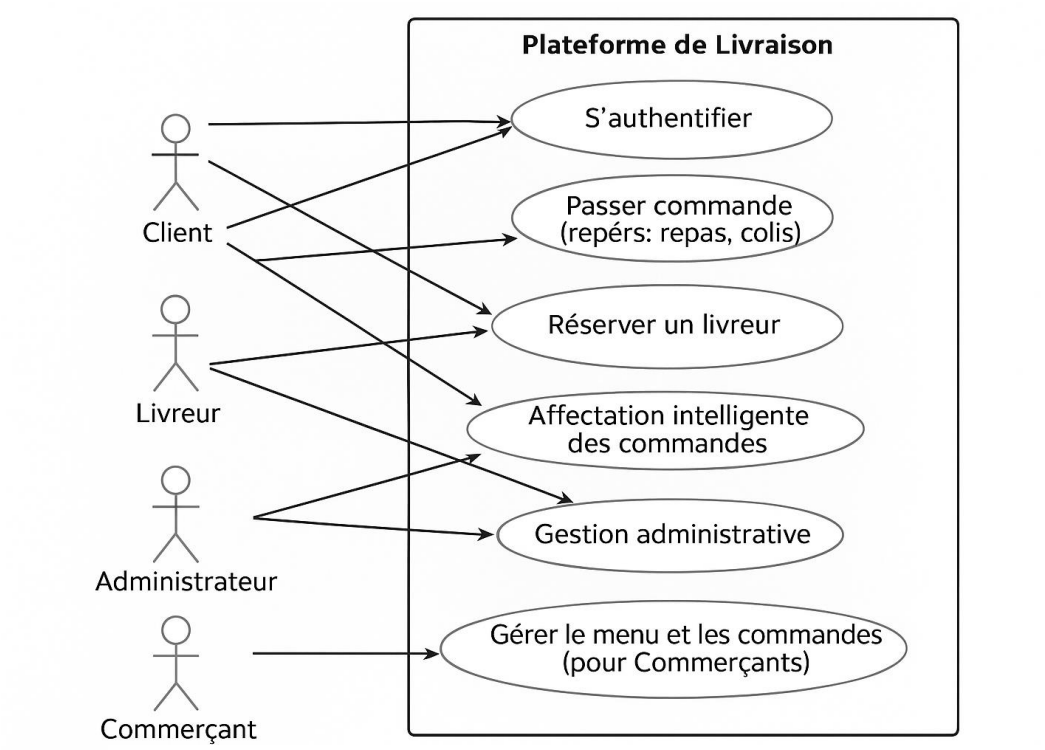


Figure 2.1 Diagramme de cas d'utilisation global de l'application.

Ce diagramme UML illustre les principales fonctionnalités de la plateforme de livraison, en lien avec ses utilisateurs : client, livreur, administrateur et commerçant. Il montre comment chaque acteur interagit avec le système (commande, gestion, suivi, etc.). L'ajout de l'authentification renforce la sécurité et permet un accès personnalisé selon le profil de l'utilisateur.

2.4.2 Diagrammes de classes

Le diagramme présente la modélisation des données d'une application de livraison, organisée autour de l'entité centrale Utilisateur. Cette entité générique comprend des attributs de base tels que l'identifiant, le nom, l'email et le numéro de téléphone. Elle se décline en plusieurs sous-types spécialisés, chacun doté d'attributs spécifiques :

- Livreur : caractérisé par sa disponibilité (booléen) et sa capacité de transport (entier).
- Véhicule : défini par son type (moto/scooter) et son immatriculation.

- Commerçant : associé à un nom d'entreprise et un type d'activité.
- Client : distingué par son adresse de livraison.

Les relations entre ces entités sont implicites à travers les verbes livre, propose et passer, qui décrivent les interactions principales du système (ex. : un livreur livre une commande, un commerçant propose des produits).

L'entité Commande est décrite avec précision :

- Attributs : identifiant, type (repas/colis/transport), date, statut et tarif.
- Méthode : calculerTarif(), suggérant une logique métier pour déterminer le coût.

Ce modèle sert de base technique pour le développement, en définissant la structure de la base de données et les relations entre les acteurs. Il complète les aspects fonctionnels en fournissant le cadre nécessaire à l'implémentation des fonctionnalités telles que la gestion des commandes ou la tarification.

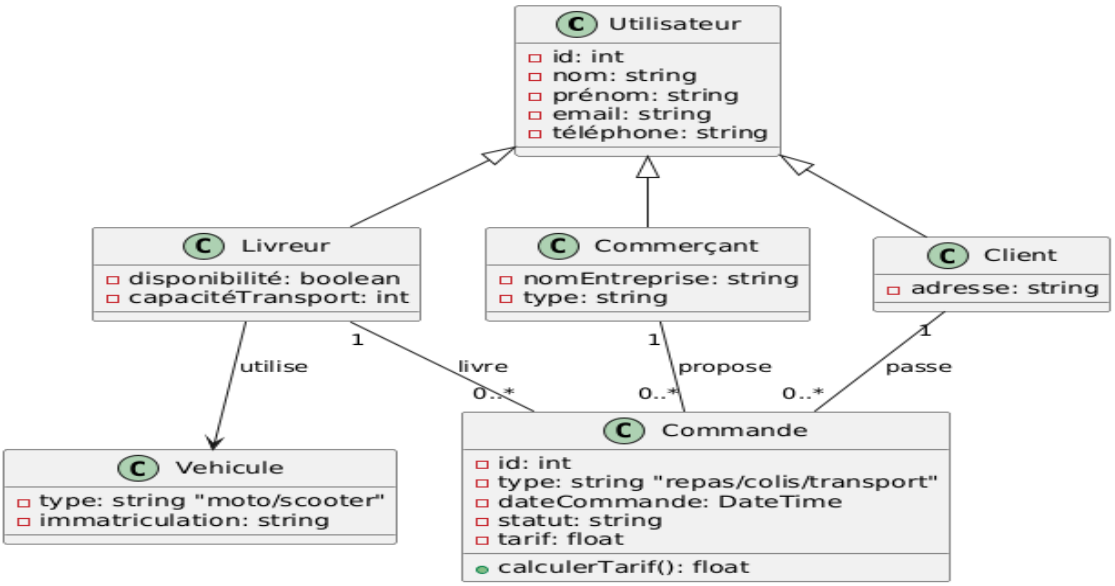


Figure 2.2 Diagramme de classes modélisant les entités

Ce schéma présente une structure basique de base de données pour une application de livraison, reliant utilisateurs (clients, livreurs, commerçants), véhicules et commandes. Bien que fonctionnelle, elle manque d'éléments clés comme la géolocalisation, les paiements et les évaluations, nécessaires pour une solution complète.

2.4.3 Diagrammes de séquence

Ce diagramme illustre le processus complet de livraison dans une application mobile, depuis la passation de commande jusqu'à la confirmation finale. Le flux met en évidence les interactions entre quatre composants principaux :

Le processus global de livraison débute avec le client, qui initie l'opération depuis l'application mobile en soumettant une commande comportant toutes les informations

nécessaires, telles que le type de livraison, les coordonnées, et les détails logistiques. L'application mobile joue un rôle central en réceptionnant la commande, en la transmettant au système d'affectation, en affichant les notifications et le suivi en temps réel, puis en fournissant les confirmations finales au client. Dès réception de la commande, l'algorithme d'affectation entre en action : il vérifie la disponibilité des livreurs, analyse la compatibilité des itinéraires existants, sélectionne le livreur le plus optimal selon des critères prédéfinis, puis gère automatiquement l'envoi des notifications correspondantes. Le livreur, de son côté, reçoit la commande, l'accepte, puis met à jour régulièrement le statut de la livraison jusqu'à sa confirmation finale une fois la mission accomplie. Enfin, un système de suivi intégré enregistre toutes les affectations réalisées, surveille en continu l'état d'avancement des livraisons et génère les mises à jour en temps réel, assurant ainsi une transparence complète à toutes les étapes du cycle de livraison.

Le processus suit une séquence logique :

1. Le client passe commande via l'application
2. L'algorithme affecte la commande à un livreur disponible
3. Le livreur accepte et commence la livraison
4. Le système suit et met à jour la progression
5. La confirmation finale clôt le cycle

Chaque étape est accompagnée d'échanges d'informations précis (données de commande, notifications, statuts) qui assurent la synchronisation entre tous les acteurs. Ce flux optimisé permet une expérience utilisateur transparente tout en garantissant l'efficacité opérationnelle.

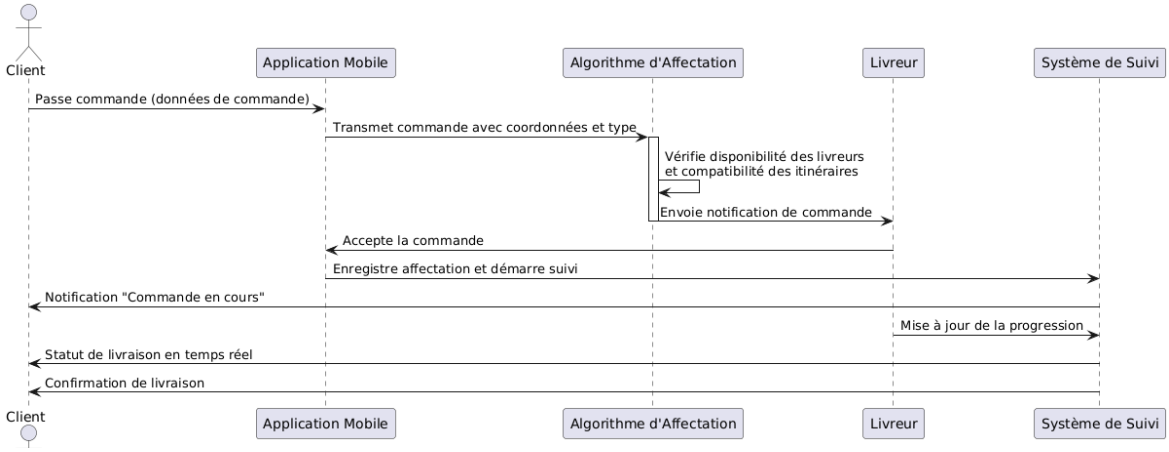


Figure 2.3 Diagramme de séquence illustrant le processus de livraison

Le client passe commande via l'application, déclenchant l'algorithme d'affectation qui sélectionne le livreur optimal en fonction de la localisation et des disponibilités. Le livreur reçoit la demande, accepte la mission et le système active le suivi GPS en temps réel. Tout au long du trajet, le client peut suivre l'avancée de sa livraison jusqu'à la confirmation finale. Ce flux optimisé garantit rapidité et transparence pour tous les acteurs.

2.5 Modélisation des Interactions des Acteurs

Introduction

Dans cette étape cruciale, nous passons à la modélisation des interactions spécifiques aux acteurs clés de notre plateforme : le Client, le Livreur et le Vendeur (commerçant). À travers l'utilisation des diagrammes de cas d'utilisation UML, nous visons à clarifier et structurer le rôle de chaque acteur dans le système ainsi que leurs interactions avec l'application.

Objectives

La modélisation fonctionnelle de l'application vise avant tout à garantir une compréhension partagée entre l'ensemble des intervenants du projet, qu'il s'agisse des développeurs, des concepteurs ou des encadrants. Elle permet de fournir une vision claire et commune des fonctionnalités attendues ainsi que du déroulement opérationnel global, allant de l'inscription de l'utilisateur jusqu'à la finalisation des transactions. Elle contribue également à la définition précise des périmètres fonctionnels en délimitant les responsabilités propres à chaque acteur et en identifiant les besoins fonctionnels, les interactions et les flux d'information au sein de la plateforme. Enfin, cette modélisation sert de socle pour l'optimisation de l'architecture logicielle, en orientant la conception vers une solution à la fois robuste, évolutive et adaptée aux spécificités du marché algérien, notamment en facilitant l'intégration future de modules intelligents comme l'algorithme d'affectation des commandes.

Les diagrammes suivants illustrent de manière synthétique les parcours et interactions de :

• Utilisateur professionnel : inscription, ajout et gestion des produits, intégration manuelle des commandes, suivi des livraisons en temps réel et communication avec les livreurs.

