

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahman Mira - Bejaia -
Faculté des Sciences Exactes
Département d'Informatique



Mémoire de Fin d'Etudes
En vue de l'obtention du Diplôme de Master en Informatique
Option : Génie Logiciel

Thème

**Conception et réalisation d'une application Android de
géolocalisation dynamique et de réservation de Taxis**

Réalisé par :

M. Mohamed Amine HAMDI
M^{lle} Wassila BENYAHIA

Encadré par :

D^r Houda EL BOUHSSI BRAHAMI

Devant le jury composé de :

Président : D^r Hachem SLIMANI
Examinateur : M. Nabil DJEBARI
Examinateur : M. Yani Athmane BENNAI

- Remerciements -

Nous tenons à remercier vivement notre encadreur **D^r Houda EL BOUHSSI BRAHAMI** qui a accepté de diriger notre travail et pour le temps qu'elle a consacré pour nous orienter, pour ses précieux conseils et pour sa disponibilité.

Nous remercions également les membres du jury composés du **D^r Hachem SLIMANI**, de monsieur **Nabil DJEBARI**, ainsi que de monsieur **Yani Athmane BENNAI** d'avoir accepté d'évaluer notre travail et pour les critiques instructives qu'ils apporteront.

Nous n'oublions pas de rendre hommage à tous nos enseignants du département d'informatique pour toutes les connaissances qu'ils nous ont prodiguées.

Enfin, nous remercions nos parents qui nous ont soutenus moralement et financièrement durant tout notre cursus universitaire.

- Dédicaces -

A nos parents qui nous ont soutenus et encouragés tout le long de notre cursus.

A nos frères et soeurs

A nos ami(e)s

A tous les enseignants du département d'informatique de l'université ABDERRAHMANE MIRA

A tous ceux qui ont contribués, de près ou de loin à la réalisation de ce travail.

Table des matières

Table des matières	i
Table des figures	v
Liste des tableaux	vii
Liste des abréviations	viii
1 Introduction et Motivation	1
1.1 Introduction	1
1.2 Contexte et problématique	1
1.3 Objectifs	2
1.4 Méthodologie de travail	2
1.5 Organisation du mémoire	2
2 Background	4
2.1 Introduction	4
2.1.1 Mobilité	4
2.1.2 Évolution des réseaux mobiles	4
2.2 Applications mobiles	5
2.2.1 Définition	5
2.2.2 Caractéristiques des applications mobiles	5
2.2.3 Les types d'applications mobiles	6
2.2.4 Les principaux avantages et inconvénients des applications mobiles	8
2.3 Systèmes d'exploitations mobiles	9
2.3.1 Définition	9
2.3.2 Les différents systèmes d'exploitation mobiles	10
2.3.3 Comparatif des différents systèmes d'exploitation mobiles	11
2.4 Le système d'exploitation Android	11
2.4.1 Historique	11
2.4.2 Versions	12
2.4.3 Architecture	12
2.4.4 Les principaux avantages et inconvénients d'Android	13

2.4.5	Les principaux atouts qui font le succès du système	14
2.5	Application Android	14
2.5.1	Définition d'une activité	14
2.5.2	Etat d'une activité Android	15
2.5.3	Cycle de vie	16
2.6	Conclusion	18
3	État de l'art	19
3.1	Introduction	19
3.2	Solutions existantes	19
3.2.1	Uber	19
3.2.2	Yassir	21
3.2.3	ITaxi	23
3.2.4	CarDispo	25
3.3	Tableau comparatif des solutions existantes	26
3.4	Problématique	28
3.5	Présentation du sujet	28
3.6	Solution retenue	28
3.7	Conclusion	29
4	Analyse et conception	30
4.1	Introduction	30
4.2	Processus Unifié	30
4.2.1	Définition	30
4.2.2	Les caractéristiques de UP	31
4.3	Langage de modélisation unifié (UML)	33
4.3.1	Définition	33
4.3.2	Les différents diagrammes UML	34
4.4	Spécification des besoins	35
4.4.1	Besoins fonctionnels	35
4.4.2	Besoins non fonctionnels	36
4.5	Analyse des besoins	36
4.5.1	Les acteurs	36
4.5.2	Diagramme de cas d'utilisation	37
4.6	Conception	40
4.6.1	Diagramme d'activités	40
4.6.2	Patrons de conceptions	41
4.6.3	Diagrammes de séquences	42
4.6.4	Diagramme de classes	45
4.6.5	Création de la base de données	46
4.6.6	Diagramme de déploiement	50

4.7 Conclusion	51
5 Expérimentation	52
5.1 Introduction	52
5.2 Présentation de l'environnement de développement	52
5.3 Outils de développements	52
5.3.1 SDK	53
5.3.2 JDK	53
5.3.3 AVD	53
5.4 Langages de développements	53
5.4.1 Java	53
5.4.2 XML	53
5.4.3 JSON	54
5.5 Présentation de Firebase	54
5.5.1 Définition	54
5.5.2 Utilité de Firebase	54
5.5.3 Les services utilisés	54
5.6 Les services web utilisés	55
5.6.1 Google Maps Distance Matrix	56
5.6.2 Place AutoComplete	56
5.6.3 Google Maps Android API	56
5.6.4 Direction API	56
5.6.5 Geocoding API	56
5.7 Geofire	57
5.8 Logiciel de gestion de version	57
5.8.1 Définition	57
5.8.2 Fonctionnalités	57
5.8.3 Les principaux types de logiciels de gestion de versions	58
5.9 Gestion de projets	59
5.9.1 Trello	59
5.9.2 La méthode Kanban	60
5.10 Présentation des interfaces graphiques	60
5.10.1 Interface de choix de l'utilisateur	61
5.10.2 Interface d'inscription	62
5.10.3 Interface principal du client	63
5.10.4 Interface "à propos"	64
5.10.5 Interface de l'historique	65
5.10.6 Etapes de Réservation	66
5.11 Conclusion	71

6 Conclusion générale	72
6.1 Introduction	72
6.2 Contribution	73
6.3 Perspectives et travaux futurs	74
Bibliographie	75

Table des figures

2.1	Les différentes applications mobiles	6
2.2	L'architecture d'android	13
2.3	Illustration d'une activité Android	15
2.4	Fonctionnement d'une pile d'activité	15
2.5	Cycle de vie d'une activité Android	17
3.1	Etape d'inscription à Uber	20
3.2	Indication du point de récupération sur la carte	21
3.3	Indication de la source et de la destination	22
3.4	Estimation du prix/ Chauffeur en route	23
3.5	Interface principal de l'application	24
3.6	Noter la course	24
3.7	Commander un Taxi avec CarDispo	25
3.8	Saisir les informations dans CarDispo	26
4.1	L'architecture du processus unifié	32
4.2	Les diagrammes UML	34
4.3	Diagramme de cas d'utilisation global.	38
4.4	Diagramme global d'activités.	41
4.5	Shéma du MVC	42
4.6	Diagramme de séquence du cas d'utilisation "S'inscrire".	43
4.7	Diagramme de séquence du cas d'utilisation "Réserver un taxi".	44
4.8	Diagramme de classes.	46
4.9	Exemple d'une BDD NoSQL orienté document.	48
4.10	Shéma de notre BDD.	49
4.11	Diagramme de déploiement	50
5.1	Logiciel centralisé	58
5.2	Logiciel distribué	59
5.3	Exemple d'un tableau Trello	60
5.4	Interface de choix de l'utilisateur.	61
5.5	Etape d'inscription	62
5.6	Menu du client.	63

5.7	Interface à propos	64
5.8	Interface de l'historique	65
5.9	Le client indique la destination/Il choisi un taxi	66
5.10	Cliquer sur recherche/modifier la destination	67
5.11	Le client confirme l'envoi de la demande	68
5.12	Etapes de réception et d'acceptation d'une demande de réservation	69
5.13	Le client accepte le prix	70

Liste des tableaux

2.1	Tableau comparatif des applications natives et web	8
2.2	Les principaux avantages et inconvénients des applications mobiles	9
2.3	Les différents systèmes d'exploitation mobiles.	10
2.4	Comparatif des différents systèmes d'exploitation mobiles.	11
2.5	Les versions d'android les plus utilisées	12
2.6	Les différents états d'une activité	16
3.1	Tableau comparatif des solutions existantes.	27
4.1	Les différents cas d'utilisation	37
4.2	Description du cas d'utilisation "S'inscrire".	39
4.3	Description du cas d'utilisation "Gérer les demandes".	39
4.4	Description du cas d'utilisation "Réserver un Taxi".	40
4.5	Dictionnaire de données.	45

Liste des abréviations

LTE	Long Term Evolution
VTC	Véhicule de Tourisme avec Chauffeur
PDA	Personal Digital Assistant
WIFI	Wireless Fidelity
GPRS	General Packet Radio Service
EDGE	Enhanced Data GSM Environment
GPS	Global Positioning System
GSM	Global System for Mobile
PC	Personal Computer
OS	Operating System
REST	REpresentational State Transfer
CSS	Cascading Style Sheets
HTML	Hyper Text Markup Language
UML	Unified Modeling Language
UP	Unified Process
NoSQL	Not only SQL
BDD	Base De Données
MVC	Model View Controller
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
SDK	Software Development Kit
JDK	Java Development Kit
JRE	Java Runtime Environment
AVD	Android Vistual Device
XML	Extensible Markup Language
JSON	Java Script Object Notation
API	Application Programming Interface
SI	Système d'Information

Chapitre 1

Introduction et Motivation

1.1 Introduction

Depuis l'apparition de l'internet, la façon de penser et de vivre dans le monde a été révolutionnés, elle a permis aux consommateurs de faire des transactions, et d'accomplir leurs tâches sans devoir se déplacer physiquement.

Une dizaine d'années après, cette innovation est suivie par l'apparition de la technologie mobile qui a pris une place importante dans notre société, les assistants personnels, téléphones cellulaires, smartphones, tablettes, etc.

Les moyens de connexion, comme les réseaux sans fil (Wifi, GPRS et d'autres) ont permis de suivre et d'accéder aux informations dont nous aurons besoin partout où il y a une couverture réseau et cela se fait à l'aide d'applications mobiles.

De nos jours, avoir un dispositif mobile est devenu incontournable pour tout individu, ce qui nous a poussé et motivé à réfléchir sur une application mobile utile.

1.2 Contexte et problématique

Depuis quelques années, l'offre de transport public n'a cessé de se développer et de s'améliorer, et avec elle les demandes et les exigences des usagés, rendant ces moyens de transport moins compétitifs surtout dans les milieux ruraux et périurbains. L'utilisation des taxis reste assez répondu dans ces zones de par son confort et son efficacité malgré les quelques inconvénients qu'ils possèdent en l'occurrence sa disponibilité en temps voulu, le temps nécessaire pour trouver un taxi, ainsi que le coût élevé.

1.3 Objectifs

Notre travail consiste à explorer les solutions existantes relatives à la réservation de taxi et à concevoir et développer une application mobile sous Android pour la réservation de taxi en ligne. Cette application est destinée, à la fois, aux clients et aux chauffeurs de taxis et sera installée sur terminaux mobiles.

Ainsi ce mémoire détaille les différentes phases théoriques et pratiques de notre application.

1.4 Méthodologie de travail

La démarche adoptée pour notre travail est guidée par de nombreuses questions issues des préoccupations de la communauté des utilisateurs de taxis.

Etant donné que la recherche de taxi est un processus parfois long et difficile, nous nous sommes forcés tout au long de ce mémoire de prendre suffisamment de recul afin de proposer un cadre méthodologique relativement global et suffisamment complet afin de mieux aider et mieux guider les utilisateurs dans leur processus.

Notre démarche de travail repose plus précisément sur les étapes suivantes :

- Etape de recherche et d'analyse : qui consiste à établir un état de l'art des différentes solutions proposées dans le cadre de la recherche de taxi avec une comparaison des avantages et inconvénients des approches proposées.
- Etape d'identification du problème et de proposition de solution : qui permet de définir la problématique et la solution proposée en vigueur.
- Etape d'implémentation et d'expérimentation des systèmes proposés : qui met en évidence le système proposé, son fonctionnement et son intérêt, accompagnée d'études de cas pour la validation, sans oublier l'implémentation de notre système.

1.5 Organisation du mémoire

La présente introduction et motivation fournit une présentation générale du sujet, le contexte ainsi que la problématique.

Le reste du mémoire est organisé comme suit :

- **Chapitre 2 :** Ce chapitre intitulé «Background» porte sur les applications mobiles et leurs types ainsi que les différents systèmes d'exploitation mobiles existants.
- **Chapitre 3 :** Ce chapitre «État de l'art» comporte une présentation du sujet, un état de l'art des solutions proposées et la solution retenue.

- **Chapitre 4 :** Le quatrième chapitre sera consacré à l’analyse et la conception du système à travers la présentation des différents diagrammes.
- **Chapitre 5 :** Ce chapitre présente l’environnement et les outils de développement utilisés pour la réalisation et la mise en œuvre de notre application « Taxi06 » avec des copies d’écran du système proposé.

Enfin ce mémoire se termine par une conclusion où nous avons expliqué l’intérêt du sujet et ce que nous avons réalisé et ce qui reste à faire ainsi que quelques perspectives et des directions pour des travaux futurs.

Chapitre 2

Background

2.1 Introduction

Les technologies de l'information et de la communication évoluent à une vitesse vertigineuse, c'est à cet effet qu'un centre d'innovation technologique LTE (Long Term Evolution) a été inauguré, ce dernier est situé dans le Centre d'études et de recherche des télécommunications et des technologies de l'information et de la communication (Certic) au niveau du Cyberparc de Sidi Abdellah, le centre d'innovation technologique LTE est le fruit d'un partenariat entre l'Algérie et la Chine, à travers la société Huawei, un équipementier présent en Algérie depuis de longues années, offrant ainsi des vitesses supérieures à la téléphonie mobile.

Dans ce chapitre, nous allons en premier lieu parler des réseaux mobiles, ensuite nous allons passer à l'introduction des applications mobiles ainsi qu'aux différents types, nous entamerons ensuite les différents systèmes d'exploitation mobiles existants.

2.1.1 Mobilité

La mobilité dans les réseaux de communication est définie comme la capacité d'accéder, à partir de n'importe où, à l'ensemble des services disponibles normalement dans un environnement fixe et câblé.

Tandis que l'informatique mobile est définie comme la possibilité pour des usagers munis de périphériques portables ou d'ordinateurs mobiles d'accéder à des services et à des applications évoluées, à travers une infrastructure partagée de réseau, indépendamment de la localisation physique ou du mouvement de ses usagers [5]

2.1.2 Évolution des réseaux mobiles

Les réseaux mobiles ont connu un développement progressif et ce depuis 1983 avec l'apparition de la première génération (1G) qui possédait un fonctionnement analogique. Suivie de la deuxième génération (2G) en 1991 avec l'inauguration des réseaux GSM, GPRS et EDGE. La connexion devient permanente

sur le mobile avec l'apparition de la 3G. Enfin, la venue de la quatrième génération (4G), permet de maintenir des débits mobiles élevés [5]

2.2 Applications mobiles

2.2.1 Définition

Une application mobile se définit comme étant un logiciel téléchargeable et qu'on installe facilement sur son smartphone comme nous le faisons avec tout logiciel sur notre PC portable. Le téléchargement de l'application mobile se fait suivant deux options [6] :

- Sur téléphone par le biais de connexion internet.
- Sur PC en le branchant avec le téléphone mobile.

Chez les mobinautes, l'application mobile est similaire à un site internet pointu en raison de sa connexion à internet, de plus l'interface de site et de l'application mobile s'avèrent identiques sauf que l'application demeure fondamentalement définie comme un logiciel. En ce sens, les applications mobiles se regroupent en plusieurs séries suivant des critères basiques :

- **Applications fonctionnant sans internet** : Appelées applications indépendantes, ce sont des applications qui fonctionnent sans avoir besoin de connexion internet ou téléphonique dont on cite liste de contacts, calculatrice et autres.
- **Applications exigeant une connexion** : Contrairement aux applications indépendantes, ces applications doivent avoir accès à l'internet pour fonctionner.
- **Applications connectées** : C'est une application qui nécessite une connexion internet pour un bon fonctionnement.
- **Applications interagissant avec les autres équipements de smartphone** : tout smartphone dispose d'une suite d'équipements pointus davantage que les téléphones portables ordinaires et ces équipements sont en interaction permanente avec certaines applications. On en cite le scan de code-barres ou code pour savoir le prix et les caractéristiques d'un produit vendu sur les hypermarchés.
- **Application interagissant avec autres mobinautes** : La génération Y est familière à se connecter sur internet et le téléphone mobile ce qui justifie l'importance des applications qui renforcent les liaisons entre les mobinautes au lieu de se connecter passivement sur les réseaux sociaux [6].