• Livreurs : inscription, consultation des commandes entrantes, acceptation, réception d'un itinéraire optimisé et mise à jour du statut de livraison.

• Administrateur : supervision des utilisateurs, gestion des commandes en cours, suivi des performances des livreurs et pilotage global de la plateforme.

2.5.1 Client

2.5.1.1 diagramme de cas d'utilisation

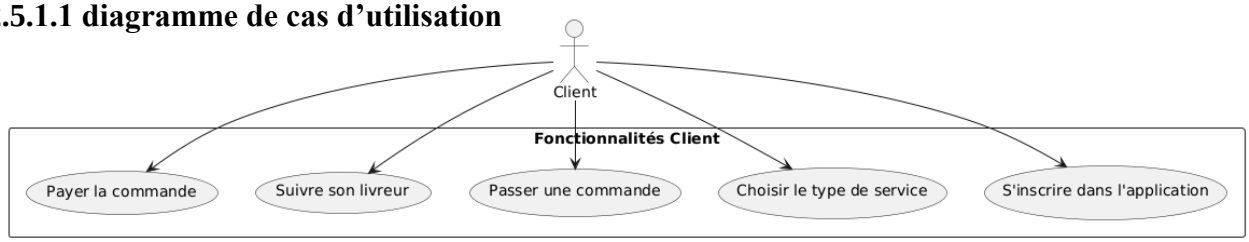


Figure 2.4 Diagramme de cas d'utilisation spécifique au Client.

L'application permet de commander, choisir un service, payer et suivre sa livraison en temps réel, le tout via une interface simple et intuitive. Toutes les étapes clés sont intégrées pour une expérience utilisateur fluide.

2.5.1.2 diagramme de classes

Ce modèle présente la structure de base de données avec les tables principales (Client, Commande, Livreur, Vendeur, Produit, Paiement) et leurs relations. Chaque entité contient ses attributs clés (ID, noms, statuts) et ses méthodes métiers essentielles (passerCommande(), effectuerPaiement(), etc.). Le schéma couvre l'ensemble du cycle, de la commande à la livraison, avec un système de suivi via les statuts

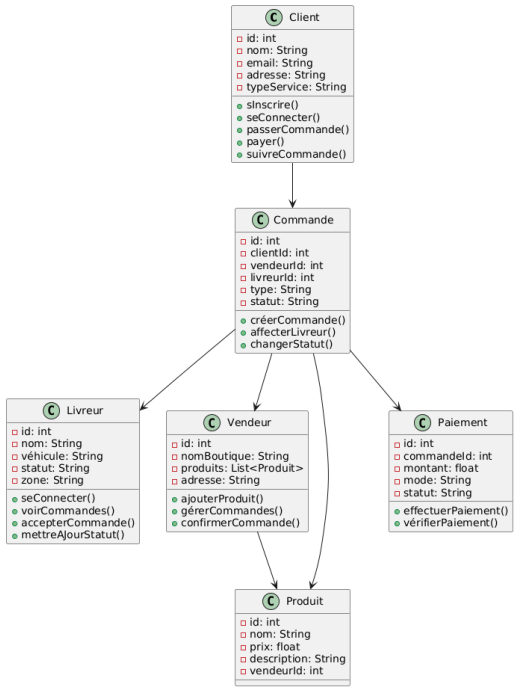


Figure 2.5 Diagramme de classes pour le Client

2.5.1.3 diagramme de séquences

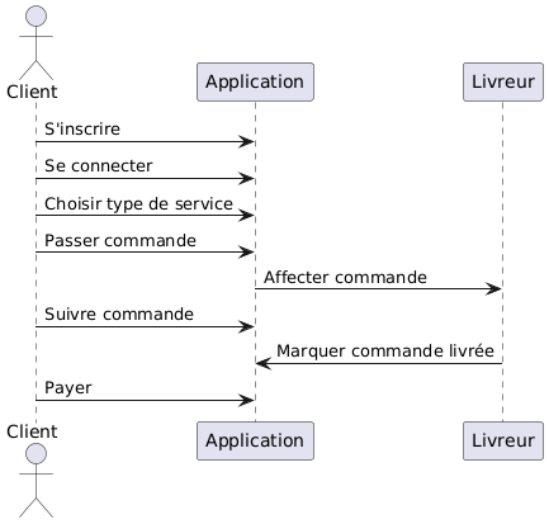


Figure 2.6 Diagramme de séquences pour le Client

L'application connecte clients et livreurs via un flux automatisé : inscription → commande → paiement → affectation intelligente → suivi temps réel → confirmation de livraison. Chaque interaction est optimisée pour rapidité et transparence.

2.5.2 Livreur

2.5.2.1 diagramme de cas d'utilisation

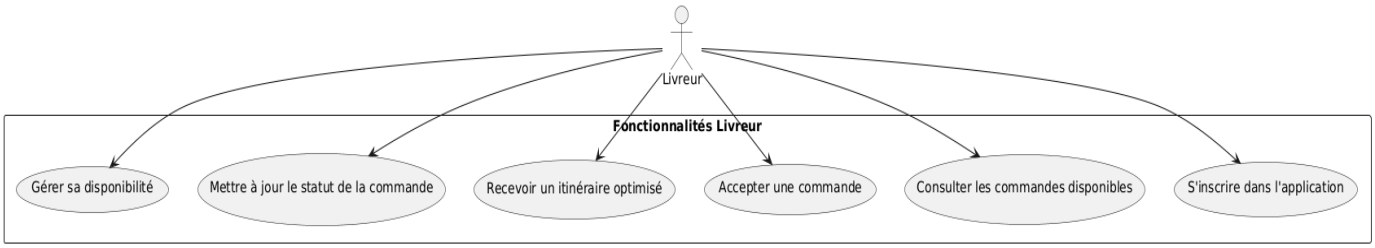


FIGURE 2.7 Diagramme de cas d'utilisation spécifique au Livreur

L'application permet aux livreurs de gérer leur disponibilité, recevoir des commandes avec itinéraires optimisés, et mettre à jour le statut des livraisons en temps réel. Ils peuvent également consulter les missions disponibles et s'inscrire facilement sur la plateforme.

2.5.2.2 diagramme de classes

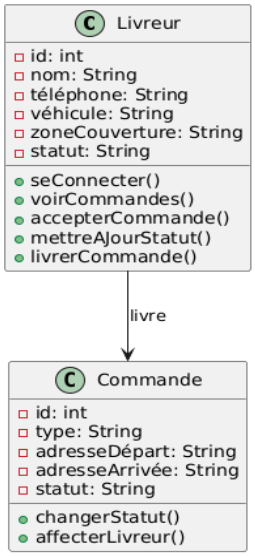


Figure 2.8 Diagramme de classes pour le Livreur

Le livreur peut se connecter, consulter et accepter des commandes, puis mettre à jour le statut de la livraison jusqu'à sa finalisation. Chaque commande contient les adresses de départ et d'arrivée, son type et son statut, permettant une gestion claire et une affectation optimisée.

2.5.2.3 diagramme de séquences

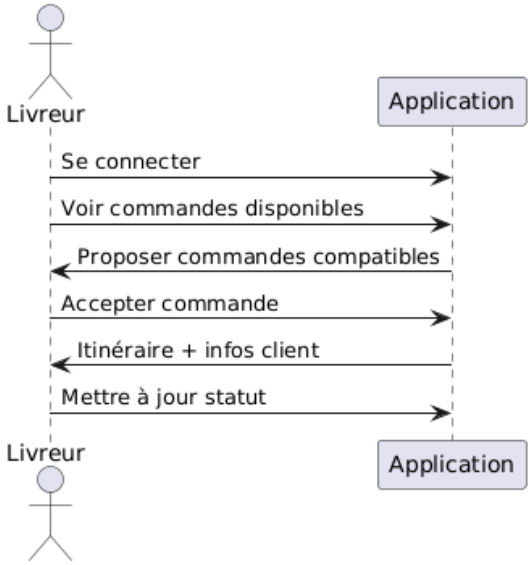


Figure 2.9 Diagramme de séquences pour le Livreur

L'application propose au livreur des commandes compatibles avec sa zone et son véhicule. Après

acceptation, il reçoit un itinéraire optimisé et met à jour l'état de la livraison en temps réel.

2.5.3 Administrateur

2.5.3.1 Diagramme de cas d'utilisation

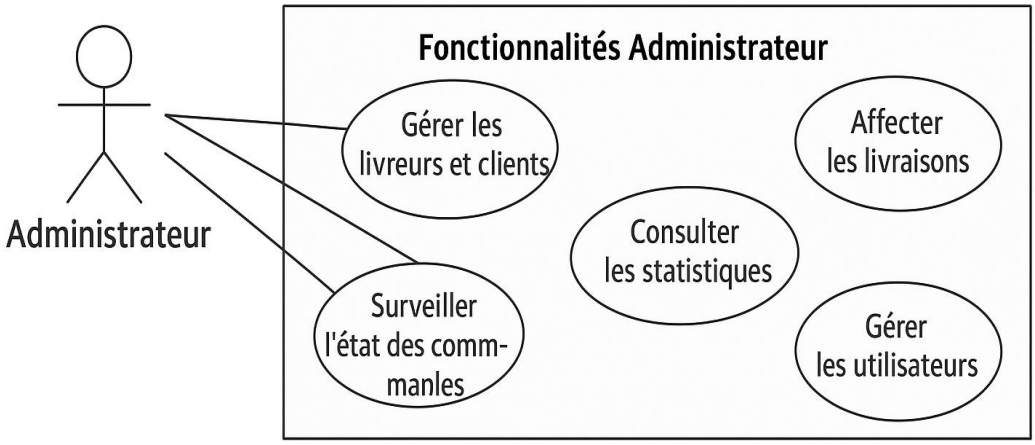


FIGURE 2.10 Diagramme de cas d'utilisation spécifique au l'administrateur

Ce diagramme illustre les principales fonctionnalités attribuées à l'administrateur au sein de la plateforme. Il met en évidence la capacité de ce dernier à gérer les livreurs et les clients, surveiller l'état des commandes, affecter les livraisons, consulter les statistiques globales et administrer les utilisateurs. Ce rôle central permet d'assurer le bon fonctionnement et le suivi opérationnel de l'ensemble du système.

2.5.3.2 Diagramme des classes

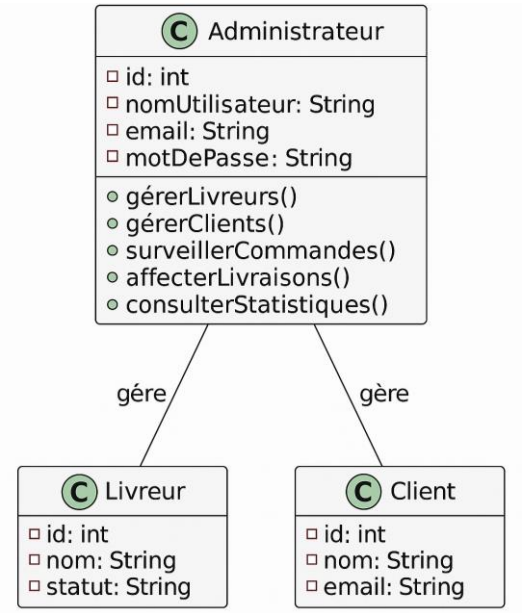


Figure 2.11 Diagramme de classes pour l'administrateur.

Ce diagramme montre les relations entre l'administrateur et les différentes entités qu'il supervise au sein de la plateforme. L'administrateur peut consulter et gérer les comptes des livreurs et des clients, suivre l'état de chaque commande, accéder à des statistiques globales (nombre de commandes, taux de réussite, délais moyens), et intervenir manuellement en cas de besoin. Chaque Commande est liée à un Livreur et à un Client, et possède un statut, une date, et des informations de suivi. L'Administrateur centralise ces données, ce qui lui permet de maintenir le bon fonctionnement du système et de garantir la qualité du service.