2.2.2 Caractéristiques des applications mobiles

Pour pouvoir dire qu'une application mobile est réussie, elle doit avoir certains critères très importants qui sont les suivants [7] :

- Rapidité et fluidité, un premier critère pour une application réussie.
- Le poids de l'application mobile, un facteur important pour les utilisateurs.

- L'ergonomie, un critère pour fidéliser l'utilisateur.
- Une bonne qualité d'affichage, un critère pour séduire l'utilisateur.
- Maintenance et évolutivité, la clé pour un bon suivi.

2.2.3 Les types d'applications mobiles

Il existe 3 types d'applications mobiles : application web, application native et hybride, comme le montre la figure appMobile suivante :

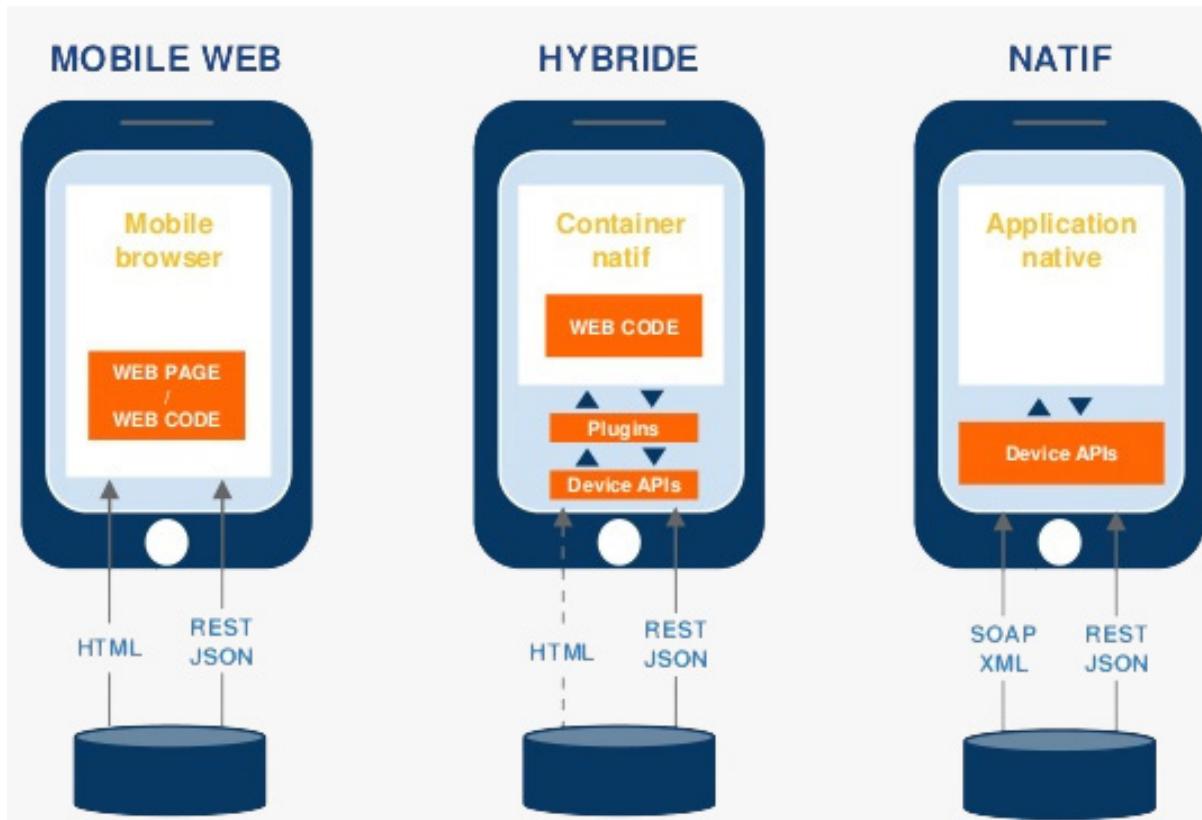


FIGURE 2.1 – Les différentes applications mobiles [6].

Dans ce qui suit, nous allons présenter ces types avec plus de détail.

- **Application web**

Toute application conçue avec HTML et CSS de plus opérationnelle sur navigateur internet pour un smartphone est appelée application web, en d'autres termes la version mobile d'un site web, c'est une application web.

Le but d'une application web est de rendre le contenu disponible, ou du moins fonctionnel sur mobile. Elle se comporte comme une application normale, ressemble à une application normale, mais

son accessibilité et son prix n'ont rien à voir avec les applications mobiles.

Peu importe la marque de votre smartphone, vous pouvez accéder à l'application web par le biais de son navigateur et donc vous n'avez pas besoin de la télécharger. Vu qu'elle ne tient pas en compte les divergences persistantes entre les systèmes d'exploitation et les marques de smartphone, l'application web manque d'ergonomie et de plus elle ne se sert pas de la mémoire du smartphone ce qui la place en infériorité par rapport à l'application native [6].

• **Application native**

Il s'agit d'application conçue pour une grande partie de systèmes d'exploitation fiables par les smartphones en se référant à un langage particulier à chacun d'eux. Ce mode d'application est accessible seulement sur les plateformes d'applications suivent ses particularités et ses formules. Le développement de l'application native nécessite le recours à la mémoire du smartphone sans omettre les options reliées au système d'exploitation en question. De cette façon, le résultat se résume dans l'aboutissement à des applications mobiles avec des fonctions plus professionnelles et plus performantes que les applications développées en HTML5/CSS3 et les applications hybrides [6].

Les applications natives sont meilleures pour les raisons suivantes :

- Meilleure rapidité, fiabilité et dotée d'une meilleure réactivité ainsi qu'une résolution supérieure ce qui assure une meilleure expérience utilisateur.
- Elle permet un accès plus facile à toutes les fonctionnalités du téléphone, de l'accéléromètre en passant par la caméra et même le micro.
- Les notifications push, uniquement disponibles sur les applications natives. Ces notifications vous permettent d'alerter vos utilisateurs et d'attirer leur attention chaque fois que vous le souhaitez, que ce soit pour du nouveau contenu ou une offre promotionnelle.
- Ne requiert pas forcément internet pour fonctionner, ce qui est un réel avantage. Car même en ce temps présent, il existe encore des zones très peu couvertes par le réseau internet, et permettre à ses utilisateurs d'accéder à l'application sans connexion web est un très gros point fort à ne pas négliger.

• **Application hybride**

Il s'agit d'une application mobile qui fusionne les caractéristiques d'application web (développement en HTML 5) et celles de l'application native. De cette manière, l'application mobile sera accessible sur toutes les plateformes d'application.

Ce type d'application mobile minimise les charges et la durée de son développement même si cela sera au détriment de perfectionnement et de la qualité qui caractérise l'application native [6].

Le tableau suivant représente une comparaison des applications natives et les applications web [8].

Critères	Applications native	Application web
Portabilité	Développement spécifique à chaque plateforme	Navigateur web, mais une intégration distincte selon la plateforme.
Référencement	Arriver à se positionner dans une boutique d'applications Accessible par la recherche dans une boutique d'applications.	Accessible par les moteurs de recherche classiques et liens externes éventuels
Accessibilité technique	Dépendante de la plateforme et de l'éventuelle validation par une boutique d'applications mode offline possible.	Éventuelle dépendance aux navigateurs mode offline impossible, support HTML5 nécessaire
Exploitation du mobile	Utilise toutes les possibilités du mobile (GPS, contacts, camera, voix... etc.)	Se limite aux possibilités du navigateur.
Développement/coût	Plus long, plus fastidieux nécessite un sdk+la connaissance d'un langage spécifique	Généralement moins onéreux, HTML/JavaScript/CSS.
Effet immersif	Plus de possibilités, richesse fonctionnelle et multimédia, logique marketing forte.	Limité. Des possibilités plus importantes avec l'arrivée de HTML5.
Expérience utilisateur	Maximale. Possibilité de notifier l'utilisateur (Push) même quand l'application n'est pas utilisée	Limitée mais conforme à l'utilisation classique du web
Mise à jour	Processus de soumission à un magasin d'application. Constrained dans le cas de l'App Store d'Apple donc mise à jour en mode par action de l'utilisateur.	Mise à jour instantanée sur le serveur web, en une seule opération
Développement	Dépend de la compétence existante sur le SDK du mobile visé	Compétence HTML/CSS/JavaScript Plus classique.
Potentiel performance	Maximum	Dépend du développement du site, de la connexion

TABLE 2.1 – Tableau comparatif des applications natives et web[8].

2.2.4 Les principaux avantages et inconvénients des applications mobiles

Les applications mobiles présentent un certain nombre d'avantages du fait qu'elles favorisent la mobilité et l'accès direct, cependant elles possèdent quelques inconvénients.

Le tableau ci-dessous représente les avantages ainsi que les inconvénients des 3 types d'applications mobiles cités précédemment [9].

Types d'application	Avantages	Inconvénients
Application native	<ul style="list-style-type: none"> Accessibilité directe de l'application en mode hors connexion. Meilleure expérience utilisateur. Meilleur référencement due aux téléchargements sur les plateformes comme l'App Store ou le Play Store. 	<ul style="list-style-type: none"> Pas la même application sur les différentes plateformes (Apple, Android). Coût de développement important. Problème de compatibilité après les mises à jour.
Application web	<ul style="list-style-type: none"> Un seul et unique code pour les différentes plateformes. Coût de développement moins important. Compatible avec tous les navigateurs. 	<ul style="list-style-type: none"> Non accessible en mode hors connexion (sauf s'il y a une mise en cache du site). Ne peut pas accéder aux applications natives du mobile (GPS, appareil photo...). Manque de fluidité sur les anciens modèles de smartphones.
Application hybride	<ul style="list-style-type: none"> Un seul et unique code pour les différentes plateformes. Coût de développement moins important. Accessibilité direct de l'application hors connexion. Disponibilité de l'application mobile sur les stores. 	<ul style="list-style-type: none"> Une ergonomie pas forcément optimisée. Performance de l'application par rapport à un développement natif. Difficultés de trouver le bon prestataire digital.

TABLE 2.2 – Les principaux avantages et inconvénients des applications mobiles[9].

Enfin, notre choix s'est porté sur le développement d'une application native sous Android principalement grâce aux multiples avantages dont elle dispose et qu'on a cité précédemment, de plus notre projet consistera à développer une application spécialement conçue pour le système d'exploitation mobile Android ce qui fait qu'une application native correspond parfaitement à ce qu'on veut réaliser dans ce projet.

2.3 Systèmes d'exploitations mobiles

2.3.1 Définition

Les systèmes d'exploitation mobiles (OS) peuvent être définis comme les logiciels permettant à un smartphone ou un téléphone mobile basique de fonctionner. Ils permettent de ce fait aux utilisateurs de

pouvoir passer un appel téléphonique, naviguer sur leurs téléphones parmi toutes les rubriques, télécharger des applications ou encore paramétrier et personnaliser leurs smartphones [10].

2.3.2 Les différents systèmes d'exploitation mobiles

Il existe différents systèmes d'exploitation mobiles, qui sont présentés brièvement ci-dessous.

Système d'exploitation	Définition
	Android est un système d'exploitation et plate-forme logicielle pour smartphones et tablettes créé par Google dont la première version beta a été proposée en Novembre 2007[10].
	Le système d'exploitation bada a était développé par Samsung. Il s'agit d'un système d'exploitation qui bénéficie de plus de 550 applications à télécharger [11].
	BlackBerry OS est un système d'exploitation mobile conçu spécifiquement pour les terminaux BlackBerry RIM. Il fonctionne sur les téléphones BlackBerry variant, tels que BlackBerry Bold, Curve, Pearl et Storm [10]
	iOS est le nom définitif du système d'exploitation “iPhone OS” développé par Apple pour les appareils iPhone, iPad, iPod Touch, Apple TV, iOS est un dérivé du système Mac OS X et reprend les fondations avec le noyau hybride XNU, les services Unix, Cocoa [10].
	Symbian est un système d'exploitation (OS) pour téléphone mobile. Développé par un consortium de constructeurs du secteur, il se destine aux smartphones. Ces appareils disposent de fonctions d'agenda, de carnet d'adresses, mais ils acceptent des programmes supplémentaires à la manière d'un PC [11].
	Le Windows Phone est un système d'exploitation spécialement développé pour un usage mobile. Développé par Microsoft [10].

TABLE 2.3 – Les différents systèmes d'exploitation mobiles.

2.3.3 Comparatif des différents systèmes d'exploitation mobiles

Il existe des tonnes de différences entre les différents systèmes d'exploitation existants, dans le tableau suivant nous avons souligné les différences les plus pertinentes entre ces derniers [13]

	Android	Bada	BlackBerry OS	iOs	Symbian OS	Windows Phone
Appareils compatibles	Samsung galaxy,HTC	Samsung waves 3	BlackBerry torch, bold, curve	Iphone,ipod ,ipad	N8,N9	Windows phone, Nokia Lumia 710
Dernière version	7.1	2.0.5	10	10. 0.2	Nokia Belle	10.0.14393 321
Date de sortie	4/10/2016	15/03/2012	30/01/2013	23/08/2016	01/08/2011	11/10/2016
Open source	✓	X	X	X	✓	X
Mis à jour	29/04/2017	23/01/2017	28/04/2017	28/04/2017	23/01/2017	28/04/2017
Details techniques						
Support d'adobe flash	✓ Intégré directement dans les applications	✓ FlashLite3.1	X	X	✓	✓
Marché						
Place de marché	Google Play	Samsung Apps	BlackBerry App world	App store	Ovi Store	Windows Phone marketplace
Nombre d'application	+800 000	3000(Q1 2011)	+70 000	+1000000	+30 000	+9 000

TABLE 2.4 – Comparatif des différents systèmes d'exploitation mobiles.

2.4 Le système d'exploitation Android

Android est un système d'exploitation et plate-forme logicielle pour smartphones et tablettes créé par Google dont la première version beta a été proposée en Novembre 2007 [10].

2.4.1 Historique

L'histoire d'Android commence en octobre 2003, où la société Android Inc. est créée. Officiellement, elle développe des logiciels pour mobiles. Mais en réalité, elle se prépare à sortir un tout nouveau

système d'exploitation pour smartphones. En 2005, Google rachète cette entreprise, et sort une première bêta en novembre 2007, avant de lancer la version 1.0 en septembre 2008 avec le HTC Dream. À partir de ce moment-là, le rythme des nouvelles sorties est très élevé : pas moins de 11 versions différentes sont sorties en 3 ans. On remarquera au passage que chaque version d'Android porte le nom d'un dessert[14].

2.4.2 Versions

Le tableau ci-dessous montre les versions les plus utilisées d'Android ainsi que leurs distributions

Version	Nom de code	API	Distribution
2.3.3-2.3.7	GingerBread	10	0.5%
4.0.3-4.0.4	Ice Cream Sandwich	15	0,5%
4.1x	Jelly Bean	16	2.2%
4.2x	Jelly Bean	17	3.1%
4.3	Jelly Bean	18	0.9%
4.4	KitKat	19	13.8%
5.0	Lollipop	21	6.4%
5.1	Lollipop	22	20.8%
6.0	Marshmallow	23	30.9%
7.0	Nougat	24	17.6%
7.1	Nougat	25	3%
8.0	Oreo	26	0.3%

TABLE 2.5 – Les versions d'android les plus utilisées[15].

2.4.3 Architecture

Android est basé sur un noyau (kernel) linux 2.6.xx, au-dessus du kernel on retrouve le hardware abstraction layer, qui permet de séparer la plateforme logique du matériel. Au-dessus de cette couche d'abstraction, on retrouve les librairies C/C++ utilisées par un certain nombre de composants du système Android. Ensuite, on retrouve l'Android Runtime, cette couche contient les librairies cœurs du Framework ainsi que la machine virtuelle exécutant les applications. Au-dessus la couche "Android Runtime" et des librairies cœurs on retrouve le Framework permettant au développeur de créer des applications. Enfin la dernière couche est la couche application. Le schéma suivant illustre les principaux composants du système d'exploitation Android que l'on a expliqué brièvement [16].

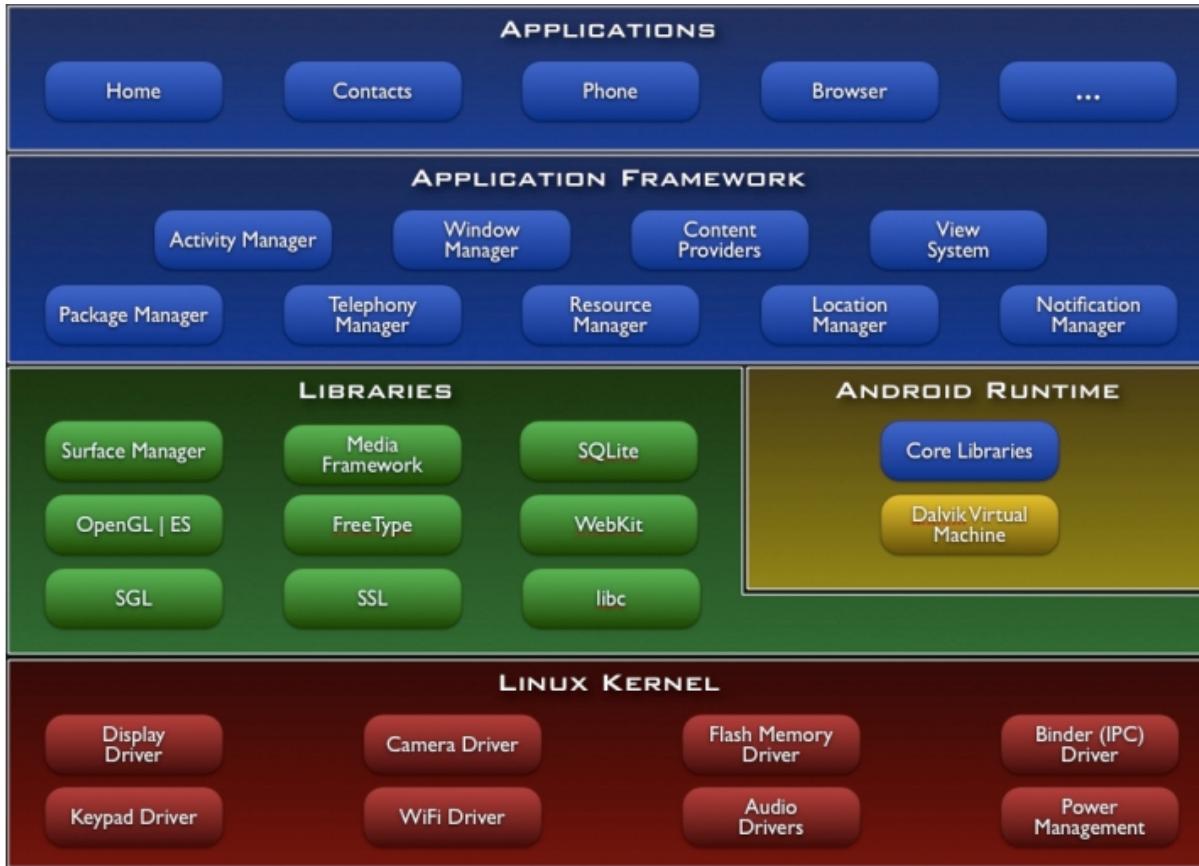


FIGURE 2.2 – L’architecture d’android [16].

2.4.4 Les principaux avantages et inconvénients d’Android

Cette flexibilité, qui contraste avec la maîtrise totale qu’Apple exerce sur iOS, fait à la fois la force et la faiblesse d’Android. Force de par son ouverture, sa compatibilité avec des langages de programmation répandus (Java Development Kit) et sa capacité à toucher un très large public en équipant aussi bien des téléphones haut de gamme que premier prix. Faiblesse car Android est fragmenté en de multiples versions, avec pas moins de 16 itérations publiées entre 2007 et 2015.

Ceci pose de nombreux problèmes de compatibilité des applications et de sécurité. En 2015, on dénombrait plus de 24.000 terminaux Android distincts proposés par 1.294. Cette année-là, une importante faille de sécurité baptisée Stagefright, qui touchait potentiellement 95% des mobiles sous Android, est venue illustrer avec fracas cette situation. À partir de 2014, Google a commencé à décliner Android pour partir à la conquête de nouveaux marchés : Android TV pour les téléviseurs connectés, Android Auto pour les voitures et Android Wear pour les montres connectées [12].

2.4.5 Les principaux atouts qui font le succès du système

Le petit robot vert nommé «BugDroid» a incontestablement conquis le monde entier [16] car :

- Le projet est open source et gratuit.
- Le système est évolutif.
- Facile à développer
- Facile à vendre
- Le développement est accessible.

2.5 Application Android

Une application Android est une application mobile spécifiquement développée pour les smartphones utilisant le système d'application Android acheté et développé par Google. Comme les applications iPhone dont elles sont souvent des répliques, les applications Android sont de nature très variables : jeux, mobile commerce, utilitaire, service d'information [16].

2.5.1 Définition d'une activité

Si vous observez un peu l'architecture de la majorité des applications Android, vous remarquerez une construction toujours à peu près similaire. Prenons par exemple l'application du Play Store. Vous avez plusieurs fenêtres à l'intérieur même de cette application : si vous effectuez une recherche, une liste de résultats s'affichera dans une première fenêtre et si vous cliquez sur un résultat, une nouvelle fenêtre s'ouvre pour vous afficher la page de présentation de l'application sélectionnée. Au final, on remarque qu'une application est un assemblage de fenêtres entre lesquelles il est possible de naviguer.

Ces différentes fenêtres sont appelées des activités. Un moyen efficace de différencier des activités est de comparer leur interface graphique : si elles sont radicalement différentes, c'est qu'il s'agit d'activités différentes. De plus, comme une activité remplit tout l'écran, une application ne peut en afficher qu'une à la fois. La figure suivante illustre ce concept [16].

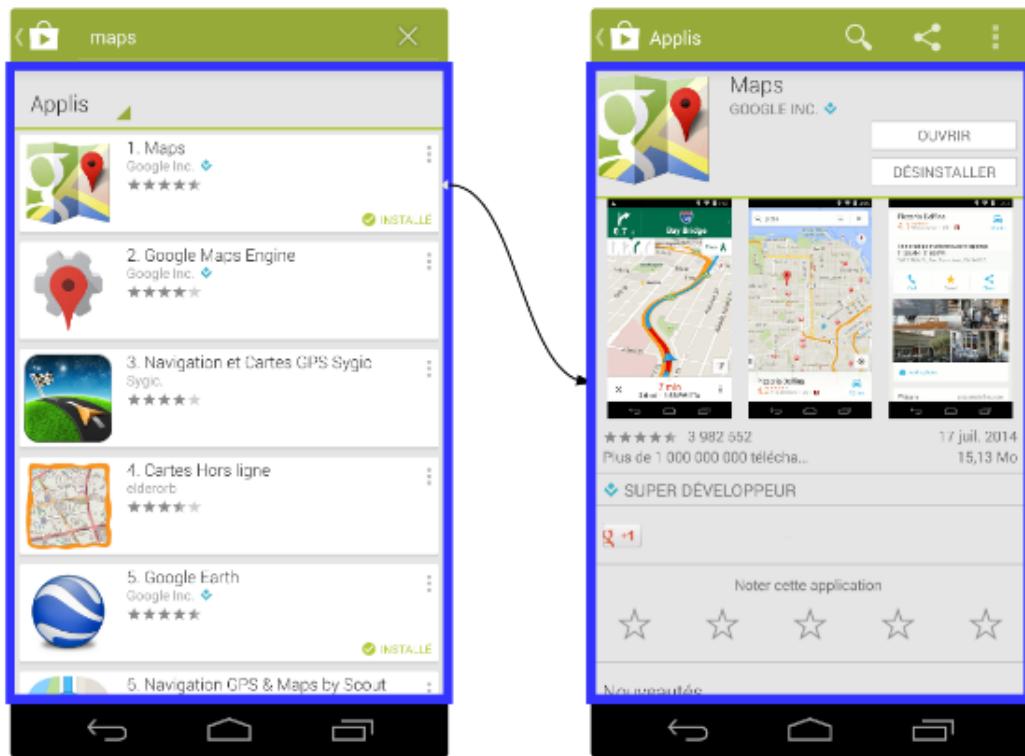


FIGURE 2.3 – Illustration d'une activité Android [16].

2.5.2 Etat d'une activité Android

Quand une application se lance, elle se met tout en haut de ce qu'on appelle la pile d'activités comme le montre la figure suivante :

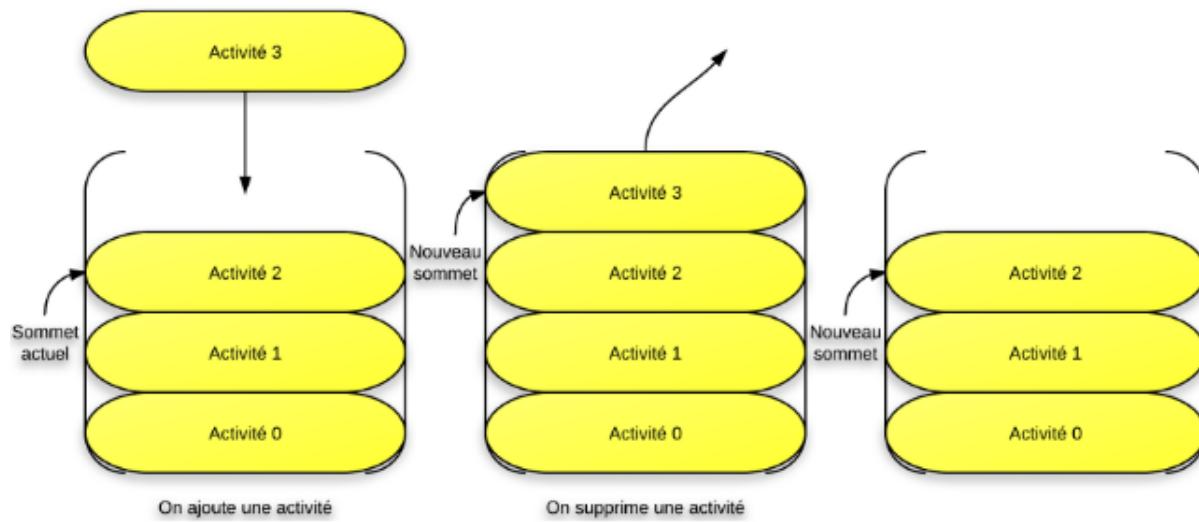


FIGURE 2.4 – Fonctionnement d'une pile d'activité [16].

L'activité que voit l'utilisateur est celle qui se trouve au-dessus de la pile. Ainsi, lorsqu'un appel arrive, il se place au sommet de la pile et c'est lui qui s'affiche à la place de notre application, qui n'est plus qu'à la deuxième place. Notre activité ne reviendra qu'à partir du moment où toutes les activités qui se trouvent au-dessus d'elle seront arrêtées et sorties de la pile. Une activité peut se trouver dans trois états qui se différencient surtout par leur visibilité comme le montre le tableau suivant :

Etat	Visibilité	Description
Active (« active » ou « running»)	L'activité est visible en totalité	Elle est sur le dessus de la pile, c'est ce que l'utilisateur consulte en ce moment même et il peut l'utiliser dans son intégralité. C'est cette application qui a le focus, c'est-à-dire que l'utilisateur agit directement sur l'application.
Suspendue (« paused»)	L'activité est partiellement visible à l'écran. C'est le cas quand vous recevez un SMS et qu'une fenêtre semi-transparente se pose devant votre activité pour afficher le contenu du message et vous permettre d'y répondre par exemple.	Ce n'est pas sur cette activité qu'agit l'utilisateur. L'application n'a plus le focus, c'est l'application au-dessus qui l'a. Pour que notre application récupère le focus, l'utilisateur devra se débarrasser de l'application partiellement au-dessus pour que l'utilisateur puisse à nouveau interagir avec notre activité. Si le système a besoin de mémoire, il peut très bien tuer l'application.
Arrêtée (« stopped»)	L'activité est tout simplement invisible pour l'utilisateur, car une autre activité prend toute la place sur l'écran.	L'application n'a évidemment plus le focus, et puisque l'utilisateur ne peut pas la voir, il ne peut pas agir dessus. Le système retient son état pour pouvoir reprendre, mais il peut arriver que le système tue votre application pour libérer de la mémoire système.

TABLE 2.6 – Les différents états d'une activité [16].

2.5.3 Cycle de vie

Le cycle de vie d'une activité correspond aux différents états d'une activité lors de sa gestion par le système Android. Il est très important car il va vous permettre de suivre l'état de votre activité au fur et à mesure de son existence dans le système Android [16].

La figure suivante illustre le schéma du cycle de vie d'une activité Android [16].

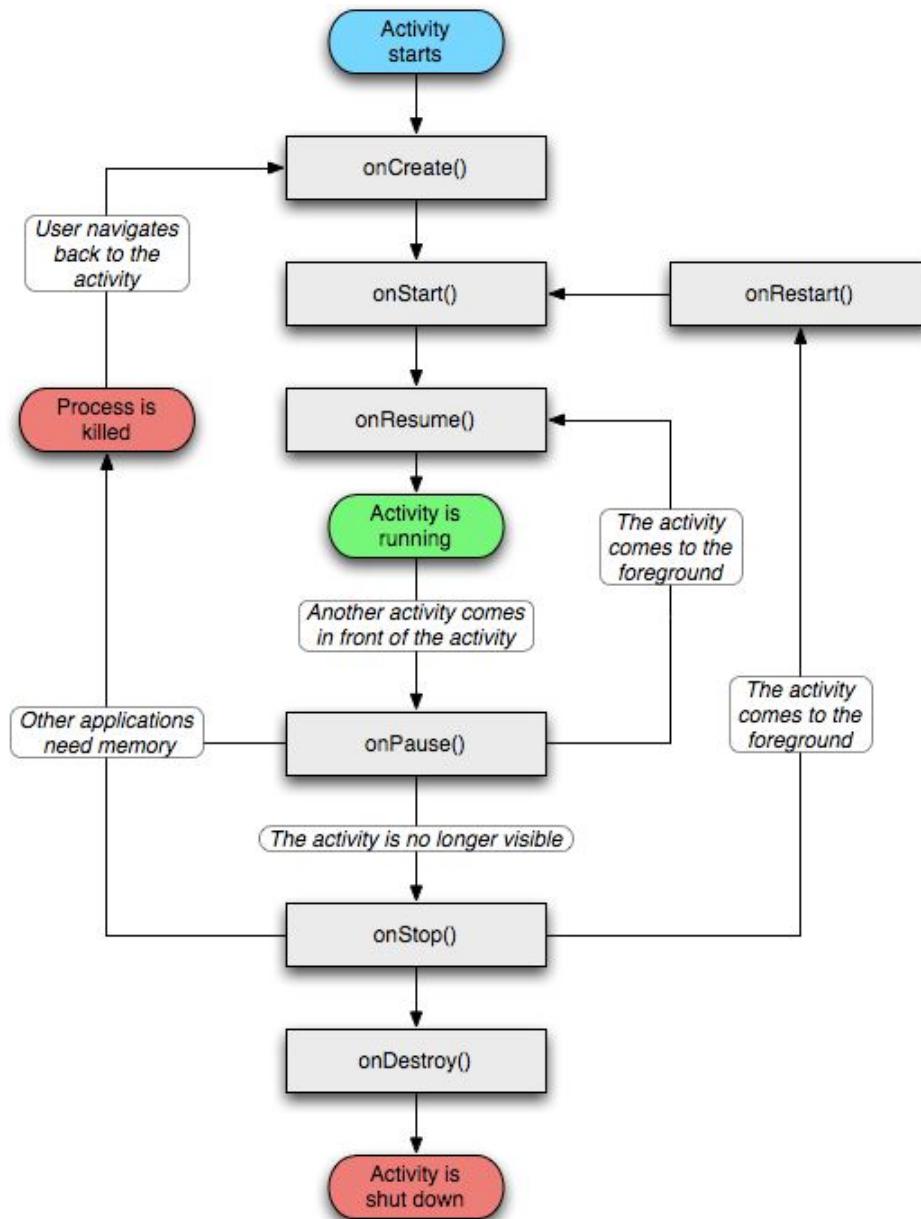


FIGURE 2.5 – Cycle de vie d’une activité Android [16].