2.5.3.3 Diagramme des séquences

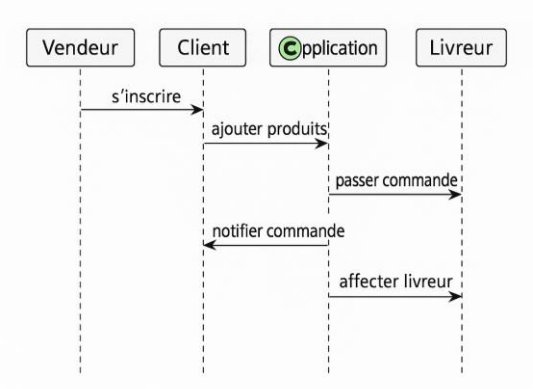


Figure 2.12 Diagramme de séquences pour l'administrateur

Ce diagramme de séquence illustre le rôle central de l'administrateur dans le processus de gestion des commandes. Lorsqu'une commande est effectuée, l'administrateur reçoit une notification lui permettant d'intervenir en cas d'anomalie, de réassigner un livreur, ou de mettre à jour manuellement le statut de la commande. Il peut également surveiller en temps réel l'évolution de toutes les livraisons et consulter les données liées aux performances des livreurs.

2.6 Choix de l'architecture

L'architecture choisie pour le développement de notre plateforme de livraison en Algérie repose sur une combinaison de modèle client-serveur, du paradigme MVC (Modèle-Vue-Contrôleur). Ce choix s'explique par les raisons suivantes :

Notre application repose sur une architecture client-serveur classique, dans laquelle le client, qu'il s'agisse d'une interface mobile ou web, envoie des requêtes à travers une interface utilisateur intuitive. Ces requêtes sont ensuite traitées par un serveur central qui regroupe l'ensemble de la logique métier, interagit avec la base de données, et coordonne la communication avec les services tiers. Ce modèle permet de centraliser les traitements, de mieux gérer les utilisateurs, et de sécuriser efficacement les échanges de données. En parallèle, l'architecture interne de l'application adopte le paradigme MVC (Modèle – Vue – Contrôleur), choisi pour structurer proprement le code, aussi bien côté serveur que côté client. Dans ce modèle, la couche Modèle est responsable de la gestion des données et de la logique métier, notamment pour les commandes, les livreurs, les clients ou encore les paiements. La couche Vue correspond à l'interface utilisateur, accessible depuis une application mobile ou un navigateur web. Enfin, la couche Contrôleur fait l'intermédiaire : elle traite les requêtes issues du client, applique la logique appropriée et renvoie les réponses correspondantes. Ce découpage clair permet une meilleure séparation des responsabilités, favorise la maintenance du code et assure la scalabilité de la solution à long terme.

2.6.1 Aspects fonctionnels de l'application

2.6.1.1 Présentation des fonctionnalités principales

L'application mobile de livraison développée dans le cadre de ce projet a pour but de faciliter la gestion des livraisons urbaines en proposant une interface intuitive, fonctionnelle et intégrée. Plusieurs fonctionnalités majeures ont été prévues :

- Prise de commande : Le client saisit directement les informations nécessaires à la commande, notamment l'adresse de départ , l'adresse d'arrivée et une description détaillée de la commande via une zone de texte. La commande est ensuite affichée au livreur, qui la valide manuellement.
- Affectation intelligente des livreurs : Un système d'affectation basé sur des critères de proximité et de disponibilité est en cours de conception. Il permettra, grâce à des

algorithmes optimisés, d'attribuer les commandes de manière dynamique et efficace. Ce mécanisme d'optimisation fera l'objet d'un développement ultérieur.

- Géolocalisation et navigation : Grâce aux API Google Maps (intégrées au chapitre 2), l'application permet de localiser les livreurs et clients en temps réel, de générer des itinéraires optimisés, et de suivre la position du livreur durant la livraison.
- Suivi en temps réel : Le client peut visualiser en direct la position du livreur, depuis la validation de la commande jusqu'à la livraison.
- Multi-livraison : L'application permettra au livreur de gérer plusieurs livraisons au sein d'un même trajet, en s'appuyant sur une logique d'optimisation des points de passage (expliquée plus tard dans le mémoire).

2.6.1.2 Comportement attendu de l'application selon les utilisateurs

L'interface client a été conçue pour offrir une expérience simple et intuitive. Elle intègre notamment la géolocalisation automatique de la position actuelle de l'utilisateur via les services de localisation de son téléphone, tout en lui laissant la possibilité de sélectionner manuellement une adresse de livraison différente. Une zone de saisie textuelle libre permet de détailler la commande, tandis qu'un bouton dédié facilite la validation de la demande. Une carte interactive affiche en temps réel la position du livreur en charge de la commande, et l'utilisateur peut également enregistrer une ou plusieurs adresses favorites pour simplifier les commandes récurrentes. À ce stade de développement, certaines fonctionnalités ne sont pas encore intégrées, telles que l'estimation automatique du prix ou du délai de livraison, ainsi qu'un système de notation ou de retour d'expérience sur le service ou les livreurs.

Du côté du livreur, l'interface a été pensée pour maximiser l'efficacité opérationnelle. Elle présente une liste des commandes en attente accompagnées des informations essentielles, une carte affichant en temps réel l'itinéraire optimisé depuis le point de retrait (vendeur) jusqu'au point de livraison (client), ainsi qu'un bouton de confirmation de livraison à activer une fois la commande remise. Le livreur peut également signaler toute anomalie rencontrée au cours de sa tournée (colis non prêt, client absent, changement d'itinéraire, etc.), et dispose d'un module de gestion des tournées multiples permettant d'optimiser les trajets impliquant plusieurs commandes grâce à l'algorithme d'affectation.

2.7 Aspects techniques complémentaires

2.7.1 Authentification des utilisateurs

L'application prévoit un système d'authentification permettant de sécuriser l'accès selon les rôles des utilisateurs (clients, livreurs, administrateurs). Ce système permet également de lier chaque commande ou trajet à un utilisateur identifiable.
L'authentification peut être conçue à l'aide :

L'authentification des utilisateurs peut être conçue de plusieurs manières selon les besoins en

termes de sécurité et de simplicité d'usage. Le mode le plus courant repose sur une connexion par adresse e-mail et mot de passe, offrant un niveau de sécurité standard tout en étant facile à implémenter. Une alternative de plus en plus utilisée consiste à permettre la connexion via un numéro de téléphone accompagné d'un code de vérification à usage unique (OTP), généralement envoyé par SMS. Ce mode d'accès est particulièrement adapté aux utilisateurs mobiles et simplifie le processus d'identification. Enfin, il est également envisageable d'intégrer une authentification sociale, telle que Google Sign-In, afin de faciliter la connexion pour les utilisateurs disposant déjà d'un compte Google, tout en renforçant la fiabilité du processus grâce aux mécanismes de sécurité fournis par ces services tiers.
Chaque utilisateur est ensuite associé à un rôle spécifique :

- Client : peut créer une commande, suivre sa livraison, gérer ses adresses favorites.
- Livreur : reçoit des commandes, visualise ses trajets, confirme les livraisons.
- Administrateur : supervise l'ensemble du système, gère les utilisateurs et les anomalies.

L'authentification sera gérée via des services sécurisés et intégrés dans l'application (Firebase Authentication, ou toute autre solution adaptée), avec gestion des sessions actives, récupération de mot de passe, et vérification des identifiants.[19]

2.7.2 Base de données

Pour le stockage, la gestion et la synchronisation des données de l'application, nous avons opté pour Firebase, la plateforme Backend-as-a-Service (BaaS) proposée par Google. Cette solution a été retenue pour sa simplicité d'intégration avec Flutter, ses performances en temps réel, et sa souplesse dans les architectures mobiles modernes.
Firebase propose une suite complète de services, mais dans notre projet, nous utilisons principalement les modules suivants :

2.7.2.1. Cloud Firestore

Il s'agit de la base de données principale de l'application. Firestore est une base de données NoSQL orientée documents, idéale pour les applications mobiles nécessitant des mises à jour fréquentes et instantanées. Elle permet de stocker les données relatives aux utilisateurs (clients, livreurs, administrateurs), aux commandes, à la géolocalisation en temps réel, ainsi qu'à l'historique des livraisons.
Avantages de Firestore

- Synchronisation en temps réel entre tous les clients,
- Évolutivité automatique avec peu de configuration,
- Requêtes complexes et indexation performante,
- Accès hors-ligne et mise en cache automatique.

2.7.2.2. Firebase Authentication

La gestion des utilisateurs passe par le module Firebase Authentication. Celui-ci permet une authentification sécurisée via plusieurs méthodes :

- Email / mot de passe,
- Numéro de téléphone avec envoi de code OTP (One-Time Password),
- Authentification sociale (Google Sign-In) – option envisagée pour de futures versions.

2.7.2.3. Cloud Storage

Ce module est utilisé pour le stockage des fichiers volumineux, tels que les éventuelles photos de profil des utilisateurs ou des captures de commandes.

2.7.2.4. Firebase Cloud Messaging (FCM)

Les notifications push sont gérées par FCM, pour informer en temps réel les clients et les livreurs des changements d'état de commande (commande acceptée, livreur en route, colis livré, etc.).

2.7.2.5. Règles de sécurité

La base de données est protégée par des règles de sécurité Firebase, basées sur l'authentification et le rôle de l'utilisateur (client, livreur, admin). Seuls les utilisateurs autorisés peuvent accéder, modifier ou supprimer certaines données, garantissant ainsi la confidentialité et l'intégrité des informations.[20], [22]

Conclusion

Ce chapitre constitue le cœur conceptuel et technique du mémoire. Il détaille d'abord le fonctionnement global de l'application mobile, avec une description du front-end et de ses principales interfaces, notamment celles du client et du livreur. Ces interfaces permettent respectivement la passation de commande et la gestion des livraisons en temps réel.

La section suivante est consacrée au système d'affectation intelligente des livreurs, un élément central du projet. Elle commence par une présentation générale du concept, puis introduit la méthode de scoring pondéré utilisée pour optimiser l'attribution des commandes. Chaque critère décisionnel (proximité géographique, capacité de charge, temps estimé de livraison, historique de performance, équilibrage de la charge) est défini et justifié. La pondération a été établie selon la méthode du jugement d'experts, en s'appuyant sur des retours d'acteurs du secteur tels que Fast Delivery, Yassir Food ou encore d'anciens restaurateurs.

Une sous-partie traite également de la politique tarifaire de la plateforme, reposant sur un

tarif fixe de prise en charge et un coût variable au kilomètre, à l'inverse des concurrents tels que Jumia ou Fast Delivery.

Le chapitre aborde ensuite la conception UML de l'application, avec la modélisation des cas d'utilisation, des classes et des séquences, permettant de formaliser les interactions entre les utilisateurs (client, livreur, vendeur) et le système.

Enfin, les choix d'architecture sont expliqués, notamment l'adoption du modèle MVC et l'organisation des composants fonctionnels. Le chapitre se conclut sur des aspects techniques importants comme l'authentification des utilisateurs et la gestion des données via Firebase, garantissant ainsi sécurité, évolutivité et accessibilité.

Chapitre 3 l’implémentation de la solution

Introduction

Dans ce chapitre, nous présentons l'ensemble du processus technique lié au développement de notre solution, depuis le choix des technologies jusqu'à l'intégration des services tiers.

L'objectif est d'expliquer les raisons ayant motivé l'adoption de certains outils et environnements de développement, ainsi que de décrire les différentes étapes de conception et de réalisation de l'application.

Nous débuterons par une présentation des technologies principales utilisées, à savoir Flutter et Dart, qui constituent la base du développement de l'interface utilisateur (front-end). Ensuite, nous détaillerons le fonctionnement général de l'application mobile et les principales fonctionnalités implémentées dans l'interface.

La deuxième partie du chapitre sera consacrée à l'intégration des API de services Google, éléments essentiels pour la gestion de la géolocalisation, la cartographie, la recherche de lieux et l'optimisation des itinéraires.

Nous expliquerons comment ces services ont été configurés et intégrés dans notre solution, en abordant successivement les principales API mobilisées, telles que Google Maps SDK, Directions API, Geocoding, Places, ou encore Roads et Geolocation.

Cette implémentation permet de garantir une expérience utilisateur fluide et intelligente, tout en assurant une interaction efficace avec les données géographiques et les services de localisation.

3.1 Choix technologiques et environnement de développement

3.1.1 Présentation de Flutter

Flutter est un framework open source développé par Google, permettant de créer des applications mobiles, web et desktop multiplateformes à partir d'une seule base de code. Flutter est apprécié pour sa rapidité de développement, ses performances natives et sa capacité à fournir des interfaces utilisateur modernes et fluides.
Flutter utilise le moteur graphique Skia qui garantit une expérience cohérente sur Android, iOS, Web, Windows, macOS et Linux. [1], [2]

3.1.2 Présentation de Dart

Dart est le langage de programmation utilisé par Flutter. Il s'agit d'un langage orienté objet, à la fois simple, moderne et spécifiquement optimisé pour le développement d'interfaces utilisateur. Il se distingue par une syntaxe claire, un système de typage puissant, ainsi qu'un double système de compilation. D'une part, le compilateur Ahead-of-Time (AOT) permet de générer des performances optimales en production ; d'autre part, le compilateur Just-in-Time (JIT) facilite un

rechargement à chaud du code (hot reload), très utile durant le développement pour visualiser instantanément les modifications. Le choix de Flutter et Dart dans le cadre de ce projet s'explique par plusieurs avantages concrets. D'abord, Flutter permet un développement multiplateforme, avec un seul code source pour Android et iOS, réduisant significativement les coûts et les délais de production. Ensuite, il offre une grande souplesse pour concevoir des interfaces riches, dynamiques et réactives, particulièrement adaptées aux exigences d'une application de livraison en temps réel. Sur le plan des performances, Flutter, combiné à Dart, assure une exécution fluide grâce à la compilation AOT. Par ailleurs, la fonctionnalité de hot reload accélère grandement les phases de test et de correction. Enfin, Flutter bénéficie d'une excellente intégration avec les services Google Maps grâce à des packages dédiés, tels que google_maps_flutter, ce qui constitue un atout essentiel pour ce projet centré sur la géolocalisation, l'affichage de cartes et le calcul d'itinéraires. [3] , [4] ,

Outils et architecture

- Flutter SDK et Dart SDK pour le développement,
- IDE comme Visual Studio Code ou Android Studio avec plugins Flutter/Dart,
- Packages complémentaires pour la gestion de l'état, les requêtes réseau, et l'affichage de cartes**.**

3.2 description

3.2.1 Présentation de l'application et fonctionnement du front-end

L'application mobile de livraison a été développée en Flutter, avec une logique client-serveur optimisée. Elle se compose de deux interfaces principales : l'interface client et l'interface livreur, chacune adaptée aux rôles et aux fonctionnalités spécifiques des utilisateurs.

3.2.1.1Démarrage de l'application

Lors de l'ouverture, l'utilisateur est invité à choisir son rôle : Client , Livreur ou Administrateur. Selon ce choix, il est redirigé vers l'interface correspondante, où les composants front-end sont dynamiquement chargés comme indiqué ci-dessous dans les figures 3.1 3.2 .

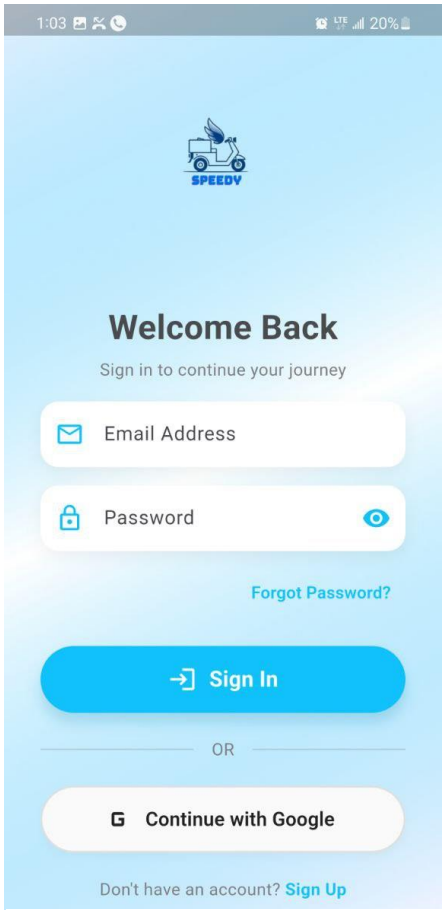


Figure 3.1 Interface de démarrage de l'application

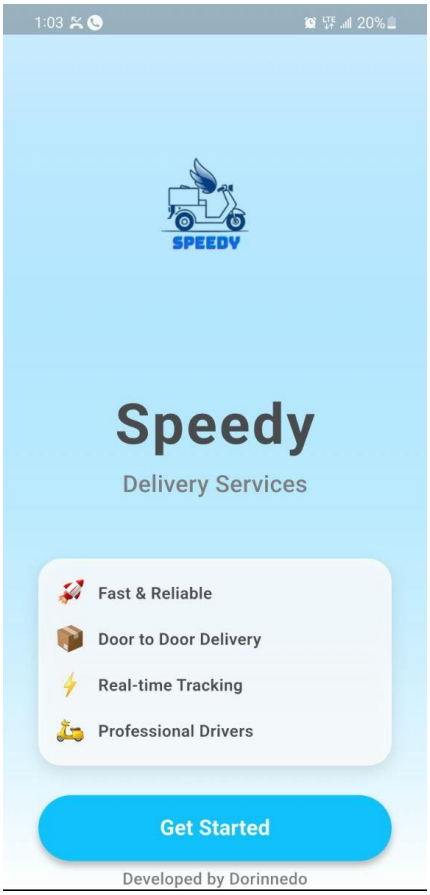


Figure 3.2 Interface de démarrage de l'application

3.2.1.2Interface Client :

L’écran principal affiche une Google Map avec :

- Le point de départ (pick-up) ;
- Le point de livraison ;
- Une zone de commentaire où le client peut décrire sa commande.

Après validation de la commande, l’interface affiche :

- Une liste de livreurs disponibles (avec leurs positions actuelles sur la carte) ;
- Le temps estimé d’arrivée ;
- Le kilométrage total ;
- La note moyenne (rating) de chaque livreur ;
- Le prix estimatif de la livraison.

Cette interaction est rendue possible grâce aux services Google Maps intégrés (Maps SDK, Directions API, etc.) ainsi qu’une logique d'affectation intelligente.

Les figures 3.3, 3.4 et 3.5 ci-dessous présentent les détails affichés à l'écran sur smartphone.

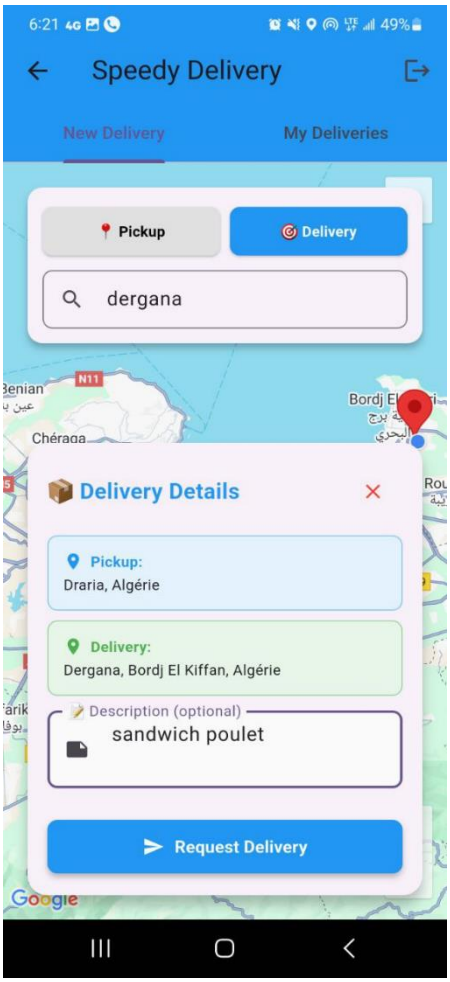


Figure 3.3 : Interface client affichant les pick up et destinations

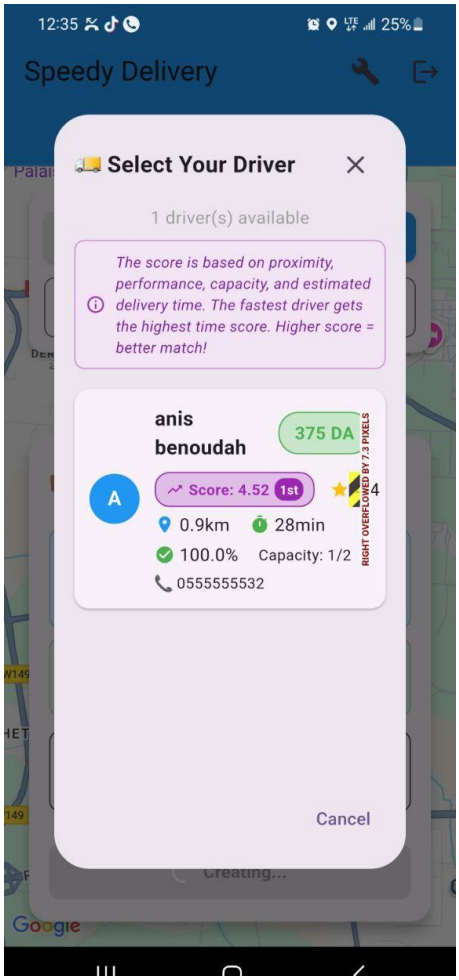


Figure 3.5 Interface client avec détails des livreurs

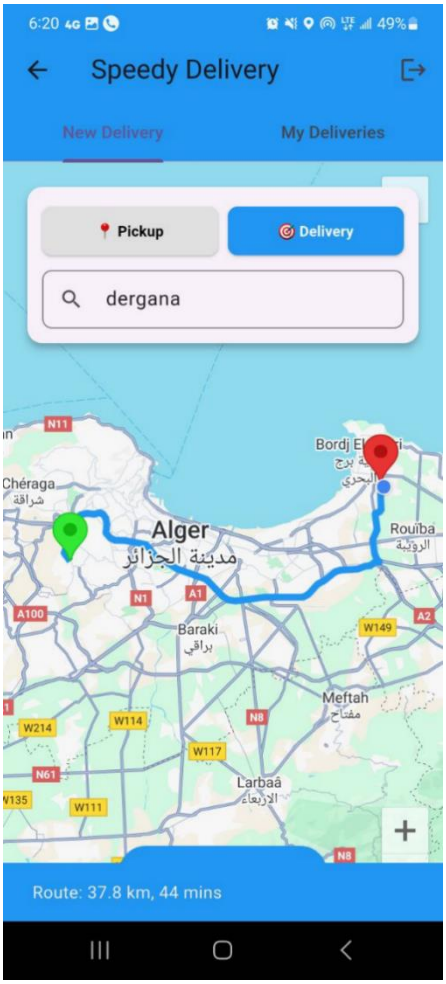


Figure 3.4 Interface client avec une maps

3.2.1.3 Interface Livreur

L’interface du livreur est orientée opération et performance. Elle comprend :

Une carte interactive avec l'affichage des commandes à proximité.

Un bouton ON/OFF permettant au livreur d’activer ou de désactiver sa disponibilité.

Une zone de paramètres personnels affichant :

- Le taux de réussite (Success Rate) ;
- Le taux de ponctualité (On Time Rate) ;
- Le rating moyen ;
- La capacité actuelle de livraison (nombre de commandes qu’il peut encore accepter).

Les figures 3.6 et 3.7 affichent respectivement une commande entrante et le tableau de bord du livreur.