- Lorsque l’activité est lancée, le système Android appelle les méthodes `onCreate`, `onStart` et `onResume`.
- Lorsque l’activité s’arrête, le système Android appelle les méthodes `onPause`, `onStop` et `onDestroy`.
- Lorsque l’activité n’est plus au premier plan, mais qu’elle est tout de même affichée, `onPause` est appelée. Lorsque l’activité retourne au premier plan, `onResume` est appelée.
- Lorsque l’activité n’est plus affichée, `onResume` et `onStop` sont appelées. Lorsque l’activité est de nouveau affichée, `onRestart`, `onStart` et `onResume` sont appelées.
- Lorsque le système Android décide de tuer votre application, il appelle la méthode `onSaveInstanceState` qui vous permet de sauvegarder l’état de votre activité. Cet état sera passé en paramètre à la méthode `onCreate`, afin que vous puissiez restaurer l’état de l’activité.

- Le système Android s'occupe lui-même de sauvegarder et restaurer l'état des vues des layouts, à condition que ces vues aient chacune un ID unique dans le layout.
- Le changement de configuration comme le changement d'orientation de l'écran provoque un redémarrage de l'activité, vous devez donc sauvegarder et restaurer l'état dans ce cas.

2.6 Conclusion

Dans ce chapitre, nous avons mis le point sur les applications mobiles et les différents systèmes d'exploitation existants, en se focalisant sur le système d'exploitation Android, car c'est ce dernier qu'on utilisera pour notre application.

Le prochain chapitre sera consacré à l'analyse et la conception de notre système, en spécifiant les différents besoins de notre application, nous procèderons ensuite à la modélisation de ces besoins en s'appuyant sur le langage UML (Unified Modeling Language).

Chapitre 3

État de l'art

3.1 Introduction

De nos jours, les systèmes logiciels sont devenus indispensables dans de nombreux domaines. La taille et la complexité des logiciels est en croissance continue. Les systèmes informatiques sont devenus de plus en plus complexes et hétérogènes avec un marché toujours plus exigeant et évolué en matière de performance, d'efficacité, d'innovation, de compétitivité et de productivité.

En particulier, les applications mobiles sont devenues la partie prenante d'usage quotidien, sur un simple appareil mobile, nous pouvons enregistrer un tas d'application utiles et indispensables.

Dans ce chapitre nous allons donner un aperçu du domaine d'étude, la problématique que nous avons formulée, aussi nous allons présenter un état de l'art de quelques solutions existantes et les objectifs à atteindre.

3.2 Solutions existantes

Cette section présente quelques solutions existantes de réservations de taxi

3.2.1 Uber

Uber, un service né de la culture du numérique. La société Uber – anciennement UberCab, est née au cœur de la Silicon Valley, à San Francisco. L'idée derrière cette application est de proposer aux utilisateurs des voitures de tourisme avec chauffeur - VTC est le terme consacré - à des prix abordables. Pour plus de simplicité, la réservation s'opère directement depuis le smartphone à l'aide de la géolocalisation. Comme de nombreux services en ligne, Uber mise énormément sur le sentiment d'appartenance à une communauté.

Pour arriver à ses fins, l'application propose à ses clients un service haut de gamme. L'entreprise Uber exige par exemple que de l'eau soit mise à disposition des clients dans toutes les berlines. En fonction

du service choisi, les chauffeurs peuvent parfois vous proposer des boissons énergétiques, des sucreries, etc. Chez Uber, la communauté est reine. Il est important de tisser des liens entre clients, chauffeurs et maison-mère. Enfin, toutes les transactions financières se font en ligne. Lorsque vous arrivez à destination, Uber préleve directement le prix de la course sur votre carte. Vous ne devez à aucun moment payer en espèces, laisser un pourboire ou marchander le prix. Le voyage n'en est que plus agréable puisque l'absence de relation d'argent génère une plus grande confiance entre le conducteur du véhicule et vous[1].

- **Inscription à Uber**

Le processus de création d'un compte Uber est particulièrement simple. Il vous suffit d'introduire les informations suivantes : une adresse mail valide, un numéro de téléphone, un numéro de carte bancaire ou de compte PayPal et enfin, un mot de passe. L'inscription peut s'effectuer depuis les applications ou via le site Uber.com. Il existe 2 moyens d'utiliser Uber pour faire appel à un chauffeur privé : via les applications mobiles iPhone et Android ou via le site mobile d'Uber, accessible depuis n'importe quel smartphone ou d'une tablette doté d'un navigateur web. Autre information importante : pour que le service fonctionne correctement, vous devez autoriser Uber à accéder à vos données de géolocalisation [1].

La figure 3.1 illustre l'étape d'inscription à Uber.

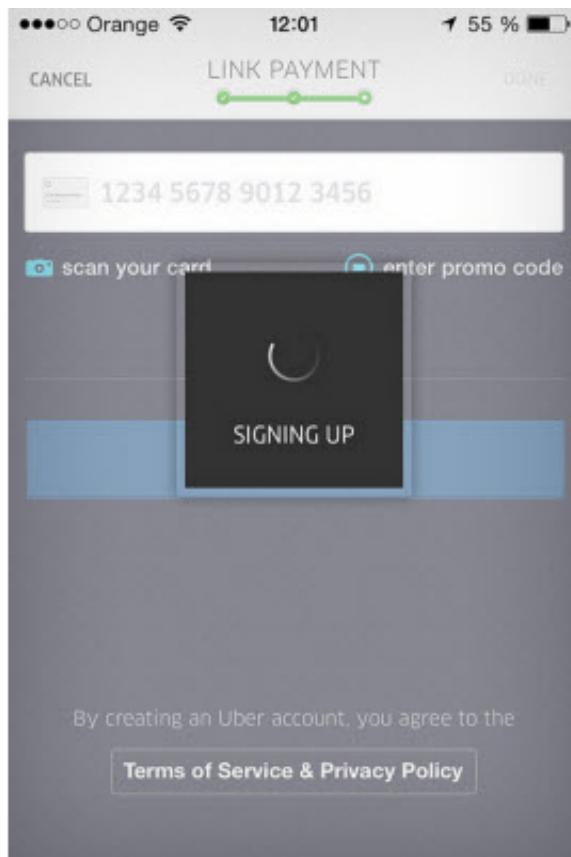


FIGURE 3.1 – Etape d'inscription à Uber [1]

- **Etape de réservation de taxi**

Uber détecte automatiquement votre position à l'aide de la géolocalisation. Pour appeler un taxi, indiquez votre point de récupération sur la carte puis validez comme le montre la figure 3.2. Uber recherche alors instantanément le chauffeur le plus proche de vous et vous indique à combien de temps ce dernier se trouve.

Les collaborateurs Uber présentent tous une fiche descriptive avec nom, photo, plaque immatriculation, véhicule utilisé et avis des consommateurs. Si le chauffeur accepte la course, vous avez accès à ces informations et un SMS vous est envoyé. Vous avez également la possibilité d'appeler le chauffeur en cas de besoin. Toujours grâce à la localisation, il vous est possible de suivre l'approche du véhicule sur la carte en temps réel. Enfin, après la course, vous pouvez laisser un avis via l'application et noter le chauffeur. Notez que les conducteurs notent eux-aussi les clients [1].

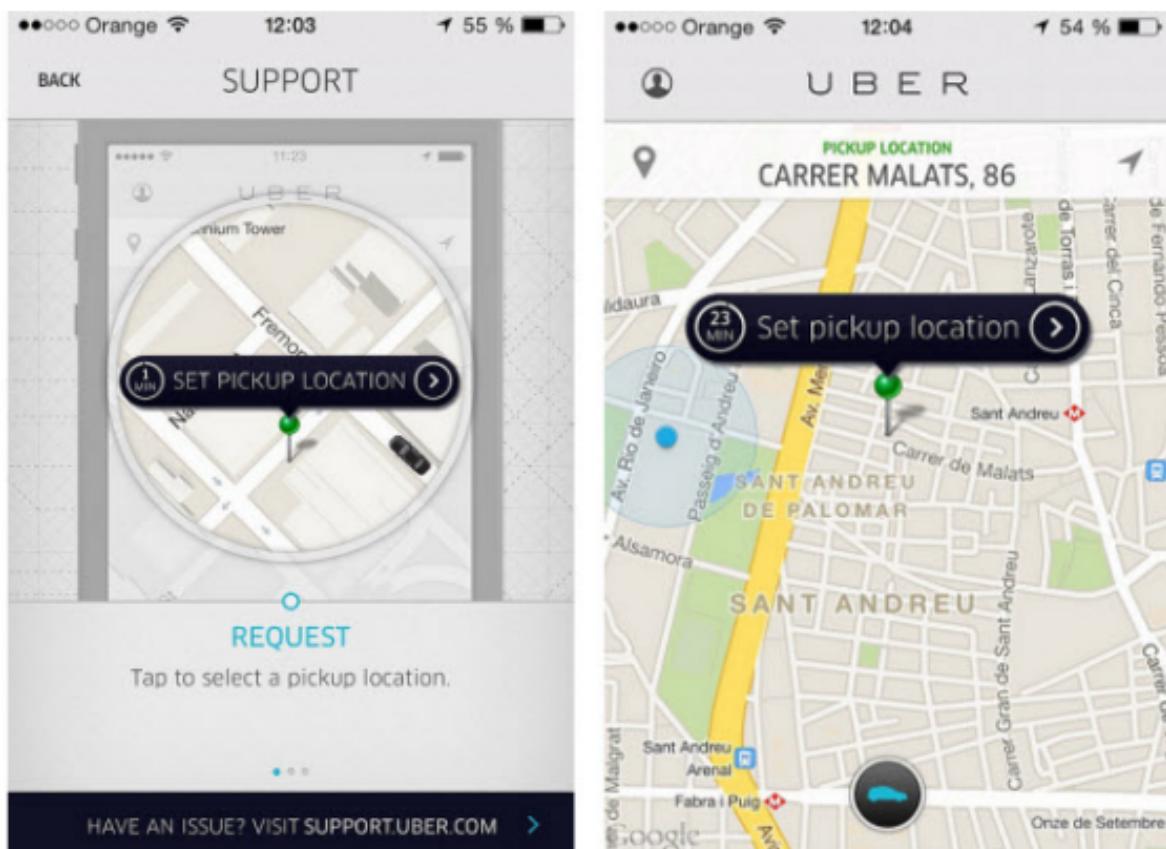


FIGURE 3.2 – Indication du point de récupération sur la carte[1].

3.2.2 Yassir

Yassir est une application qui fonctionne seulement à Alger, cette dernière sert à mettre en relation les chauffeurs de taxis et les personnes qui ont besoin de se déplacer par le biais de leur smartphone [2].

• Le principe de Yassir

Le principe de l'application est simple. Lorsque vous souhaitez vous déplacer et que vous avez besoin de prendre un taxi, l'application Yassir vous indique le nombre et lieux des chauffeurs qui se trouvent autour de vous et vous n'avez plus qu'à attendre que l'un d'entre eux réponde à votre demande.

Pour s'inscrire, l'utilisateur doit indiquer son prénom, son nom et son numéro de téléphone. Quand l'inscription est validée, et que l'utilisateur souhaite faire une réservation de taxi il n'aura qu'à suivre les étapes suivantes :

- L'utilisateur doit indiquer l'endroit où il se trouve exactement et où il souhaite se rendre comme le montre la figure 3.3.
- Il doit estimer le prix avant de demander un chauffeur.

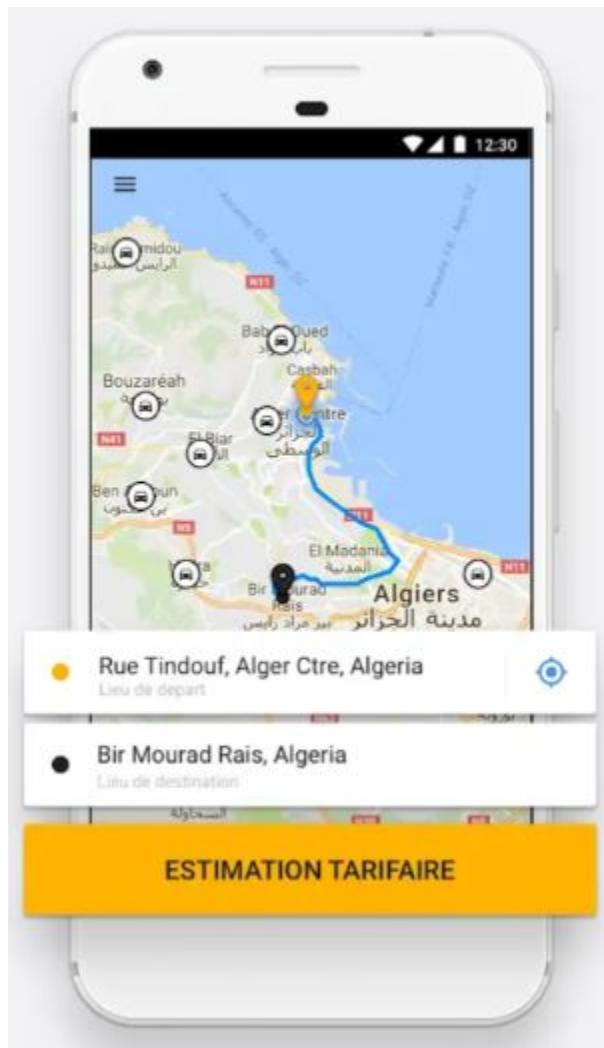


FIGURE 3.3 – Indication de la source et de la destination [2].

- Dès que la course est validée, le client reçoit alors une estimation du temps d'attente, la marque et la plaque d'immatriculation de la voiture qui viendra le chercher et le prénom du chauffeur comme le montre la figure 3.4.
- Une fois la course acceptée, vos informations s'affichent automatiquement chez le chauffeur qui vous appelle automatiquement sur votre téléphone portable pour confirmer la demande.
- Une fois la course terminée, le client peut la noter.

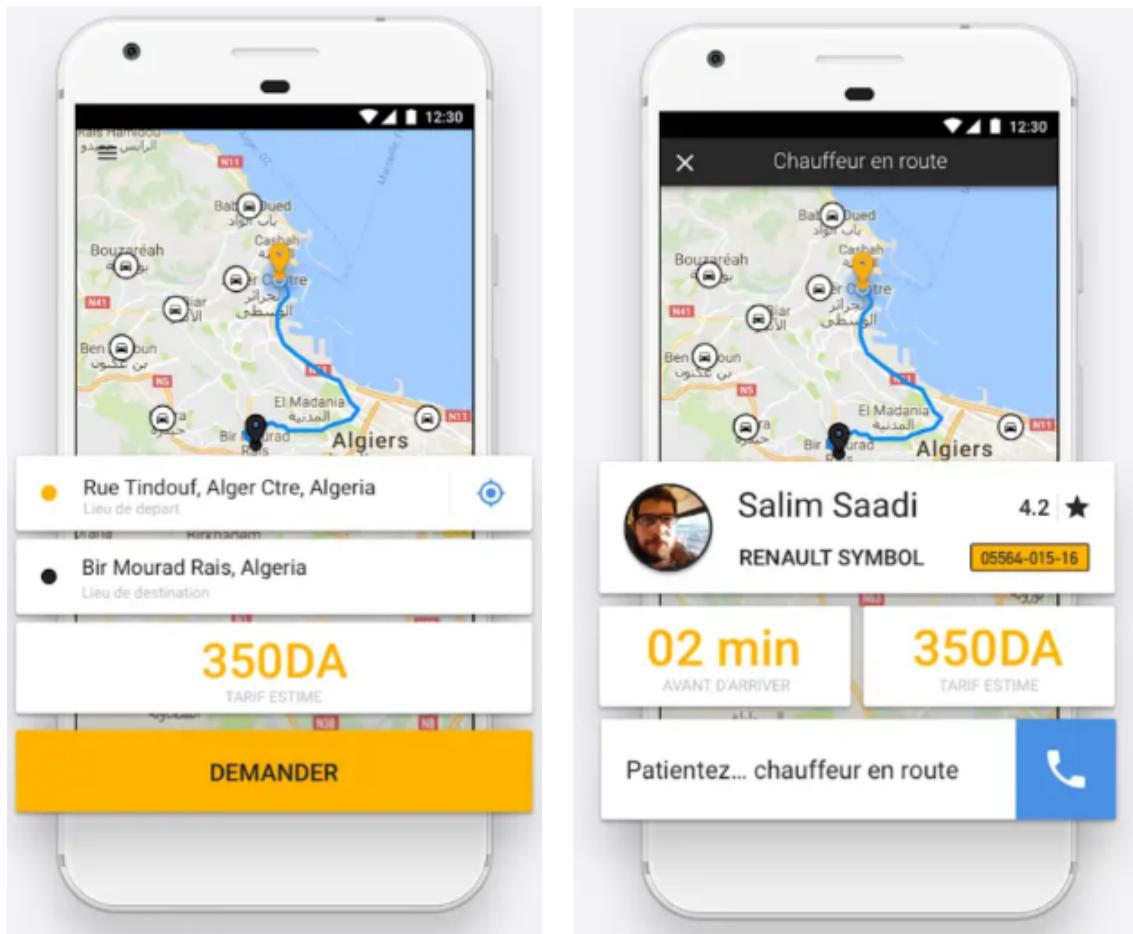


FIGURE 3.4 – Estimation du prix/ Chauffeur en route [2].

3.2.3 ITaxi

ITaxi a démarré le 21/09/2014, uniquement à Casablanca, et compte, moins d'un an après, 170 chauffeurs. Contrairement à ce qui a été dit, cela n'est pas le « Uber Marocain », mais le « Taxi G7 » marocain à la sauce Web 2.0 marocaine (Les taxis G7 sont une grosse centrale parisienne, très critiquée).

En effet, une première différence essentielle avec Uber : les voitures sont des petits taxis, en toute légalité. Le seul recours aux transporteurs touristiques concerne les transferts aéroports, que les petits taxis ne sont pas autorisés à faire. Les taxis sélectionnés sont tous des voitures récentes, de moins de cinq ans. La seconde différence, très importante, c'est qu'en tant que centrale de réservation, ITaxi offre plusieurs

modes de commande de taxi, y compris le téléphone, et permet de réserver un taxi à l'avance [3].

La figure 3.5 montre un visuel de l'application côté client.

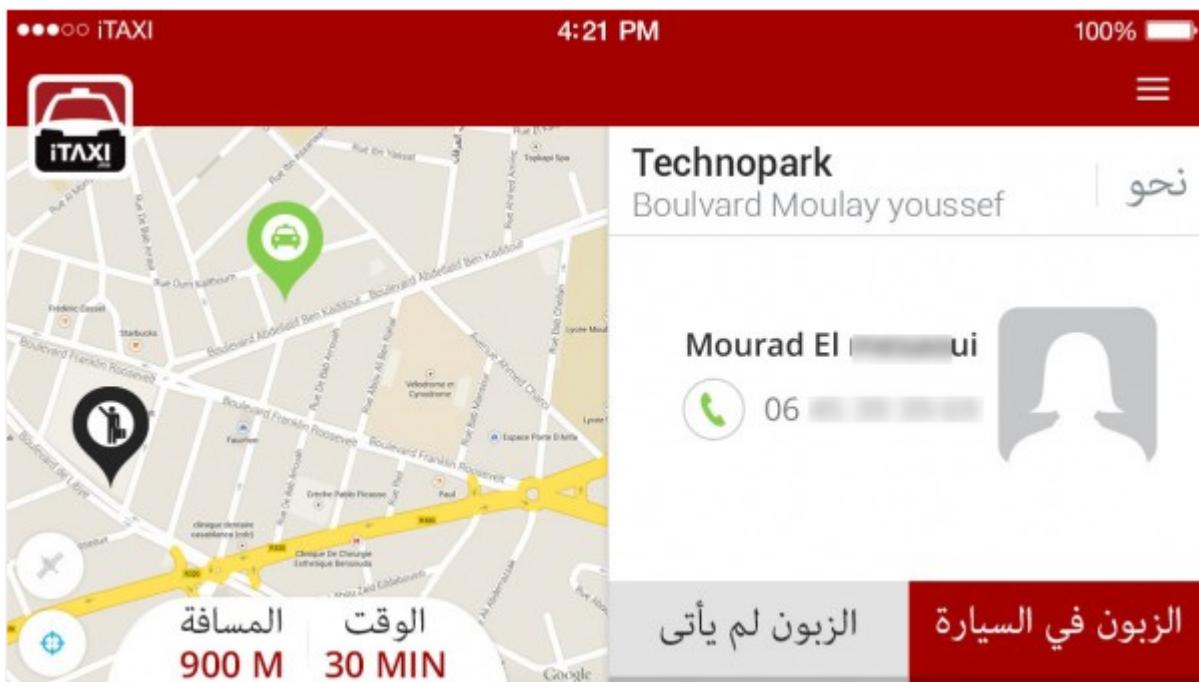


FIGURE 3.5 – Interface principal de l'application[3].

La figure 3.6 montre la dernière étape qui est la fin de la course où le client a la possibilité de la noter.

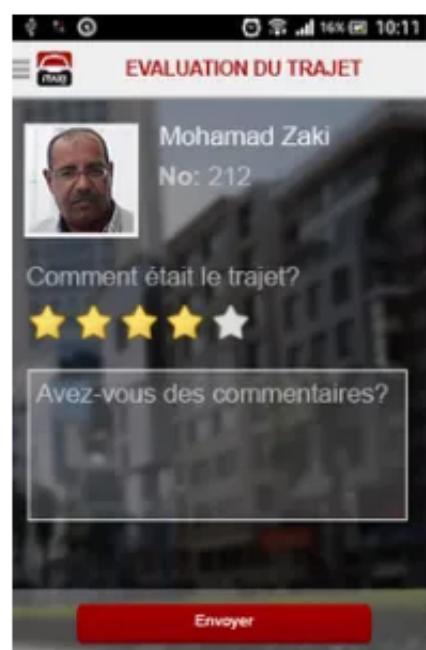


FIGURE 3.6 – Noter la course [3].

3.2.4 CarDispo

CarDispo est une société de services qui sert à faciliter la mise en relation des Usagers avec les Taxi grâce aux technologies numériques (Internet, Applications Mobiles, Objets connectés). CarDispo est présenté comme étant le premier réseau de taxis disponible au Cameroun et en Afrique Centrale. c'est une application qui permet à ses utilisateurs de réserver les taxis géo localisés à partir de leurs téléphones mobiles ou d'Internet. Concrètement, il suffit juste, pour un client de télécharger via Google Play Store ou via le site www.cardispo.com l'application CarDispo et il faut par la suite suivre le processus de réservation. Et même si le client ne sait pas comment s'y prendre avec Internet, ou il n'a pas un smartphone, il peut réserver un taxi en contactant le service client [4].

- **Le processus de réservation**

1. **Lancer l'application :** Rendez-vous sur le site internet ou installer l'application sur le téléphone. La figure 3.7 montre l'interface de réservation de taxi.

FIGURE 3.7 – Commander un Taxi avec CarDispo[4].

2. **Introduire quelques informations :**

- la position actuelle.
- la destination.
- le type de déplacement (dépôt 2 ou course 3).
- L'heure de départ (elle doit être supérieure d'au moins 30 minutes à l'heure courante).

- Les informations personnelles : nom (s), prénom (s), e-mail (éventuellement) et numéro de téléphone.

3. Cliquez sur le bouton «Validez la réservation».

La figure 3.8 montre l'étape de la saisie des informations.

FIGURE 3.8 – Saisir les informations dans CarDispo [4].

Dès l'acceptation de la demande, le client reçoit un sms et/ou un email contenant les informations de sa commande : un code unique, nom et photo du chauffeur, son numéro de carte grise.

3.3 Tableau comparatif des solutions existantes

Le tableau suivant représente une comparaison entre les solutions existantes

Critères Solutions	Le rayon d'utilisation	Type de réservation	Avantages	Inconvénients
Uber	253 villes	En temps réel	<ul style="list-style-type: none"> Les courses sont assurées en quelques minutes. Courses économiques. Paiement en ligne. Application compatible avec le système d'exploitation iOS et Android. Recrute ses chauffeurs sur la base de plusieurs critères. 	<ul style="list-style-type: none"> La destination du client n'est pas indiquée par ce dernier. la durée et la distance du trajet sont inconnues.
Yassir	Alger	En temps réel	<ul style="list-style-type: none"> Les taxieurs sont fiables car il existe une administration derrière qui s'assure de l'identité du chauffeur. L'estimation du prix se fait avant l'envoi de la demande de réservation. 	<ul style="list-style-type: none"> La source n'est pas localisée automatiquement lors de l'envoi de la demande de réservation. Le besoin de «vérifier» par téléphone montre que les utilisateurs, qu'ils soient usagers ou chauffeurs, ont encore du mal à se fier totalement à l'application.
Itaxi	Casablanca	Réservation à l'avance ou en temps réel.	<ul style="list-style-type: none"> Permet la réservation de taxi à l'avance. Une tarification raisonnable. Application compatible avec le système d'exploitation iOS et Android. Plusieurs modes de commande de taxi, y compris le téléphone. 	<ul style="list-style-type: none"> Service très lent. le taxi récupère d'autres passagers en chemin sans le demander.
CarDispo	Douala et Yaoundé	Réservation à l'avance de 30minutes.	<ul style="list-style-type: none"> Disponibilité 24h/24 et 7j/7. La participation au développement des villes africaines. Paiement en ligne. Deux types de taxi sont disponibles : Taxi simple et VIP. 	<ul style="list-style-type: none"> Le client ne peut pas suivre son taxi sur la map. Réservation lente. Le délai d'attente est très grand.

TABLE 3.1 – Tableau comparatif des solutions existantes.

3.4 Problématique

Nous sommes probablement à l'aube d'une évolution radicale du rôle des taxis dans la mobilité urbaine avec d'une part des besoins croissants pour des services personnalisés que les taxis sont les plus aptes à offrir, et d'autre part, la perspective de voir se desserrer trois freins à leur développement : la disponibilité, le temps nécessaire pour trouver un taxi ainsi que le coût élevé.

Pour palier à ces contraintes l'une des solutions est d'utiliser les applications mobiles qui sont actuellement en pleine croissance grâce aux succès des smartphone entre autre ceux qui tournent sous Android, cette tendance offre aux transports publics de formidables opportunités pour créer de nouveaux services ou d'en améliorer les services existants.

Dans cette perspective, les clients envoient une demande de réservation à un taxieur comprenant sa source, sa destination, la durée et la distance ce qui fait que le problème de disponibilité et de temps sera résolu. Le taxieur indiquera le coût de la course et donnera la possibilité au client de refuser ce qui créera une concurrence entre taxieurs et permettra de réduire le problème du coût élevé.

3.5 Présentation du sujet

Le mémoire s'intitule « Conception et mise en œuvre d'une application mobile sous Android de géolocalisation dynamique et de réservation de taxis ». Notre application nommée « Taxi06 » est destinée, à la fois, aux clients et aux chauffeurs de taxis possédant un smartphone sous Android. En résumé, l'application côté client permet la réservation de taxis en temps réel après une géolocalisation sur la carte. L'application côté chauffeur permet de recevoir les demandes de réservation et d'afficher la position du client ainsi que sa destination, le chauffeur de taxi indique le prix au client dans le cas où il accepte la demande, et précisera s'il est libre dans l'immediat ou non en indiquant quand il le sera, et dans le cas où il décide de ne pas accepter la demande il l'annule. A son tour le client aura la possibilité d'accepter ou de refuser le prix indiqué par le taxieur si ce dernier ne lui convient pas.

3.6 Solution retenue

Cette étude de l'existant, met en évidence l'intérêt de développer une application informatique pour apporter une solution à la problématique et d'appliquer ainsi le principe de géolocalisation dynamique. Pour cela nous avons pris l'initiative d'améliorer quelques inconvénients cités précédemment et de rajouter des fonctionnalités à notre application qui sont citées parmi les principaux objectifs de cette dernière et qui sont les suivants :

- Réservation d'un taxi de manière simple et rapide en temps réel en ayant obligatoirement un smartphone Android.
- Application collaborative entre le client et le taxieur.
- C'est le client et le taxieur qui permettent à l'application de s'autogérer.

- L'application est destinée à l'intra-urbain, mais peut aussi être destinée à l'inter-urbain si le taxieur accepte les longues distances.
- L'utilisateur peut s'inscrire par téléphone, Facebook ou bien en utilisant son compte Google.
- Utilisation de la géolocalisation pour voir les taxis à un rayon de 5Km en temps réel.
- La position du client est localisée automatiquement.
- Le client peut indiquer un lieu de rendez-vous s'il le souhaite.
- Après avoir reçu une demande de réservation et dans le cas où le taxieur accepte la demande, Il indique le prix qu'il trouve adéquat, à son tour le client pourra soit accepter ou refuser.
- La demande de réservation expire automatiquement au bout de trois minutes s'il n'y a pas de réponse.
- Après avoir envoyé une demande de réservation, le client ne peut pas faire une autre demande.
- Après que la course soit confirmée, tous les taxis seront supprimés de la map du client hormis celui qu'il a réservé.
- Après avoir accepté une demande de réservation, le taxieur ne peut plus l'annuler.
- Le taxieur devient indisponible pour les autres clients, après une course confirmée.
- Le client peut suivre la position du taxi qu'il a réservé en temps réel.
- Le client sera notifié lorsque le taxieur arrive.
- Le client peut consulter un guide d'utilisation "Comment ça marche" qui est utile lors de la première utilisation de l'application.
- Accéder à l'historique des courses.
- Possibilité au client de noter le taxieur.
- Possibilité au taxieur de noter le client.
- Un utilisateur de l'application sera bloqué si la moyenne des notes qu'on lui a attribué est inférieure à 3.
- Un utilisateur bloqué sera sanctionné d'une durée de 6mois.

3.7 Conclusion

Dans ce chapitre nous avons fait une présentation générale du sujet ainsi que les objectifs à atteindre. L'objectif principal de ce mémoire est de créer un service réservation de taxi efficace et en temps réel, qui répond aux besoins des clients et qui va leur faciliter la vie en leur offrant un moyen très rapide à joindre et très opérant, sachant qu'ils pourront l'utiliser à n'importe quel moment à condition d'avoir un smartphone Android et une connexion internet.

Dans le chapitre suivant, nous allons survoler le domaine des applications mobiles ainsi que les systèmes d'exploitation relatifs à ces applications.

Chapitre 4

Analyse et conception

4.1 Introduction

Ce chapitre aborde une partie très importante de notre mémoire qui est la conception de l'application. En premier lieu, nous allons présenter le processus de développement utilisé qui est indispensable à n'importe quel type de projet informatique, et pour notre application, nous avons choisis le processus unifié (UP), ce dernier est généralement utilisé pour les grands projets, mais sa simplicité fait qu'il est aussi adapté aux petits projets comme le nôtre, il est caractérisé par le fait qu'il est accompagné par les différents diagrammes UML et piloté par les cas d'utilisation.

En second lieu, nous allons présenter les diagrammes UML et s'accentuer sur ceux que nous avons utilisés. Nous parlerons ensuite de l'analyse de notre système en utilisant les diagrammes associés à cette étape.

En dernier lieu, nous allons parler de la conception de notre système en utilisant les différents diagrammes que l'on va présenter en détails.