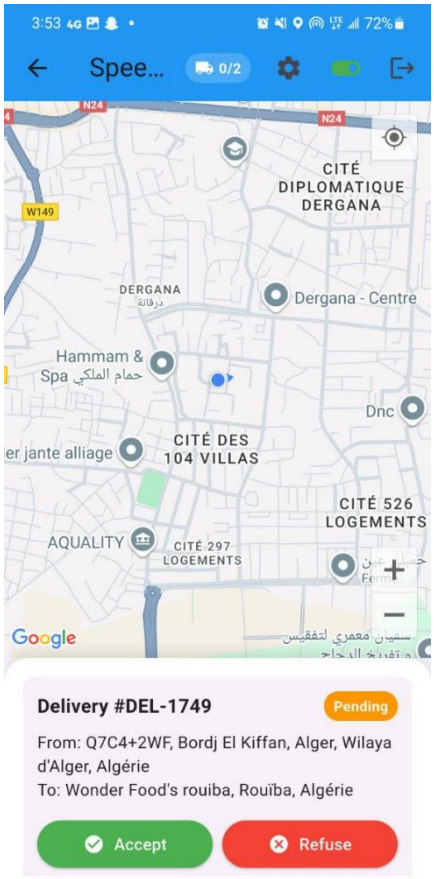


Figure 3.6 Interface livreur avec liste des commandes

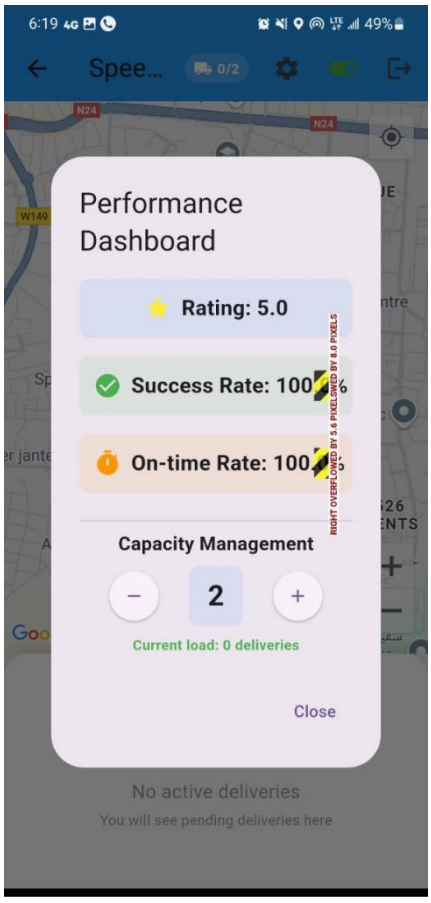


Figure 3.7 *Interface livreur avec paramètres personnels*

3.2.1.4 Interface administrateur

L'interface administrateur est orientée supervision et contrôle global de la plateforme. Elle comprend:

- Un tableau de bord (Dashboard) affichant en temps réel les indicateurs clés de performance :
  • Le nombre total de commandes traitées, en cours ou échouées ;
  • Le taux global de réussite des livraisons ;
  • Le nombre de livreurs actifs ;
  • Des statistiques par période (jour, semaine, mois).
- La vue d'ensemble affiche, sous forme textuelle, les indicateurs clés tels que les commandes, les livreurs actifs, les zones sollicitées et les performances.
- Une section de gestion des livreurs intégrant les opérations CRUD (Créer, Lire, Modifier, Supprimer), avec la possibilité de :
  • Ajouter ou désactiver un livreur ;
  • Consulter son taux de réussite, sa ponctualité, ses commandes livrées ;
  • Modifier ses données ou ses zones d'intervention.
- Un module dédié à l'historique des commandes, permettant :
  • D'afficher les détails de chaque commande (client, livreur assigné, statut, horaires, durée) ;
  • De filtrer par statut (livrée, en cours, échouée) ;
  • De réaffecter manuellement une commande si nécessaire.

La figure **3.8** présente les différentes fonctionnalités de l'interface administrateur

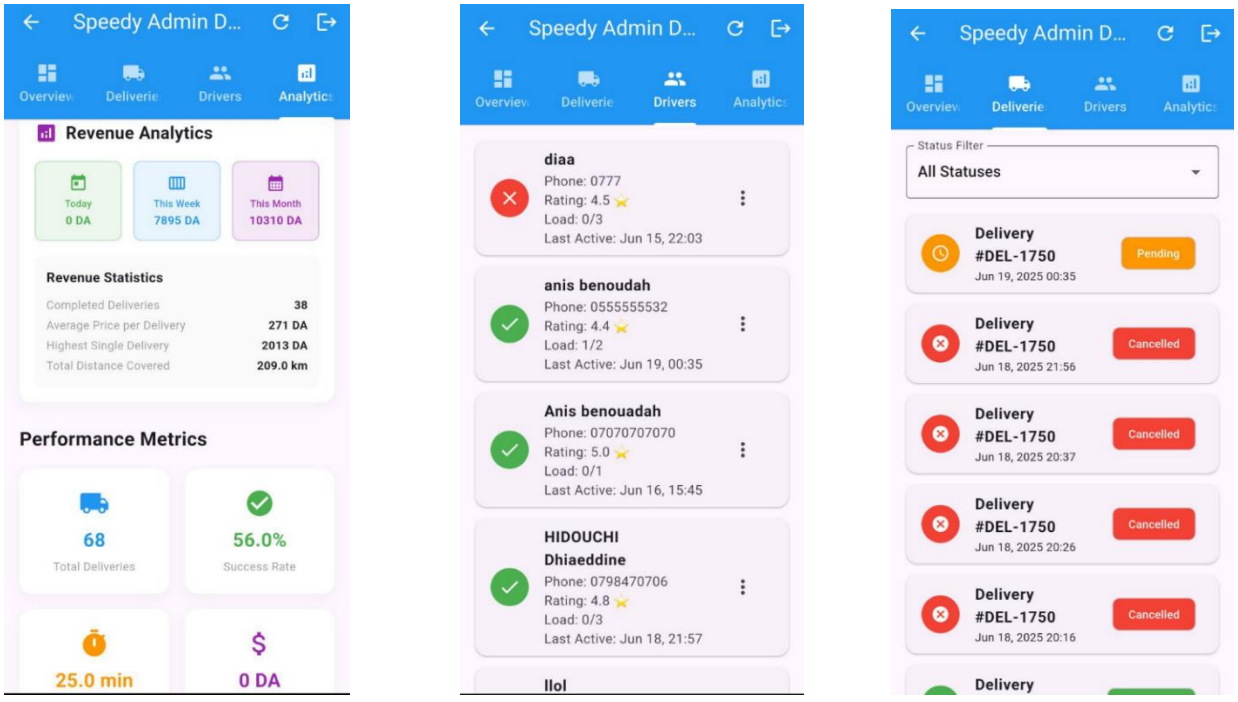


Figure3.8 Interface administrateur

3 .3 Intégration des services Google API

3.3.1 Définition générale des API

Une API (Application Programming Interface) est un ensemble de définitions et de protocoles qui permet à deux applications de communiquer entre elles. Dans le cadre d'une application de livraison, les API permettent notamment de récupérer des données en temps réel (géolocalisation, adresses, itinéraires optimisés, etc.) et d'offrir des services interactifs et performants aux utilisateurs.

3.3.2 Obtention des API Google

Pour exploiter les services offerts par Google, l'utilisation de la plateforme Google Cloud Console a été nécessaire. Cette démarche implique tout d'abord la création d'un projet dédié sur la console afin d'y centraliser les services et ressources nécessaires au développement. Elle exige également la validation de l'identité du développeur, effectuée à travers un compte Google vérifié, garantissant ainsi un niveau minimal de sécurité et de conformité. Une fois le projet créé, les différentes API indispensables au fonctionnement de l'application doivent être activées individuellement. Enfin, une clé API sécurisée est générée ; elle permet d'authentifier chaque requête envoyée aux serveurs de Google et constitue un élément central pour garantir le bon fonctionnement et la protection des services utilisés par l'application. [5] ,[25], [24]

3.4 APIs et SDK Google intégrés

3.4.1 Google Maps SDK for Android

Définition

Le Google Maps SDK for Android est une bibliothèque fournie par Google qui permet d'intégrer des cartes interactives dans les applications Android. Elle offre aux développeurs la possibilité d'ajouter des fonctionnalités de cartographie avancées, telles que l'affichage de cartes, l'ajout de marqueurs, la gestion des interactions utilisateur et la personnalisation de l'apparence des cartes.

Functionalities principals

1. Affichage de cartes interactives : Intégration de cartes Google Maps dans les applications Android, avec prise en charge des gestes tels que le zoom, la rotation et le déplacement.
2. Ajout de marqueurs : Possibilité d'ajouter des marqueurs personnalisés pour indiquer des emplacements spécifiques, tels que des points de livraison ou des entrepôts.
3. Dessiner sur la carte : Utilisation de polylignes, de polygones et de cercles pour représenter des itinéraires, des zones de couverture ou d'autres informations géographiques.

4. Personnalisation de l'apparence : Modification de l'apparence des cartes à l'aide de styles JSON, permettant d'adapter les couleurs, la visibilité des éléments et d'autres aspects visuels.
5. Gestion des événements utilisateur : Réponse aux interactions utilisateur, telles que les clics sur la carte ou sur des marqueurs, pour déclencher des actions spécifiques dans l'application.. [6] , [14]

3.4.2 Google Directions API

Définition

La Google Directions API est un service web fourni par Google qui permet de calculer des itinéraires entre différents points géographiques. Elle prend en charge plusieurs modes de transport, tels que la conduite, la marche, le vélo et les transports en commun. Cette API est essentielle pour les applications de livraison, car elle permet de déterminer les trajets optimaux entre les points de collecte et de livraison.

Fonctionnalités principales

1. Calcul d'itinéraires : Détermination des trajets entre un point d'origine et une destination, avec possibilité d'ajouter des étapes intermédiaires (waypoints).
2. Modes de transport : Support des modes de déplacement suivants : conduite (driving), marche (walking), vélo (bicycling) et transports en commun (transit).
3. Optimisation des itinéraires : Possibilité d'optimiser l'ordre des étapes pour minimiser la distance ou le temps de trajet.
4. Conditions de trafic en temps réel : Prise en compte du trafic actuel pour estimer les temps de trajet plus précisément.
5. Restrictions de trajet : Capacité à éviter certaines routes, telles que les autoroutes, les péages ou les ferries.
6. Retour d'informations détaillées : Fourniture d'instructions étape par étape, de la distance totale, du temps estimé, et de la géométrie de l'itinéraire.[7] ,

3.4.3 Google Geocoding API

Définition générale

La Google Geocoding API est un service proposé par Google qui permet de convertir des adresses physiques en coordonnées géographiques (latitude et longitude) et inversement. Ce processus est essentiel pour intégrer des données de localisation précises dans des applications web ou mobiles. Le géocodage permet, par exemple, de placer des marqueurs sur une carte à partir d'adresses saisies par l'utilisateur, tandis que le géocodage inversé permet d'obtenir une adresse lisible à partir de coordonnées GPS.

Fonctionnalités principales

1. Géocodage (Forward Geocoding)

Le géocodage consiste à convertir une adresse postale en coordonnées géographiques. Par exemple, l'adresse "1600 Amphitheatre Parkway, Mountain View, CA" peut être transformée en latitude 37.4224764 et longitude -122.0842499. Cette fonctionnalité est utile pour afficher des emplacements sur une carte ou pour calculer des itinéraires.

2. Géocodage inversé (Reverse Geocoding)

Le géocodage inversé permet de convertir des coordonnées géographiques en une adresse humaine lisible. Par exemple, les coordonnées 37.4224764, -122.0842499 peuvent être traduites en "1600 Amphitheatre Parkway, Mountain View, CA". Cette fonctionnalité est particulièrement utile pour afficher l'adresse actuelle d'un utilisateur basé sur sa position GPS.

3. Géocodage basé sur l'identifiant de lieu (Place ID)

La Google Geocoding API permet également de récupérer des informations d'adresse à partir d'un identifiant de lieu unique (Place ID). Cela garantit une précision accrue lors de la récupération des données de localisation.[9]

3.4.4 Google Places API

Définition

La Google Places API (New) est un service web proposé par Google qui permet d'accéder à des informations détaillées sur des lieux spécifiques, tels que des établissements commerciaux, des points d'intérêt géographiques ou des adresses précises. Elle offre des fonctionnalités avancées pour rechercher, identifier et obtenir des détails sur des lieux, facilitant ainsi l'intégration de données de localisation dans des applications mobiles ou web.

Fonctionnalité de la Google Places API (New)

La Google Places API (New) permet à une application d'interagir avec une base de données exhaustive de lieux à travers plusieurs services intégrés. Elle facilite l'identification, la suggestion, et l'affichage d'informations détaillées sur des établissements, adresses, commerces ou points d'intérêt géographiques. Cela permet à une application de livraison ou de navigation d'offrir à ses utilisateurs une **expérience fluide, personnalisée et localisée**, en leur proposant par exemple :

- une **autocomplétion intelligente** des adresses saisies ;
- des **détails pratiques** sur les lieux (adresse, téléphone, horaires, avis clients, etc.) ;
- des **photos de qualité** associées aux lieux ;
- une **recherche contextuelle** selon la position de l'utilisateur ou la catégorie souhaitée.