4.2 Processus Unifié

4.2.1 Définition

Le processus unifié en anglais Unified Process (UP) est un processus de développement logiciel itératif, centré sur l'architecture, piloté par des cas d'utilisation et orienté vers la diminution des risques. C'est un patron de processus pouvant être adapté à une large classe de systèmes logiciels, à différents domaines d'application, à différents types d'entreprises, à différents niveaux de compétences et à différentes tailles de l'entreprise [17].

4.2.2 Les caractéristiques de UP

Le processus unifié englobe plusieurs caractéristiques parmi elles, on peut en distinguer 3 principales caractéristiques qui sont les suivantes :

- **Itératif**

L'itération est une répétition d'une séquence d'instructions ou d'une partie de programme un nombre de fois fixé à l'avance ou tant qu'une condition définie n'est pas remplie, dans le but de reprendre un traitement sur des données différentes. Elle qualifie un traitement ou une procédure qui exécute un groupe d'opérations de façon répétitive jusqu'à ce qu'une condition bien définie soit remplie [17].

- **Piloté par les cas d'utilisation d'UML**

Le but principal d'un système informatique est de satisfaire les besoins du client. Le processus de développement sera donc accès sur l'utilisateur. Les cas d'utilisation permettent d'illustrer ces besoins. Ils détectent puis décrivent les besoins fonctionnels (du point de vue de l'utilisateur), et leur ensemble constitue le modèle de cas d'utilisation qui dicte les fonctionnalités complètes du système [17].

- **Centré sur une architecture**

Le processus unifié (UP) gère le processus de développement par deux axes.

- **L'axe vertical** : représente les principaux enchaînements d'activités, qui regroupent les activités selon leur nature. Cette dimension rend compte l'aspect statique du processus qui s'exprime en termes de composants, de processus, d'activités, d'enchaînements, d'artefacts et de travailleurs.
- **L'axe horizontal** : représente le temps et montre le déroulement du cycle de vie du processus . cette dimension rend compte de l'aspect dynamique du processus qui s'exprime en terme de cycles, de phases, d'itérations et de jalons.

La figure 4.1 représente l'architecture du processus unifié.

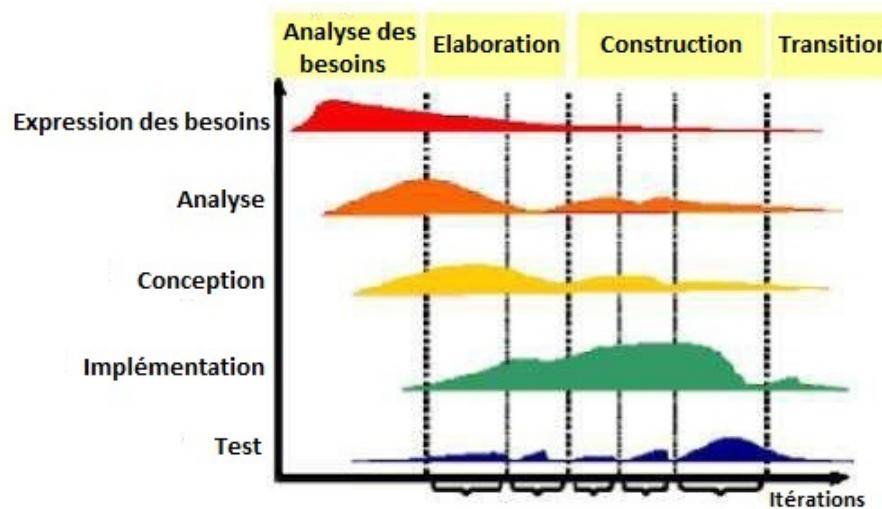


FIGURE 4.1 – L'architecture du processus unifié [17].

1. L'axe vertical

Dans l'axe vertical de l'architecture d'UP, on peut distinguer 5 activités qui sont les suivantes [17].

• Expression des besoins

L'expression des besoins comme son nom l'indique, permet de définir les différents besoins :

- Inventorier les besoins principaux et fournir une liste de leurs fonctions.
- Recenser les besoins fonctionnels (du point de vue de l'utilisateur) qui conduisent à l'élaboration des modèles de cas d'utilisation.
- Appréhender les besoins non fonctionnels (technique) et livrer une liste des exigences.

• Analyse

L'objectif de l'analyse est d'accéder à une compréhension des besoins et des exigences du client. Il s'agit de livrer des spécifications pour permettre de choisir la conception de la solution. Un modèle d'analyse livre une spécification complète des besoins issus des cas d'utilisation et les structure sous une forme qui facilite la compréhension (scénarios), la préparation (définition de l'architecture), la modification et la maintenance du futur système.

• Conception

La conception permet d'acquérir une compréhension approfondie des contraintes liées au langage de programmation, à l'utilisation des composants et au système d'exploitation. Elle détermine les principales interfaces et les transcrit à l'aide d'une notation commune.

• Implémentation

L'implémentation est le résultat de la conception pour implémenter le système sous forme de composants, c'est-à-dire, de code source, de scripts, de binaires, d'exécutables et d'autres éléments du même type.

- **Test**

Les tests permettent de vérifier des résultats de l'implémentation en testant la construction. Pour mener à bien ces tests, il faut les planifier pour chaque itération, les implémenter en créant des cas de tests, effectuer ces tests et prendre en compte le résultat de chacun.

2. L'axe horizontale

Dans l'axe horizontal on peut distinguer 4 phases qui sont les suivantes [17] :

- **Analyse des besoins**

L'analyse des besoins donne une vue du projet sous forme de produit fini. Cette phase porte essentiellement sur les besoins principaux (du point de vue de l'utilisateur), l'architecture générale du système, les risques majeurs, les délais et les coûts

- **Elaboration**

L'élaboration reprend les éléments de la phase d'analyse des besoins et les précise pour arriver à une spécification détaillée de la solution à mettre en œuvre. L'élaboration permet de préciser la plupart des cas d'utilisation, de concevoir l'architecture du système et surtout de déterminer l'architecture de référence. Au terme de cette phase, les chefs de projet doivent être en mesure de prévoir les activités et d'estimer les ressources nécessaires à l'achèvement du projet.

- **Construction**

La construction est le moment où l'on construit le produit. L'architecture de référence se métamorphose en produit complet. Le produit contient tous les cas d'utilisation que les chefs de projet, en accord avec les utilisateurs ont décidé de mettre au point pour cette version.

- **Transition**

Le produit est en version bêta. Un groupe d'utilisateurs essaye le produit et détecte les anomalies et défauts. Cette phase suppose des activités comme la formation des utilisateurs clients, la mise en œuvre d'un service d'assistance et la correction des anomalies constatées.

4.3 Langage de modélisation unifié (UML)

4.3.1 Définition

Le langage de modélisation unifié (UML) se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. En effet UML est un langage avec une syntaxe et des règles bien définies qui tentent à réaliser des buts d'écrit grâce à

une représentation graphique formée de diagrammes et une modélisation textuelles qui vient enrichir la représentation graphique [18].

Le langage de modélisation unifié a un certain nombre d'avantages, nous citons quelques-uns :

- Universel.
- Adopté par les grandes entreprises.
- Utilisé à l'international UML contrairement à Merise qui est français.
- Notation unifiée.
- Facile à comprendre.
- Adopté par plusieurs processus de développement.
- Limite les risques d'erreurs.
- N'est pas limité au domaine informatique.

4.3.2 Les différents diagrammes UML

Le langage de modélisation unifié (UML) s'articule autour de treize types de diagrammes, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système logiciel. Ces types de diagrammes sont répartis en deux grands groupes représentés[18].

la figure 4.2 illustre ces différents diagrammes.

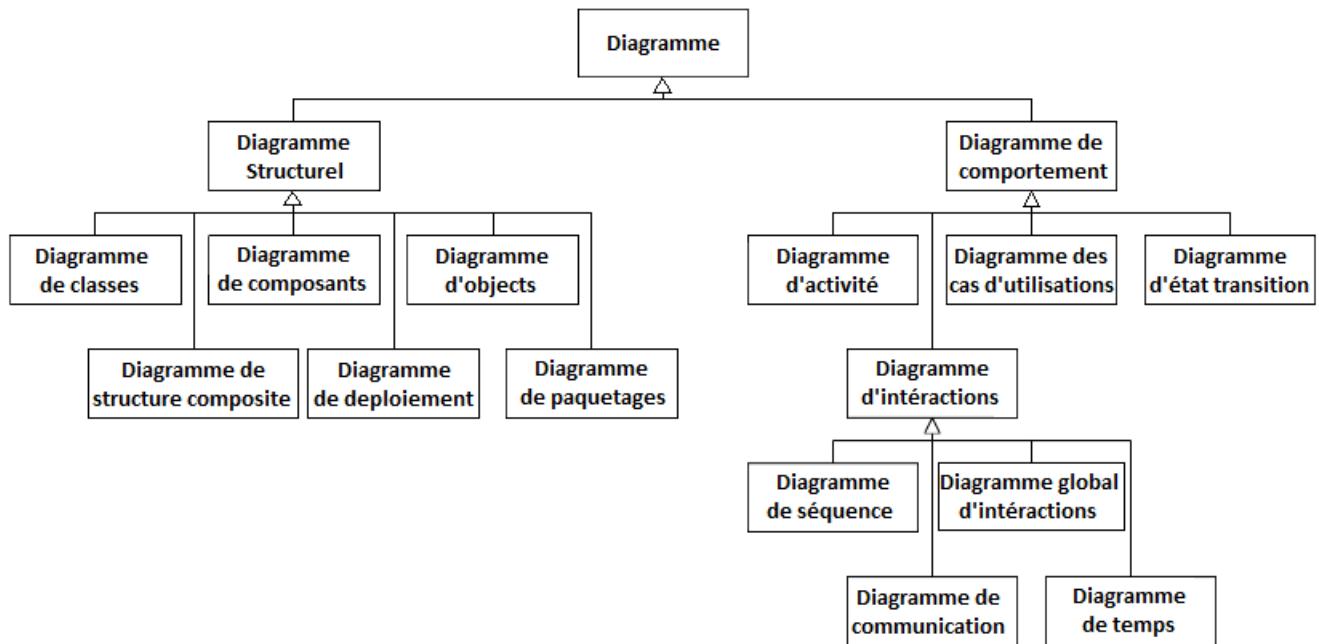


FIGURE 4.2 – Les diagrammes UML[18].

Les diagrammes que nous allons présenter sont les diagrammes que nous avons jugé nécessaire à utiliser dans notre conception.

- **Diagramme de cas d'utilisation**

Ce diagramme est destiné à représenter les besoins des utilisateurs par rapport au système. Il constitue un des diagrammes les plus structurants dans l'analyse d'un système [17].

- **Diagramme de séquence**

C'est la représentation des interactions entre objets en indiquant la chronologie des échanges. Cette représentation peut se réaliser par cas d'utilisation en considérant les différents scénarios associés, en d'autres termes le diagramme de séquence va nous permettre de décrire les scénarios des cas d'utilisation [17].

- **Diagramme d'activités**

Le diagramme d'activité est un diagramme comportemental d'UML, permettant de représenter le déclenchement d'événements en fonction des états du système, il présente un certain nombre de points communs avec le diagramme d'état-transition puisqu'il concerne le comportement interne des opérations ou des cas d'utilisation [17].

- **Diagramme de classes**

Le diagramme de classe constitue l'un des pivots essentiels de la modélisation avec UML. En effet, ce diagramme permet de donner la représentation statique du système à développer. Cette représentation est centrée sur les concepts de classe et d'association. Chaque classe se décrit par les données et les traitements dont elle est responsable pour elle-même et vis-à-vis des autres classes. Les traitements sont matérialisés par des opérations. Le détail des traitements n'est pas représenté directement dans le diagramme de classe [17].

- **Diagramme de déploiement**

Le diagramme de déploiement est une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont répartis ainsi que les relations entre eux. [17].

4.4 Spécification des besoins

4.4.1 Besoins fonctionnels

Dans cette partie nous détaillons les fonctionnalités, que le système doit fournir aux différents acteurs, qui se présentent comme suit

- Localisation de l'utilisateur.
- Réservation d'un taxi en temps réel.
- Traçage de l'itinéraire.

- Estimation du temps et de la distance.
- Gestion des demandes de réservation.
- Suivre l'approche du taxieur en temps réel.
- Notifier le client lorsque le taxieur arrive.
- Consultation de l'historique.
- Inscription .
- Noter l'utilisateur.

4.4.2 Besoins non fonctionnels

Il s'agit des besoins qui caractérisent le système. Ce sont des besoins en matière de performance, de type de matériel ou de type de conception. Pour cela notre futur système doit répondre aux caractéristiques suivantes :

- Ergonomie.
- Rapidité de traitement.
- Facilité d'utilisation.
- Sécurité.

4.5 Analyse des besoins

La première étape de la conception consiste à analyser la situation pour tenir compte des contraintes, des risques et de tout autre élément pertinent et assurer un ouvrage ou un processus répondant aux besoins du client [17].

4.5.1 Les acteurs

Un acteur est un utilisateur type qui a toujours le même comportement vis-à-vis d'un cas d'utilisation. Ainsi les utilisateurs d'un système appartiennent à une ou plusieurs classes d'acteurs selon les rôles qu'ils tiennent par rapport au système. Une même personne physique peut se comporter en autant d'acteurs différents que le nombre de rôles qu'elle joue vis-à-vis du système [17].

Les acteurs que comporte notre système sont :

- Le client
- Le taxieur

4.5.2 Diagramme de cas d'utilisation

- **Identification des acteurs**

Un cas d'utilisation représente une fonctionnalité du système. Cette fonctionnalité est définie par une action déclenchante, un ou plusieurs déroulements possibles et éventuellement une fin.

Le tableau suivant représente les différents cas d'utilisation associé à notre système

Fonction	Cas d'utilisation	Acteurs
Authentification à l'application	S'authentifier	Client, taxieur
Inscription avec Facebook	S'inscrire	Client, taxieur
Inscription avec téléphone		
Inscription avec Google		
Accepter une demande	Gérer les demandes de réservation	Taxieur
Refuser une demande		
Sélectionner un taxi	Réserver un taxi	Client
Indiquer la destination		
Confirmer		
Annuler		
Consultation de l'historique	Consulter son historique	Client, Taxieur
Le client signale un taxi	Signaler un taxi	Client
Le client note le taxi	Noter un taxi	Client
Le taxieur note le client	Noter un client	Taxieur
Disponibilité du taxieur	Disponibilité	Taxieur

TABLE 4.1 – Les différents cas d'utilisation

- **Réalisation du diagramme de cas d'utilisation**

La figure 4.3 représente le diagramme de cas d'utilisation global de notre système.