Ces services sont conçus pour s'adapter aussi bien aux interfaces mobiles qu'aux systèmes de traitement backend, avec une précision et une performance compatibles avec des applications logistiques modernes.[8]

3.4.5 Google Roads API

Définition générale

La Google Roads API est un service web proposé par Google permettant de connecter les positions GPS à la route la plus proche et de fournir des données routières précises. Cette API est conçue pour améliorer la qualité des données GPS souvent bruitées ou imprécises en les "corrigeant" pour qu'elles correspondent exactement à la route empruntée par un véhicule.
Elle est particulièrement utile pour les applications liées à la mobilité, à la navigation et à la gestion des flottes, permettant de localiser avec précision un véhicule ou un trajet sur une carte[17]

Fonctionnalités principales

1. Alignement sur la route (Snap to Roads)
   Cette fonctionnalité permet de corriger les points GPS collectés, souvent imprécis ou dispersés, en les ajustant automatiquement à la route la plus proche. Elle reconstitue avec précision le trajet réel emprunté par un véhicule, même lorsque le signal GPS est partiellement défaillant. Cela permet d'obtenir un tracé plus réaliste et fluide sur la carte.
2. Routes les plus proches (Nearest Roads)
   Cette fonctionnalité identifie la route la plus proche pour chaque point GPS, même s'il ne fait pas partie d'un itinéraire continu. Elle est particulièrement utile pour localiser précisément un véhicule ou un utilisateur en temps réel, même si un seul point de géolocalisation est disponible.
3. Limites de vitesse (Speed Limits)
   La Roads API fournit les vitesses maximales autorisées sur les segments de route parcourus. Cette donnée est précieuse pour les applications de gestion de flotte, car elle permet de surveiller les excès de vitesse, d'améliorer la sécurité routière et de sensibiliser les conducteurs au respect du code de la route.
4. Interpolation de trajets
   Lorsque les points GPS sont espacés ou manquants (par exemple en cas de perte temporaire de signal), la Roads API peut interpoler les positions manquantes entre deux points. Cela permet de générer un itinéraire complet et réaliste, même si certaines données GPS ont été perdues en cours de route.
5. Traitement par lots (Support multi-points)
   L'API permet de traiter jusqu'à 100 points GPS dans une même requête (selon les conditions d'utilisation). Cette capacité permet d'optimiser les performances lors de la reconstitution de trajets longs ou complexes, sans multiplier les appels API.

3.4.6 Google Routes API

Définition générale

La Google Routes API est un service proposé par Google qui permet de calculer des itinéraires précis entre plusieurs points, tout en prenant en compte des éléments contextuels comme le trafic en temps réel, les préférences d'itinéraire, ou encore le mode de transport. Contrairement à la Directions API, la Routes API propose une expérience enrichie, plus flexible et plus performante pour des cas d'usage complexes comme la logistique, les tournées de livraison ou les services multimodaux.

Fonctionnalités principales

1. Calcul d'itinéraires dynamiques
   Elle génère l'itinéraire optimal entre un point de départ et une ou plusieurs destinations, en tenant compte de la distance, du temps estimé et des conditions actuelles du trafic.
2. Support multi-destinations
   Elle permet d'ajouter plusieurs waypoints (étapes intermédiaires), utiles pour planifier des tournées de livraison ou des trajets complexes.
3. Optimisation automatique des arrêts
   Grâce à l'option optimizeWaypoints, l'API peut automatiquement réorganiser l'ordre des arrêts pour réduire le temps ou la distance totale du trajet.
4. Prise en compte du trafic en temps réel
   En intégrant des données de trafic actualisées, l'API peut adapter l'estimation de durée en fonction des embouteillages, incidents ou événements routiers.
5. Support multimodal
   L'API prend en charge différents modes de transport (voiture, marche, vélo, transport en commun) et peut combiner ces modes selon le besoin de l'utilisateur.
6. Retour détaillé d'informations
   Chaque réponse contient des instructions de navigation, des polylignes encodées (pour le tracé de la route), des informations sur les distances, durées, conditions de trafic, etc.[18],

3.4.7 Google Geolocation API

Définition générale

La Google Geolocation API permet de déterminer la position géographique approximative d'un appareil même en l'absence de GPS, en utilisant des informations telles que :

- Les réseaux Wi-Fi à proximité,
- Les antennes relais (tours cellulaires),
- Les adresses IP,
- Et parfois le GPS s'il est disponible.

Cette API est particulièrement utile pour des appareils mobiles ou embarqués qui ne disposent pas de GPS ou pour améliorer la rapidité d'obtention de la position initiale. [23]

Fonctionnalités principales

1. Détection rapide de la position
   Elle permet d'obtenir rapidement une estimation de la position, sans attendre la localisation GPS complète (qui peut prendre du temps).
2. Utilisation des réseaux environnants
   La position est déterminée à partir :
   - des identifiants des tours cellulaires,
   - des signaux Wi-Fi disponibles,
   - ou des deux combinés.
3. Retour d'une position approximative
   L'API retourne :
   - Latitude et longitude estimées,
   - Précision (accuracy) en mètres..

Conclusion

Ce chapitre a permis de présenter en détail les aspects techniques et technologiques ayant permis la concrétisation de l'application mobile. Le choix de Flutter et Dart s'est avéré stratégique, tant pour leur capacité à offrir un développement multiplateforme efficace que pour la rapidité d'exécution, la richesse des interfaces et la simplicité de maintenance du code.

L'intégration des services Google Maps Platform s'est révélée indispensable pour doter l'application de fonctionnalités avancées en matière de géolocalisation, d'optimisation d'itinéraires, d'affichage de cartes interactives et de traitement précis des données spatiales. Chacune des API intégrées (Maps SDK, Directions, Geocoding, Places, Roads, Routes et Geolocation) a été sélectionnée avec soin selon sa valeur ajoutée dans le contexte de la livraison urbaine.

L'exploitation de ces outils a permis de construire une solution robuste, intelligente et parfaitement adaptée aux enjeux de mobilité et de logistique en milieu urbain. En parallèle, les bonnes pratiques mises en œuvre pour la sécurisation des clés API, la gestion des quotas, et la maîtrise des coûts assurent la pérennité et la viabilité technique du projet à long terme.

Ainsi, l'implémentation technologique de l'application constitue une base solide et évolutive, prête à être déployée, enrichie et maintenue dans un environnement professionnel réel.

Chapitre 4 Intégration d’un agent DQN dans le système d’affectation des commandes

1. Introduction

L'attribution intelligente de commandes dans une application de livraison constitue un enjeu majeur en logistique urbaine. Une première approche consistait à utiliser un algorithme de scoring pondéré, combinant plusieurs critères pour choisir le meilleur livreur.

Cette étude présente l'évolution vers une approche plus intelligente et adaptative : l'utilisation du Deep Q-Learning (DQN), une méthode d'apprentissage par renforcement profond, afin de permettre à un agent logiciel d'apprendre dynamiquement à prendre les meilleures décisions. [27]

2. Contexte et limites de l'approche scoring pondéré

Le modèle initial reposait sur des critères pondérés :

- Proximité géographique
- Capacité de charge
- Historique de performance
- Temps estimé de livraison

Chaque livreur était noté pour chaque critère selon une échelle (0 à 5), puis un score global était calculé selon des poids prédéfinis. Bien que performant, ce modèle est :

- statique (les poids sont fixes),
- non adaptatif (pas d'apprentissage en fonction des erreurs),
- sans exploration automatique de nouvelles stratégies.

Introduction a l’intillengence artificiel et le Reinforcement Learning

Dans un monde où la technologie évolue à une vitesse vertigineuse, l’intelligence artificielle (IA) se trouve au cœur des innovations les plus disruptives. Une branche particulièrement fascinante de l’IA, le Reinforcement Learning (RL), ou apprentissage par renforcement, se distingue par sa capacité à permettre aux machines d’apprendre de leurs interactions avec l’environnement, en maximisant une notion de récompense. Cette introduction plonge dans l’univers du RL, explorant ses fondements, son importance croissante et comment il se différencie des autres méthodes d’apprentissage automatique. [29]

Le **Reinforcement Learning (RL)**

C’est une méthode d'apprentissage automatique inspirée de la psychologie comportementale. Il permet à des agents logiciels d'apprendre à accomplir des tâches en essayant différentes stratégies et en évaluant leurs résultats.

Principes Fondamentaux
Le RL repose sur l'interaction entre un **agent** et son **environnement**.

- L'**agent** est l'entité qui apprend et prend des décisions.
- L'**environnement** est le monde avec lequel l'agent interagit.
- Les **récompenses** sont des signaux que l'environnement envoie à l'agent, indiquant la valeur d'une action.
- La **politique** définit la stratégie que l'agent utilise pour choisir ses actions.

Fonctionnement
Le processus d'apprentissage en RL suit un cycle continu :

1. **Observation** : L'agent observe son environnement.
2. **Décision** : Il choisit une action basée sur sa politique actuelle.
3. **Action** : L'agent exécute l'action dans l'environnement.
4. **Récompense** : L'environnement fournit un retour (récompense ou pénalité) sur l'action effectuée.
5. **Apprentissage** : L'agent ajuste sa politique pour améliorer ses futures décisions.

Un aspect crucial du RL est le **dilemme exploration vs. exploitation**. L'agent doit équilibrer l'exploration de nouvelles stratégies pour découvrir de meilleures récompenses potentielles et l'exploitation des connaissances acquises pour maximiser ses récompenses actuelles.

Algorithmes et Modèles en RL :

- **Q-Learning** : Un classique. Cet algorithme permet à l'agent d'apprendre la qualité (Q) de prendre certaines actions dans certains états.
- **Deep Q-Networks (DQN)** : Une révolution. DQN combine Q-learning avec des réseaux de neurones profonds, permettant à l'agent d'apprendre dans des environnements visuellement complexes

  En somme, le Reinforcement Learning est un domaine dynamique où les machines apprennent de manière autonome par l'expérience, ajustant leurs stratégies grâce à un équilibre entre l'exploration de l'inconnu et l'exploitation des connaissances.

3. Introduction à Deep Q-Network (DQN)

Le DQN est une avancée majeure en apprentissage par renforcement qui combine deux idées puissantes :

1. **Q-learning classique** :
    - C'est une méthode pour apprendre une fonction Q(s,a) qui estime la "qualité" (Q) de prendre une action (a) dans un état donné (s)
    - Cette qualité représente la récompense totale future attendue si on prend cette action maintenant puis suit la meilleure politique ensuite
2. **Réseau de neurones profond** :
    - Pour gérer des espaces d'états complexes (comme notre problème de livraison), on utilise un réseau de neurones pour approximer cette fonction Q
    - Le réseau prend l'état en entrée et sort une valeur Q pour chaque action possible

Pourquoi cette combinaison est utile :

- Le Q-learning seul ne peut pas gérer des états complexes avec beaucoup de dimensions
- Le réseau de neurones permet de généraliser à partir d'états similaires
- On peut ainsi traiter des problèmes réels complexes comme l'attribution de livreurs

4. Modélisation du problème comme environnement RL

État (State)

L'état est une représentation numérique de la situation actuelle. Pour chaque livreur par rapport à une commande, on pourrait avoir :

1. **Proximité** : Distance entre la position actuelle du livreur et le lieu de ramassage de la commande
    - Exemple : 2.5 km → normalisé entre 0 et 1
2. **Charge** : Nombre de commandes que le livreur a déjà à livrer
    - Exemple : 3 commandes en cours → pourrait être normalisé par un maximum (disons 5)
3. **Historique** : Performance passée du livreur (temps moyen de livraison, taux de satisfaction)
    - Exemple : 4.7/5 étoiles moyennes → normalisé à 0.94

4.Temps estimé : Temps estimé pour compléter la livraison en incluant les commandes actuelles

- Exemple : 35 minutes → pourrait être normalisé par 120 minutes (2h)

Un état complet pourrait donc être un vecteur comme [0.8, 0.6, 0.94, 0.29] pour un livreur donné.

Action

Dans ce contexte, une action est simplement le choix d'un livreur parmi ceux disponibles :

- Si vous avez 5 livreurs disponibles, vous avez 5 actions possibles
- Le réseau de neurones sortira une valeur Q pour chacun de ces 5 choix

Récompense (Reward)

La récompense est un score numérique qui évalue la qualité de l'attribution, La récompense guide l'apprentissage pour favoriser les attributions optimales.

Politique (Policy)

La politique est la stratégie que l'agent apprend pour décider quelle action prendre dans chaque état. Avec DQN :

1. Initialement, l'agent explore aléatoirement (pour découvrir quelles actions donnent de bonnes récompenses)
2. Progressivement, il apprend à privilégier les actions avec les plus hautes valeurs Q
3. La politique finale est essentiellement : "Dans cet état, choisis le livreur avec la valeur Q la plus élevée"

Comment tout cela fonctionne ensemble ?

1. **Initialisation** : On commence avec un réseau de neurones dont les poids sont aléatoires
2. **Expérience** :
    - Pour chaque commande, on observe l'état (caractéristiques de chaque livreur)
    - On choisit un livreur (soit aléatoirement, soit selon les Q-values actuelles)
    - On observe la récompense (après la livraison)
3. **Apprentissage** :
    - On stocke cette expérience (état, action, récompense, nouvel état) dans une mémoire
    - Régulièrement, on sample des lots de ces expériences pour entraîner le réseau
    - Le réseau apprend à prédire les Q-values qui maximisent les récompenses futures
4. **Amélioration continue** :
    - Au fur et à mesure, le réseau devient meilleur pour prédire quels livreurs seront les meilleurs choix
    - On obtient une politique d'attribution optimale

Cette approche permet de capturer des relations complexes qu'une simple règle heuristique ne pourrait pas exprimer, comme le compromis optimal entre proximité, charge de travail et performance historique.

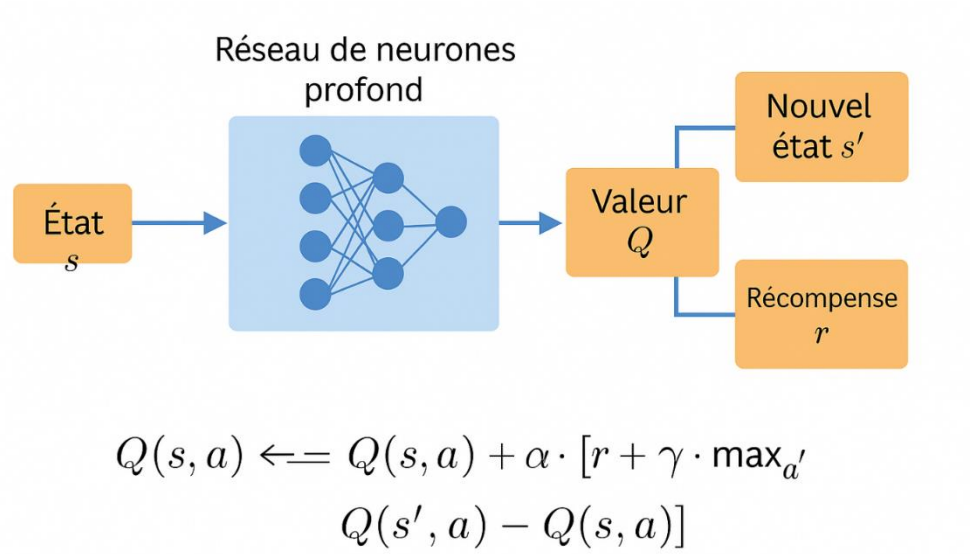


Figure 4.1 Fonctionnement du modèle DQN

Explication de schéma :

État s : C'est la situation actuelle, représentée sous forme de vecteur.
Réseau de neurones profond : Ce réseau reçoit l'état sss en entrée, et en sortie il donne une valeur Q pour chaque action possible (c'est-à-dire chaque livreur).
Chaque Q correspond à l'estimation de la qualité de l'action dans cet état.
Valeur Q(s,a) : C'est la valeur de Q pour une action spécifique (par exemple, attribuer la commande au livreur 2).
Nouvel état s′ et récompense r :

Après avoir choisi un livreur, la commande est effectuée.
L'agent observe alors :

- Le **nouvel état** s′ (ex: nouvelle situation après la commande),
- Une récompense r, par exemple :
  - +1 si la livraison a été rapide et réussie,
  - -1 si elle a échoué ou pris trop de temps.

Mise à jour de la valeur Q :

C'est la formule mathématique qui met à jour la valeur de Q pour que l'agent apprenne :

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot [r + \gamma \cdot \max Q(s',a') - Q(s,a)]$$

- α: taux d'apprentissage
- γ : facteur de réduction (importance du futur)
- r : récompense immédiate
- $\max Q(s',a')$: meilleure valeur Q estimée à partir du nouvel état

Exemple concret (jeu de données simulé)

Livreurs disponibles

Livreur	Distance (km)	Capacité	Performance (%)	Temps estimé (min)
Ali	1.5	10	90	10
Karim	4.3	5	85	18
Samir	2.0	7	95	12

Normalisation des données

On transforme ces valeurs pour que le DQN puisse les utiliser.

Donnes normalisé

Critère	Ali	Karim	Samir
Proximité (1 - d/10)	0.85	0.57	0.80
Capacité (/10)	1.0	0.5	0.7
Performance (/100)	0.90	0.85	0.95
Temps estimé (1 - t/20)	0.50	0.10	0.40

Chaque livreur est donc représenté par un **état** (vecteur) :

- Ali : [0.85, 1.0, 0.90, 0.50]
- Karim : [0.57, 0.5, 0.85, 0.10]
- Samir : [0.80, 0.7, 0.95, 0.40]

4. Valeurs Q estimées par le DQN

La valeur Q est la sortie d'un réseau de neurones entraîné, qui approxime la fonction Q(s, a) :

Livreur	Valeur Q estimée
Ali	0.72
Karim	0.55
Samir	0.81

- s = état (ici les caractéristiques normalisées de des livreurs par rapport la commande)
- a = action (choisir un livreur)

Le réseau de neurones apprend à prédire :
Q(s, a) = récompense attendue si on choisit l'action a dans l'état s et qu'on suit une bonne stratégie ensuite.
Architecture DQN simplifiée

- Un réseau de neurones avec 2 couches cachées de 64 neurones chacune
- Une couche d'entrée (4 neurones, correspondant aux 4 critères)
- Une couche de sortie avec 3 neurones (3 livreurs à choisir)

Chaque sortie représente une valeur Q pour une action (livreur), et le modèle apprend à ajuster ces valeurs à chaque étape selon la récompense reçue.
Résultat final
Le DQN choisit le livreur ayant la plus grande valeur Q **:**
Livreur sélectionné : Samir

Pseudo code :

Afin d'illustrer concrètement le passage d'un système de scoring pondéré vers un modèle d'apprentissage par renforcement basé sur le Deep Q-Network (DQN), nous proposons ci-dessous un pseudocode représentatif du processus.
Ce pseudocode s'articule en trois phases principales :

1. **Préparation des données** issues du système initial de scoring pondéré, servant de base d'entraînement supervisé pour le modèle DQN ;
2. **Entraînement de l'agent DQN**, où l'agent apprend progressivement à prédire les meilleures actions (attributions de livreurs) en maximisant la récompense cumulée ;
3. **Exploitation du modèle en production**, où l'agent prend en charge les décisions d'attribution en temps réel.

Ce pseudocode reflète une stratégie de transition progressive entre un système basé sur des règles et un système intelligent capable de s'adapter aux situations évolutives grâce à

l’apprentissage par renforcement.

```
# === Phase 1 : Génération ou récupération des données via le scoring pondéré
===

Pour chaque commande historique :
    Pour chaque livreur disponible :
        Calculer les critères :
            - Proximité géographique
            - Capacité de charge
            - Historique de performance
            - Temps estimé de livraison
        Calculer le score pondéré total
    Sélectionner le meilleur livreur (meilleur score)
    Enregistrer :
        - état = [proximité, charge, historique, temps estimé]
        - action = index du livreur choisi
        - récompense = performance réelle de la livraison (temps réel, succès,
etc.)
        - next_state = nouvel état du système (si disponible)

# === Phase 2 : Entraînement de l'agent DQN avec ces données ===

Initialiser l’agent DQN (réseau de neurones, mémoire, epsilon, etc.)

Pour chaque épisode (n fois les données) :
    Réinitialiser l’environnement (prendre une commande)
    Tant que la commande n’est pas attribuée (done == False) :
        - Agent choisit une action :
            - Avec proba ε : exploration (action aléatoire)
            - Sinon : exploitation (meilleure action selon le réseau)
        - Appliquer l’action : attribuer la commande au livreur sélectionné
        - Observer la récompense (livraison réussie ? temps rapide ?)
        - Observer le prochain état (facultatif)
        - Stocker la transition (état, action, récompense, next_state, done)
        - Mettre à jour le réseau de neurones (apprentissage DQN)
        - Réduire ε (moins d’exploration à chaque épisode)

# === Phase 3 : Utilisation en production (exploitation uniquement) ===

Quand une nouvelle commande arrive :
    - Récupérer les états des livreurs candidats
    - L’agent DQN choisit le livreur avec la meilleure valeur Q
    - Attribuer la commande
    - Facultatif : stocker l’expérience pour réentraînement plus tard
```

Figure 4.2 Pseudo code du modelé DQN

5. Implémentation du DQN simplifié en Python

Un script a été développé (voir annexe technique) avec :

- un environnement simulé
- un réseau de neurones simple (2 couches)
- une mémoire pour stocker les transitions

- une stratégie épsilon-greedy pour l'exploration

Bibliothèques utilisées : PyTorch, NumPy

6. Avantages de l'approche DQN

- Adaptatif : le modèle apprend des situations précédentes
- Capable de gérer des situations complexes (beaucoup de critères, livreurs, contextes)
- Peut apprendre de la performance réelle du système
- Exploite et explore

7. Intégration dans une application Flutter

Le modèle DQN peut être exécuté dans un serveur Python (à l'aide de Flask), appelé depuis Flutter via HTTP.

- À terme, Flutter envoie les données de la commande ("state")
- Le serveur Python répond avec l'action optimale (le livreur)
- Le retour (récompense) peut être enregistré pour l'entraînement continu

Implémentation du back-end Python avec Flask

Afin de rendre opérationnel le modèle d'agent intelligent basé sur le Deep Q-Network (DQN), un serveur Python léger a été mis en place à l'aide du framework Flask. Ce serveur expose une API REST permettant de recevoir, sous forme d'entrée JSON, l'état d'une commande (vecteur de caractéristiques) et de retourner l'indice du meilleur livreur selon le modèle appris.
Ce système permet à l'application mobile Flutter de déléguer la décision d'attribution au modèle intelligent, tout en maintenant une séparation claire entre l'interface utilisateur et la logique d'intelligence artificielle.
Le code détaillé de cette API est fourni en Annexe 3

Bibliothèques Python utilisées

L'implémentation du back-end repose sur plusieurs bibliothèques Python essentielles :

- Flask : un micro-framework web léger permettant de créer rapidement une API REST pour recevoir les états des commandes et retourner les prédictions du modèle.
- Torch (PyTorch) : utilisé pour la définition, le chargement et l'inférence du modèle de réseau de neurones profond (DQN).
- NumPy : pour la manipulation efficace des tableaux de données numériques (vecteurs d'état par exemple).
- JSON / Requests (intégré via Flask) : pour la gestion des données entrantes/sortantes au format JSON.

  Ces bibliothèques permettent de garantir une interopérabilité fluide entre l'application mobile Flutter et le système d'attribution intelligent basé sur le Deep Q-Network.

Conclusion Générale

Ce projet de fin d'études a eu pour objectif la conception et le développement d'une application mobile innovante visant à automatiser l'affectation intelligente des commandes de livraison aux livreurs disponibles, tout en optimisant les parcours et en facilitant la gestion des flux logistiques urbains. Face à la complexité croissante des défis logistiques dans les zones urbaines, notamment en Algérie, notre démarche a consisté à proposer une solution robuste, performante et intuitive, tirant parti des dernières avancées technologiques en matière de développement mobile multiplateforme et de services géospatiaux.