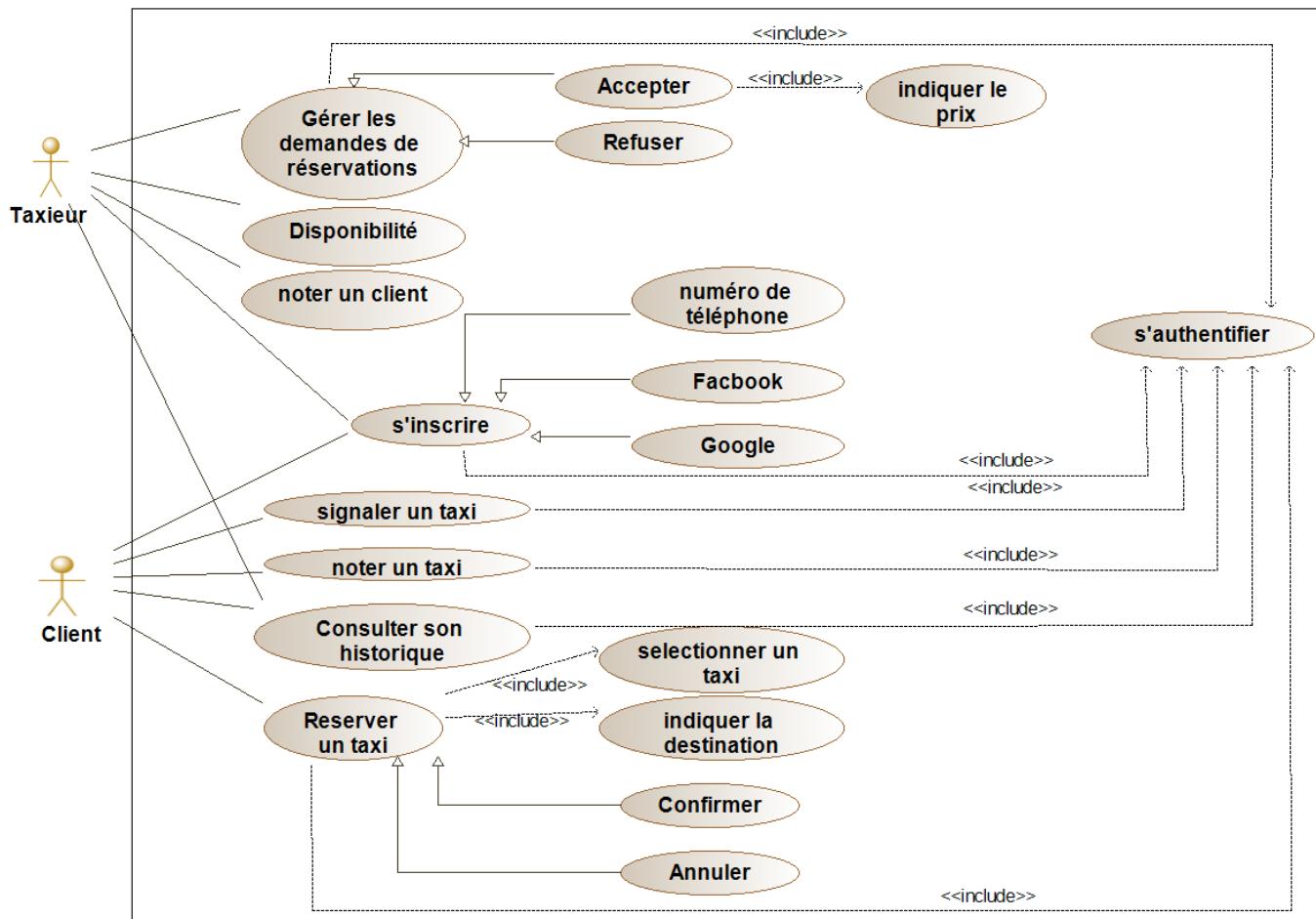


FIGURE 4.3 – Diagramme de cas d'utilisation global.

- **Description textuelles des cas d'utilisations**

Cette partie sera consacrée pour la description des cas d'utilisations, chaque description sera représenté sous forme de tableau qui indiquera le nom du cas d'utilisation, l'acteur responsable, l'objectif, ainsi que le scenario.

Cas d'utilisation « S'inscrire »

le tableau suivant représente une description textuelle du cas d'utilisation "S'inscrire"

Cas d'utilisation	S'inscrire
Acteur(s)	Client Taxieur
Objectif	Inscription à l'application L'utilisateur lance l'application. Lancement de l'animation de départ. Une interface s'affiche où il devra choisir entre client ou taxieur.
Scénario nominal	L'interface d'inscription lui sera affichée. Il choisit le mode d'inscription soit par Facebook, numéro de téléphone ou bien son compte Google. Il accède à l'application.
Alternatives	S'il choisit de s'inscrire avec son numéro de téléphone et qu'il valide sans remplir les champs téléphone et pseudo un message d'erreur lui sera affiché S'il saisit un numéro de téléphone inférieur à 10 chiffres, un message d'erreur lui sera affiché S'il saisit un pseudo inférieur à 4 caractères, un message d'erreur lui sera affiché Si le taxieur a été bloqué et qu'il essaie de se reinscrire, un message d'erreur lui sera affiché

TABLE 4.2 – Description du cas d'utilisation "S'inscrire".

Cas d'utilisation « Gérer les demandes »

le tableau suivant représente une description textuelle du cas d'utilisation "Gérer les demandes"

Cas d'utilisation	Gérer les demandes
Acteur(s)	Taxieur
Objectif	La gestion des demandes de réservation des clients
Scénario nominal	Le taxieur reçoit une demande de réservation du client. L'itinéraire lui sera affiché sur sa map Il indique le prix au client et clique sur accepter. Le client confirme la demande et notifie le taxieur. Le taxieur devient automatiquement indisponible et sera supprimé de la map des autres clients.
Alternatives	Si le taxieur n'accepte pas la demande de réservation il clique sur annuler.

TABLE 4.3 – Description du cas d'utilisation "Gérer les demandes".

Cas d'utilisation « Réserver un taxi »

le tableau suivant représente une description textuelle du cas d'utilisation "Réserver un taxi"

Cas d'utilisation	Réserver un Taxi
Acteur(s)	Client
Objectif	La réservation d'un taxi
Scénario nominal	Le client sélectionne la destination sur la map.
	Il sélectionne un taxi.
	Après qu'il l'ait confirmé la demande de réservation, il l'envoie au taxi sélectionné comprenant sa source, sa destination, le temps ainsi que la distance.
	Il accepte le prix indiqué par le taxieur après que ce dernier ait accepté la demande de réservation.
	Tous les autres taxieurs seront supprimés de sa map et ne visualisera que le taxi qu'il a réservé.
	Le taxieur recevra un message lui indiquant que la course a été confirmée.
Alternatives	Si le taxieur refuse la demande de réservation, le client recevra un message qui indique que le taxieur a refusé sa demande.
	Si le délai d'attente expire, le client recevra un message lui indiquant que le taxieur n'a pas répondu à sa demande dans les temps.

TABLE 4.4 – Description du cas d'utilisation "Réserver un Taxi".

4.6 Conception

Cette partie sera consacrée pour la conception de notre système, on utilisera le diagramme d'activité, les diagrammes de séquences pour représenter les scénarios, ainsi que le diagramme de classe pour représenter les entités manipulées par les utilisateurs, et enfin le diagramme de déploiement qui représente l'architecture de notre système.

4.6.1 Diagramme d'activités

La figure 4.4 représente le diagramme global d'activités du système.

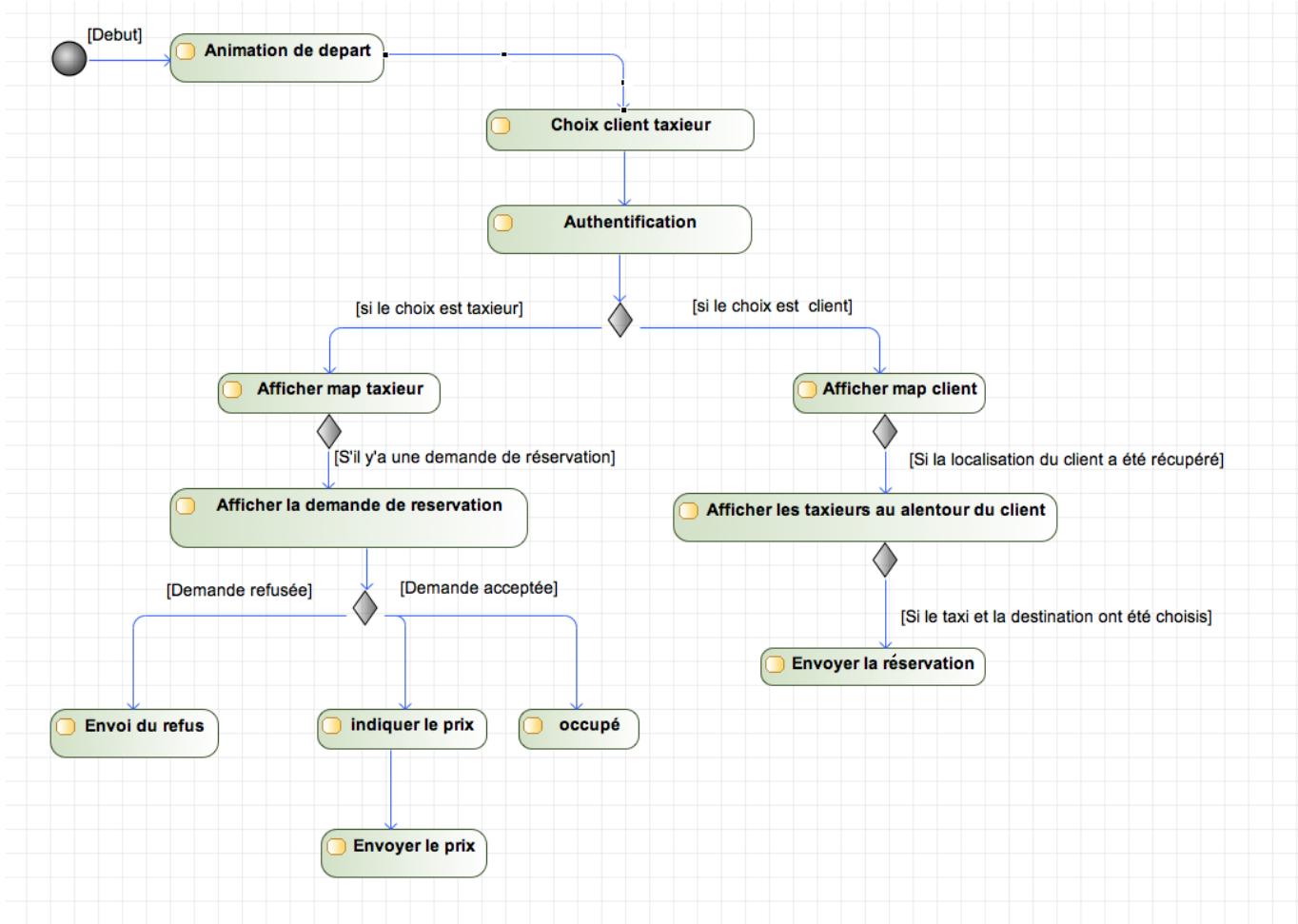


FIGURE 4.4 – Diagramme global d'activités.

4.6.2 Patrons de conception

Un design pattern ou pattern de conception consiste en un schéma à objets qui forme une solution à un problème connu et fréquent. Le schéma à objets est constitué par un ensemble d'objets décrits par des classes et des relations liant les objets. Les patterns répondent à des problèmes de conception de logiciels dans le cadre de la programmation par objets. Ce sont des solutions connues et éprouvées dont la conception provient de l'expérience de programmeurs [20]. Ainsi, dans le cadre de notre application, nous avons utilisés les designs patterns suivants :

- **Le pattern DAO**

Ce pattern permet de faire le lien entre la couche d'accès aux données et la couche métier d'une application. Il permet de mieux maîtriser les changements susceptibles d'être opérés sur le système de stockage des données . donc, par extension, de préparer une migration d'un système à un autre (BDD vers fichiers XML, par exemple...). Ceci se fait en séparant accès aux données (BDD) et objets métiers (POJO), ce modèle complète le modèle MVC qui préconise de séparer dans des classes les différentes problématiques [21].

- **Le pattern MVC**

Modèle-vue-contrôleur ou MVC est un motif d'architecture logicielle destiné aux interfaces graphiques, le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs [21].

- Un modèle (Model) contient les données à afficher.
- Une vue (View) contient la présentation de l'interface graphique.
- Un contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur.

Le schéma 4.5 représente le modèle vue contrôleur

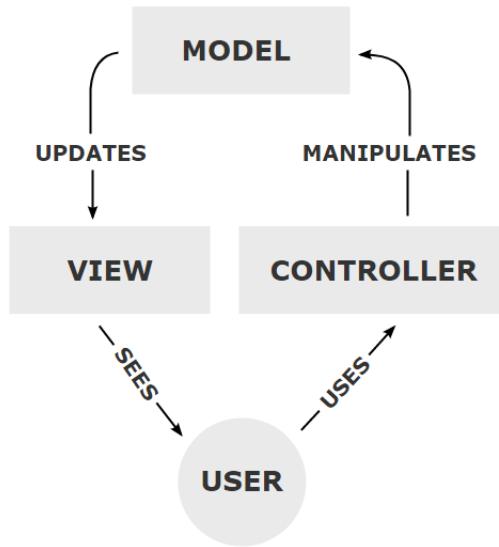


FIGURE 4.5 – Schéma du MVC [21].

4.6.3 Diagrammes de séquences

Nous allons présenter quelques diagrammes de séquences et expliquer le fragment d'interactions que nous avons utilisé dans ces diagrammes.

- **Fragments d'interactions**

Un fragment d'interaction correspond à un ensemble d'interactions auquel on applique un opérateur, il se représente globalement comme un diagramme de séquence dans un rectangle avec indication dans le coin à gauche du nom de l'opérateur [17].

L'opérateur que nous allons utiliser dans les diagrammes de séquences est l'opérateur « alt » qui correspond à une instruction de test avec une ou plusieurs alternatives possibles. Il permet aussi d'utiliser les classes de type sinon et se représente dans un fragment possédant au moins deux parties séparées par des pointillés.

- Réalisation des diagrammes de séquences

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

L'utilisateur qui représente le client ou bien le taxieur doit s'inscrire pour avoir le droit d'utiliser l'application, il peut soit s'inscrire par Facebook, par Google ou bien par numéro de téléphone, le diagramme ci-dessous représente le diagramme de séquence du cas où l'utilisateur s'inscrit par téléphone.

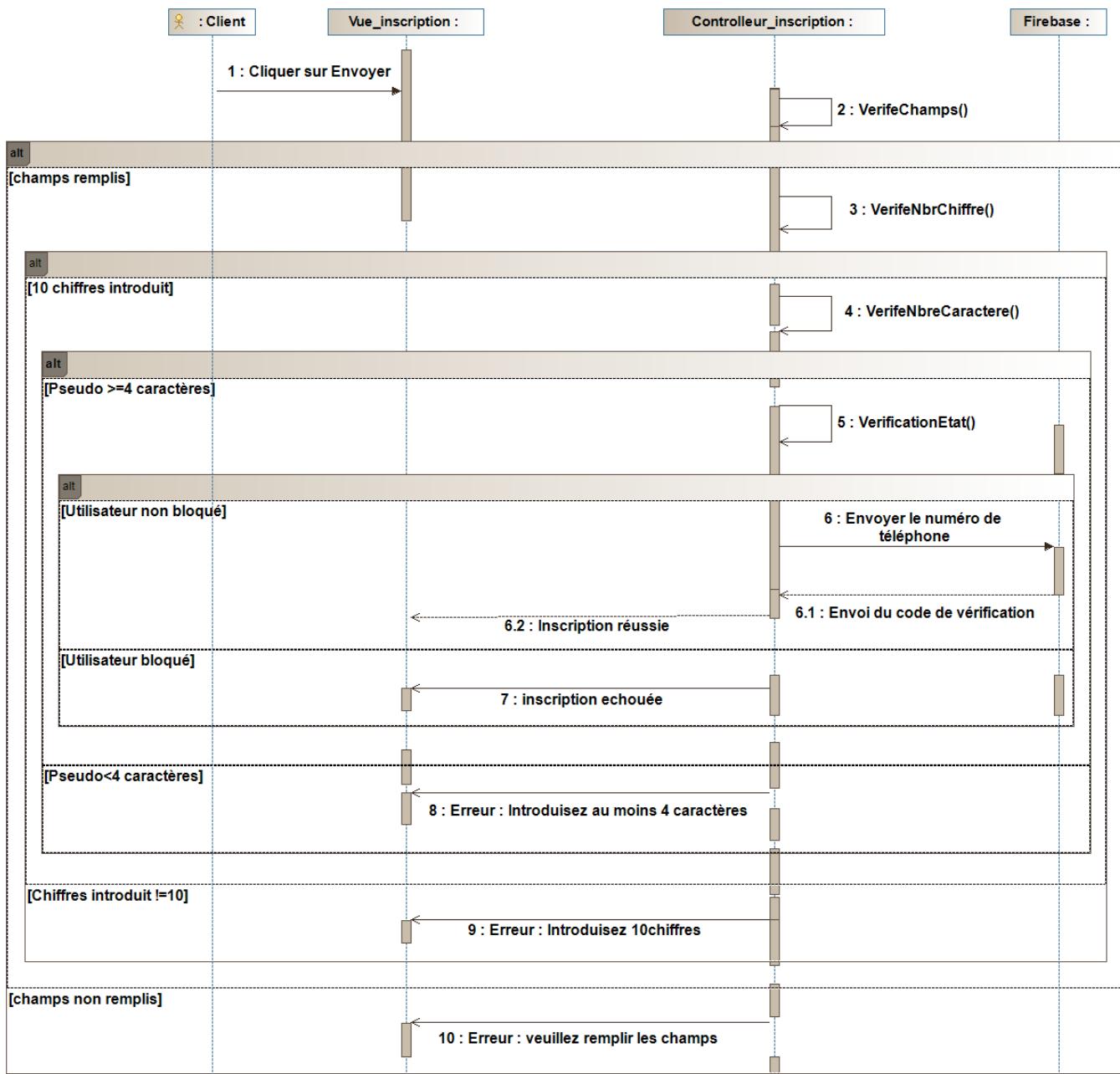


FIGURE 4.6 – Diagramme de séquence du cas d'utilisation "S'inscrire".