Au cours de ce travail, nous avons pu valider la faisabilité de notre approche en exploitant la puissance du framework Flutter, qui a démontré son efficacité pour le développement d'une interface utilisateur fluide, moderne et responsive, compatible à la fois avec Android et iOS. L'intégration des services de la Google Maps Platform, notamment les APIs Maps SDK, Routes API, Directions API, Geocoding API, Roads API et Places API, a été cruciale pour la cartographie, la géolocalisation en temps réel, le calcul d'itinéraires optimisés, la localisation d'adresses, et la fiabilité des trajets. En parallèle, Firebase a constitué un pilier de notre infrastructure backend, assurant l'authentification sécurisée des utilisateurs, le stockage des données dans Cloud Firestore, ainsi qu'une synchronisation en temps réel entre les différentes parties prenantes (client, livreur, commerçant, administrateur).

Les phases de conception ont permis de modéliser précisément les besoins fonctionnels et non fonctionnels de l'application, en décrivant les parcours utilisateurs, les rôles, les interfaces et les cas d'usage (passation de commande, réservation de livreur, suivi en temps réel, gestion du menu, etc.). Ces réflexions ont abouti à une architecture modulaire autour d'un modèle client-serveur combiné au paradigme MVC, garantissant extensibilité, maintenabilité et scalabilité.

L'implémentation a permis de concrétiser ces choix à travers une application fonctionnelle, capable de gérer les commandes multi-usagers, suivre les trajets en temps réel, afficher les livreurs disponibles avec estimation dynamique de la distance, du temps et du coût, et appliquer un système d'affectation pondérée basé sur plusieurs critères : proximité géographique, capacité de charge, temps estimé de livraison, historique de performance, et équilibrage de charge. L'algorithme de scoring pondéré mis en place a été validé par un panel d'experts (Jumia Food, Fast Delivery, Yassir Food, anciens restaurateurs et livreurs), garantissant la pertinence des pondérations attribuées à chaque critère.

Les tests menés tout au long du projet ont confirmé la stabilité de l'application, sa conformité aux exigences initiales, et une expérience utilisateur satisfaisante malgré les défis techniques rencontrés, notamment la gestion des permissions, la sécurisation des clés API, l'optimisation des requêtes Google, et l'apprentissage de plusieurs nouvelles technologies en parallèle.

Ce PFE nous a permis d'acquérir une expérience complète du cycle de vie d'un projet informatique, de l'analyse des besoins jusqu'à la mise en œuvre technique, en passant par la

conception, le prototypage, les choix technologiques et l'intégration de services externes. Il a consolidé nos compétences à la fois dans le développement mobile multiplateforme, dans la modélisation des systèmes logistiques, et dans l'usage avancé des APIs cloud géospatiales.

En somme, ce travail représente une contribution concrète et pragmatique au domaine de la logistique urbaine intelligente. Il constitue une base solide pour une potentielle commercialisation ou mise en production à plus grande échelle, ainsi qu'un point de départ pertinent pour des travaux de recherche futurs en optimisation des flux, systèmes intelligents, et services géolocalisés.

Annexe

Annexe 1 : Code de Résolution GBWM en Python

```
from scipy.optimize import linprog

# Objectif : Déterminer les poids des critères selon GBWM
# Méthode : Résolution d'un programme linéaire

# Critères :
# - w1 : Temps estimé de livraison      (Best)
# - w2 : Proximité géographique
# - w3 : Historique de performance
# - w4 : Capacité de charge             (Worst)
# - xi : Variable d’incohérence à minimiser

# Fonction à minimiser : xi (incohérence)
c = [0, 0, 0, 0, 1]  # Seul xi est dans la fonction objectif

# Contraintes d'inégalités (jugements Best-to-Worst transformés en
intervalles)
A = [
    [1, 0, 0, -6, 0],   # w1 - 6*w4 ≤ 0
    [-1, 0, 0, 5, 0],   # -w1 + 5*w4 ≤ 0
    [0, 1, 0, -5, 0],   # w2 - 5*w4 ≤ 0
    [0, -1, 0, 4, 0],   # -w2 + 4*w4 ≤ 0
    [0, 0, 1, -3, 0],   # w3 - 3*w4 ≤ 0
    [0, 0, -1, 2, 0],   # -w3 + 2*w4 ≤ 0
]

b = [0] * len(A)

# Contrainte : somme des poids égale à 1
A_eq = [[1, 1, 1, 1, 0]]
b_eq = [1]

# Bornes : [0,1] pour les poids, xi ≥ 0
bounds = [(0, 1), (0, 1), (0, 1), (0, 1), (0, None)]

# Résolution
res = linprog(c, A_ub=A, b_ub=b, A_eq=A_eq, b_eq=b_eq, bounds=bounds,
method='highs')

# Résultats
if res.success:
    w1, w2, w3, w4, xi = res.x
    print("Résultats du modèle GBWM :")
    print(f"Temps estimé de livraison (w1) : {round(w1, 4)}")
    print(f"Proximité géographique (w2)    : {round(w2, 4)}")
    print(f"Historique de performance (w3) : {round(w3, 4)}")
    print(f"Capacité de charge (w4)        : {round(w4, 4)}")
    print(f"Incohérence (xi)               : {round(xi, 4)}")
else:
    print("Erreur dans la résolution :", res.message)
```

Annexe 2: Algorithme de Sélection Optimale des Livreurs par Méthode de Pondération Multi-Critères

```
Entrées :
- livreurs : liste de livreurs avec leurs notes par critère (notes sur 5)
- poids : dictionnaire des poids normalisés par critère (ex : "proximité": 0.2912,
etc.)
Sortie :
- livreur_choisi : celui avec le score total le plus élevé

Début

  max_score ← -1
  livreur_choisi ← null

  POUR chaque livreur DANS liste de livreurs FAIRE
    score ← 0

    POUR chaque critère DANS poids FAIRE
      note ← note_du_livreur_pour_ce_critere
      score ← score + (note × poids[critère])
    FIN POUR

    SI score > max_score ALORS
      max_score ← score
      livreur_choisi ← livreur
    FIN SI
  FIN POUR

  Retourner livreur_choisi

Fin
```

Annexe 3

BACK-END PYTHON :

```
 # backend_dqn_livreurs.py
from flask import Flask, request, jsonify
import torch
import torch.nn as nn
import numpy as np
# === Modèle du réseau de neurones ===
class QNetwork(nn.Module):
    def __init__(self, state_size, action_size):
        super(QNetwork, self).__init__()
        self.fc1 = nn.Linear(state_size, 64)
        self.fc2 = nn.Linear(64, 64)
        self.out = nn.Linear(64, action_size)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        return self.out(x)
# === Paramètres du modèle ===
STATE_SIZE = 4
ACTION_SIZE = 3
model = QNetwork(STATE_SIZE, ACTION_SIZE)
model.load_state_dict(torch.load("dqn_model.pth",
map_location=torch.device('cpu')))
model.eval()
# === API Flask ===
app = Flask(__name__)
@app.route("/predict", methods=[POST])
def predict():
    data = request.get_json()
    state = data.get("state")

    if not state or len(state) != STATE_SIZE:
        return jsonify({"error": "Entrée invalide"}), 400
    state_tensor = torch.FloatTensor(state).unsqueeze(0)
    with torch.no_grad():
        q_values = model(state_tensor)
        action = torch.argmax(q_values).item()
    return jsonify({"best_livreur_index": action})
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Bibliographie

[1] Flutter Team, "Flutter documentation," Flutter.dev, 2024. . [Online]. Available: https://flutter.dev
[2] Flutter Team, "Flutter widgets catalog," Docs.flutter.dev, 2024. [Online]. Available: https://docs.flutter.dev/ui/widgets
[3] Dart Team, "Dart language overview," Dart.dev, 2024. [Online]. Available: https://dart.dev/language
[4] Dart Team, "Dart documentation," Dart.dev, 2024. [Online]. Available: https://dart.dev/docs
[5] Dart API Reference, "API Dart SDK," api.dart.dev, 2024. [Online]. Available: https://api.dart.dev
[6] Google Developers, "Google Maps SDK for Android," 2024. [Online]. Available: https://developers.google.com/maps/documentation/android-sdk/overview
[7] Google Developers, "Google Directions API Overview," 2024. [Online]. Available: https://developers.google.com/maps/documentation/directions/overview
[8] Google Developers, "Google Places API Overview," 2024. [Online]. Available: https://developers.google.com/maps/documentation/places/web-service/overview
[9] Google Developers, "Google Geocoding API," 2024. [Online]. Available: https://developers.google.com/maps/documentation/geocoding/overview
[10] Google Developers, "Google Roads API Overview," 2024. [Online]. Available: https://developers.google.com/maps/documentation/roads/overview
[11] Google Developers, "Google Routes API Overview," 2024. [Online]. Available: https://developers.google.com/maps/documentation/routes/overview
[12] Google Developers, "Multi-stop Optimization with Routes API," 2024. [Online]. Available: https://developers.google.com/maps/documentation/routes/multi-stop
[13] Google Developers, "ComputeRoutes Reference," 2024. [Online]. Available: https://developers.google.com/maps/documentation/routes/reference/rest/v1/Routes/ComputeRoutes
[14] R. Wenderlich, "Google Maps SDK Tutorial for Android," Raywenderlich.com, 2024. [Online]. Available: https://www.raywenderlich.com/141843/google-maps-sdk-tutorial-for-android-getting-started
[15] Google Codelabs, "Maps Platform for Android," 2024. [Online]. Available: https://codelabs.developers.google.com/codelabs/maps-platform-android
[16] FlutterDevs, "Integrating Google Maps with Flutter," Medium.com, 2024. [Online]. Available: https://medium.com/flutterdevs/integrating-google-maps-with-flutter-29b21d7c6a7f
[17] A. Mourad, "Using Google Maps Roads API with Flutter," Medium.com, 2024. [Online]. Available: https://medium.com/flutter-community/flutter-google-maps-roads-api-40a3eb60f0b4
[18] DevGenius, "Optimize Your Delivery Routes with Routes API," Blog.devgenius.io, 2024. [Online]. Available: https://blog.devgenius.io/optimize-your-delivery-routes-with-google-maps-routes-api-85c9bf47c3f8

[19] Google Developers, "Firebase Authentication," 2024. [Online]. Available: https://firebase.google.com/docs/auth

[20] Google Developers, "Cloud Firestore," 2024. [Online]. Available: https://firebase.google.com/docs/firestore
[21] Google Developers, "Quotas and Billing for Google Maps Platform," 2024. [Online]. Available: https://developers.google.com/maps/billing/gmp-billing
[22] Google Cloud, "API Key Security Best Practices," 2024. [Online]. Available: https://cloud.google.com/docs/authentication/api-keys
[23] CNIL, "RGPD et données de géolocalisation : Ce que doivent faire les acteurs," CNIL.fr, 2024. [Online]. Available: https://www.cnil.fr/fr/rgpd-que-doivent-faire-les-acteurs-qui-utilisent-des-donnees-de-geolocalisation
[24] A. Smith, *Google Maps API Cookbook*, Packt Publishing, 2023.
[25] J. Doe, "Optimization of Vehicle Routing Problem with Time Windows using Google Maps APIs," *Journal of Logistics*, vol. 12, no. 3, pp. 45–60, 2023.
[26] A. Mahmoudi, X. Mi, H. Liao, M. R. Feylizadeh et Z. Turskis, « Grey Best-Worst Method for Multiple Experts Multiple Criteria Decision Making Under Uncertainty », Informatica (Vilnius), vol. 31, no. 2, pp. 331–357, 2020.
[27] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015).
[28] H. S. Shih, H. J. Shyur, et E. S. Lee, “An extension of TOPSIS for group decision making,” Mathematical and Computer Modelling, vol. 45, no. 7–8, pp. 801–813, 2007. DOI: https://doi.org/10.1016/j.mcm.2006.03.023
[29] "Le Reinforcement Learning : La Nouvelle Frontière de l’IA," DataRockstars, 2024. [En ligne]. Disponible : https://www.datarockstars.ai/le-reinforcement-learning-la-nouvelle-frontiere-de-lia/.