b. Diagramme de séquence du cas d'utilisation "Réserver un taxi"

Pour que le client puisse faire une réservation, il devra passer par le scenario indiquer dans la figure 4.7 qui représente le diagramme de séquence du cas d'utilisation « Réserver un taxi»

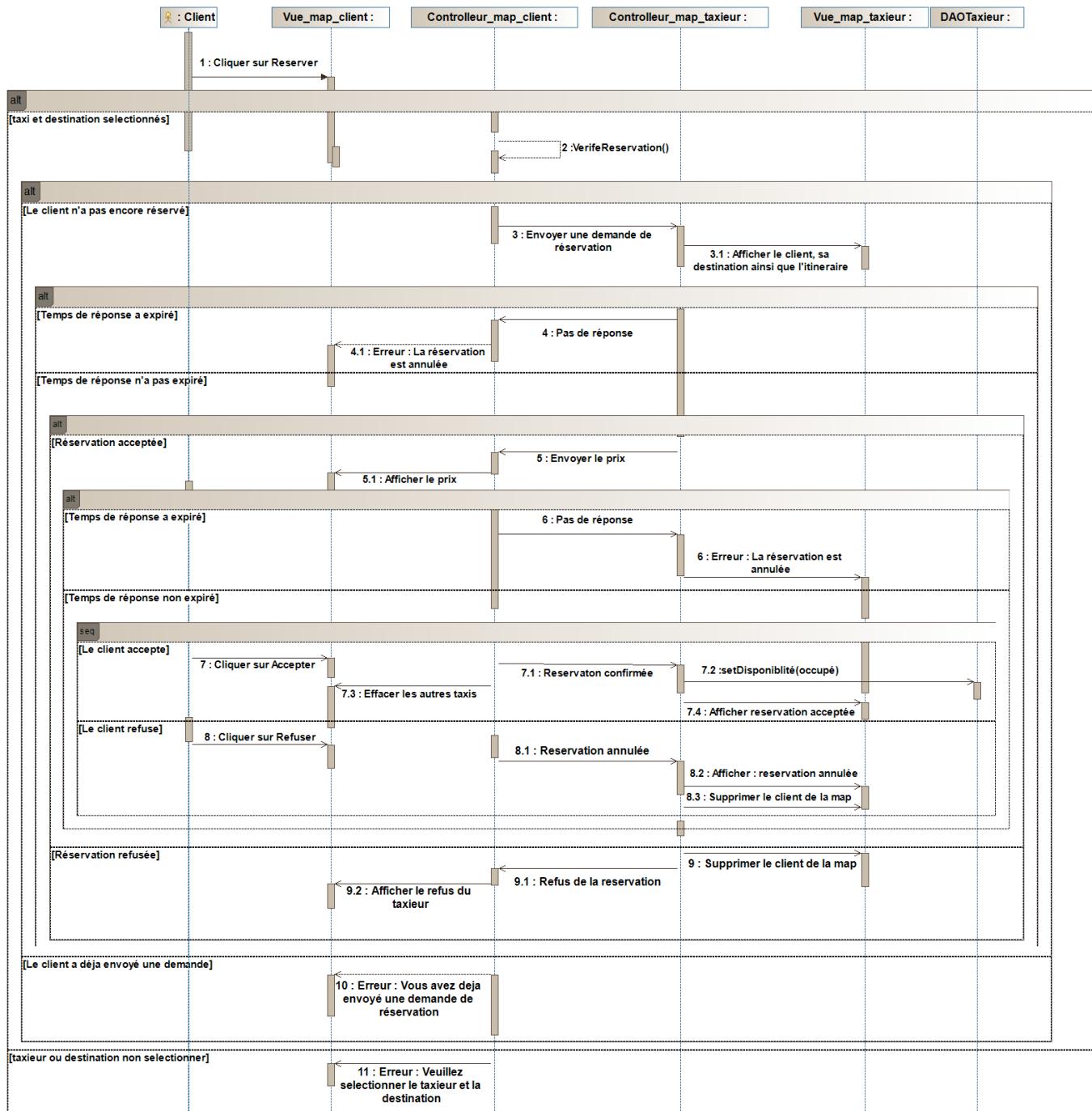


FIGURE 4.7 – Diagramme de sequence du cas d'utilisation "Réserver un taxi".

4.6.4 Diagramme de classes

- **Dictionnaire de données**

Le tableau ci-dessous représente le dictionnaire de donnée du diagramme de classe. Ces informations sont le fruit d'une collecte d'informations pertinentes jugées utiles, il est à signaler que ce dictionnaire a été épuré.

Classe	Attribut	Définition de l'attribut	Type
Client	Id-client	L'identifiant du client	Alphanumérique
	Pseudo-client	Pseudonyme du client	Alphanumérique
	Longitude-client	Position verticale du client	Numérique
	Latitude-client	Position horizontale du client	Numérique
	Num-tel	Le numéro du téléphone du client	Numérique(entier)
Taxieur	Id-Taxieur	L'identifiant du taxieur	Alphanumérique
	Connecter	Utilisateur connecté ou non	Booleen
	Réservé	Reservation du taxi	Alphanumérique
	Bloquer	Taxieur bloqué ou pas	Booleen
	Note	Note du taxieur	Numérique(entier)
	Longitude-taxieur	Position verticale du taxi	Numérique
	Latitude-Taxieur	Position horizontale du taxi	Numérique
	Pseudo-taxieur	Pseudonyme du taxieur	Alphanumérique
	Num-tel	Numéro du téléphone du taxieur	Numérique(entier)
	Nbr_signalement	Nombre de fois que le taxieur a été signalé par le client	Numérique(entier)
Réservation	Distance	Distance du trajet	Numérique
	Source	Source du client	Alphanumérique
	Destination	Destination du client	Alphanumérique
	Durée	Durée du trajet	Numérique
	Prix	Montant de la course	Numérique(entier)

TABLE 4.5 – Dictionnaire de données.

- **Elaboration du diagramme de classes**

La figure 4.8 représente le diagramme de classes de notre système

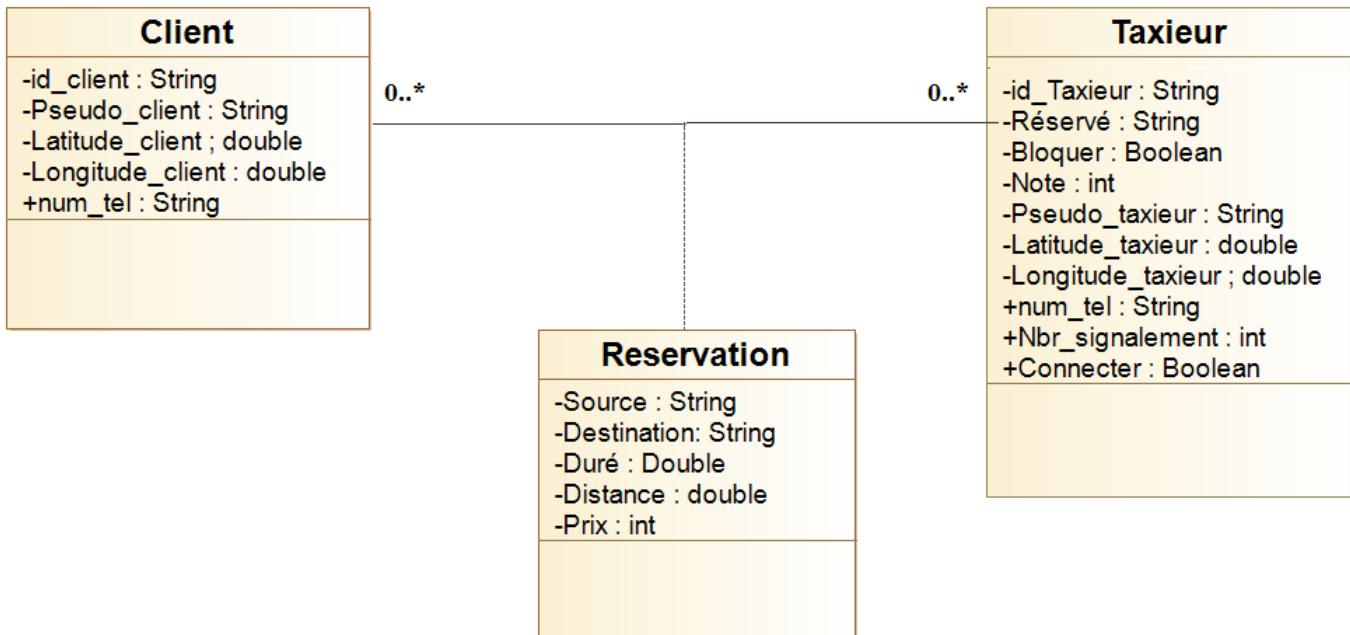


FIGURE 4.8 – Diagramme de classes.

4.6.5 Crédation de la base de données

Nous assistons ces dernières années à un changement des habitudes des utilisateurs notamment sur l’Internet mondial avec la démocratisation des offres haut débits. Dans ce contexte certaines personnes se sont rendu compte qu’un modèle relationnel des données atteignait ses limites : le NoSQL allait faire son entrée dans le monde de la représentation des données.

Le NoSQL, pour "not only SQL" désigne les bases de données qui ne sont pas fondées sur l’architecture classique des bases de données relationnelles. Développé à l’origine pour gérer du Big Data, l’utilisation des bases de données NoSQL a explosé depuis quelques années, elles ont été conçues pour résoudre les problèmes de traitements de données en volume, multi-sources et multi-formats [19].

Aujourd’hui le NoSQL apporte une nouvelle façon d’appréhender la modélisation des données. Le NoSQL n’est pas là pour remplacer les bases de données relationnelles, il répond à des besoins différents mais les deux approches peuvent cohabiter. Le choix de l’une ou de l’autre sera donc fortement dépendant du contexte et du besoin.

L’intérêt des systèmes de stockage NoSQL réside surtout dans les choix d’architecture logicielle qui ont été pris lors de leurs conceptions. Parmi les raisons principales qui ont mené à la création de ces systèmes, on retrouve surtout deux points principaux :

- La possibilité d'utiliser autre chose qu'un schéma fixe sous forme de tableaux dont toutes les propriétés sont fixées à l'avance .
- La possibilité d'avoir un système facilement distribué sur plusieurs serveurs et avec lequel un besoin supplémentaire en stockage ou en montée en charge se traduit simplement par l'ajout de nouveaux serveurs.

Même si initialement le NoSQL est une réponse à la croissance toujours plus importante sur Internet, il trouve également sa place dans le monde de l'entreprise sur des échelles moindres.

Notre application relève des systèmes distribués et gère des données hébergés sur des serveurs et qui changent en temps réel nous avons donc besoin de stocker nos données en temps réel et y accéder aussi rapidement, et les bases de données traditionnelles à savoir relationnelles n'arrivent pas à fournir un temps de réponse assez satisfaisant, c'est la raison pour laquelle, nous avons opté donc pour l'utilisation des bases de données NoSQL.

- **Les familles de bases de données NoSQL**

La mise en œuvre du bases de données NoSQL est possible grâce à la dénormalisation, dans une relation entre deux tables A et B, la dénormalisation qui consiste à dupliquer certaines données de la table B dans la table A afin d'optimiser les requêtes qui pourront se contenter d'interroger la table A sans avoir à faire de jointures entre A et B.

Nous allons présenter dans ce qui suit, les principales familles du concept NoSQL.

a. Clé/valeur :

Le but de la famille clé-valeur est l'efficacité et la simplicité. Un système clé-valeur agit comme une énorme table de hachage distribuée sur le réseau. Tout repose sur le couple Clé/Valeur. La clé identifie la donnée de manière unique et permet de la gérer. La valeur contient n'importe quel type de données [19].

b. En colonnes :

Traditionnellement, les données sont représentées en ligne, représentant l'ensemble des attributs. Le stockage orienté colonne change ce paradigme en se focalisant sur chaque attribut et en les distribuant. Il est alors possible de focaliser les requêtes sur une ou plusieurs colonnes, sans avoir à traiter les informations inutiles (les autres colonnes) [19].

c. Orientées graphes :

Dans la base orientée graphe, les données stockées sont : les nœuds, les liens et des propriétés sur ces nœuds et ces liens. Les requêtes que l'on peut exprimer sont basées sur la gestion de chemins [19].

d. Orientées document :

Les bases orientées documents ressemblent sans doute le plus à ce que l'on peut faire dans une base de données classique pour des requêtes complexes. Le but de ce stockage est de

manipuler des documents contenant des informations avec une structure complexe (types, listes, imbrications). Il repose sur le principe de la clé/valeur, mais avec une extension sur les champs qui composent ce document [19]. La figure 4.9 illustre un exemple de cette famille

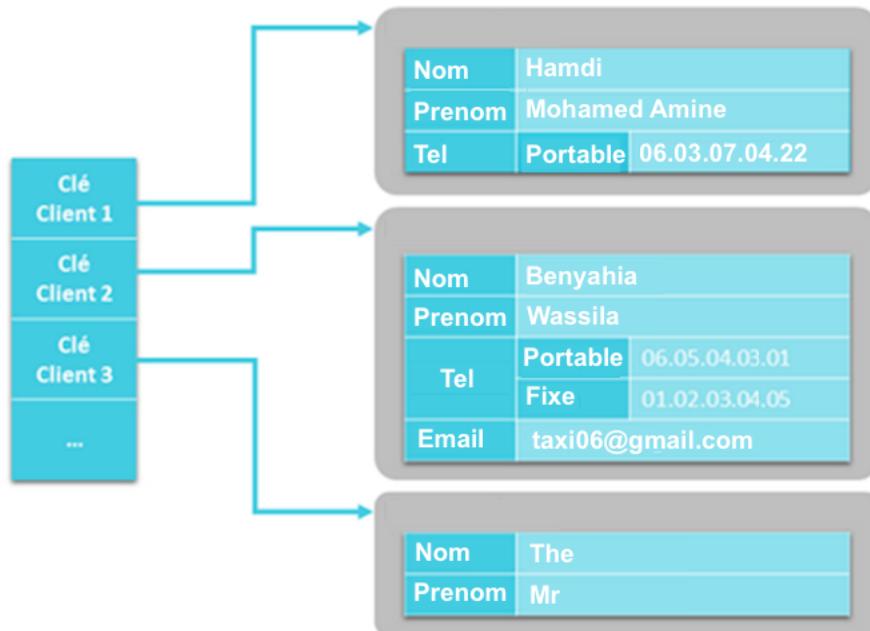


FIGURE 4.9 – Exemple d'une BDD NoSQL orienté document.

Dans le cadre de notre application, notre choix est porté sur la conception orientée document et ceci pour les raisons suivantes :

- L'avantage de cette solution est d'avoir une approche structurée de chaque valeur, formant ainsi un document.
- les documents sont sous le format du très populaire format d'échange de données Json (JavaScript Object Notation), ce qui nous permet donc une grande interopérabilité.
- Elle nous permet une sérialisation de nos objets sans mapping : on stock directement nos objets java et on les récupère en tant qu'objets.
- Nous pouvons requêter et manipuler ces documents, et notamment récupérer, via une seule clé, un ensemble d'informations structurées de manière hiérarchique.

- **Schéma de notre base de données**

la figure illustré ci-dessous représente notre schéma de base de données qui est en NoSQL de type orienté document.

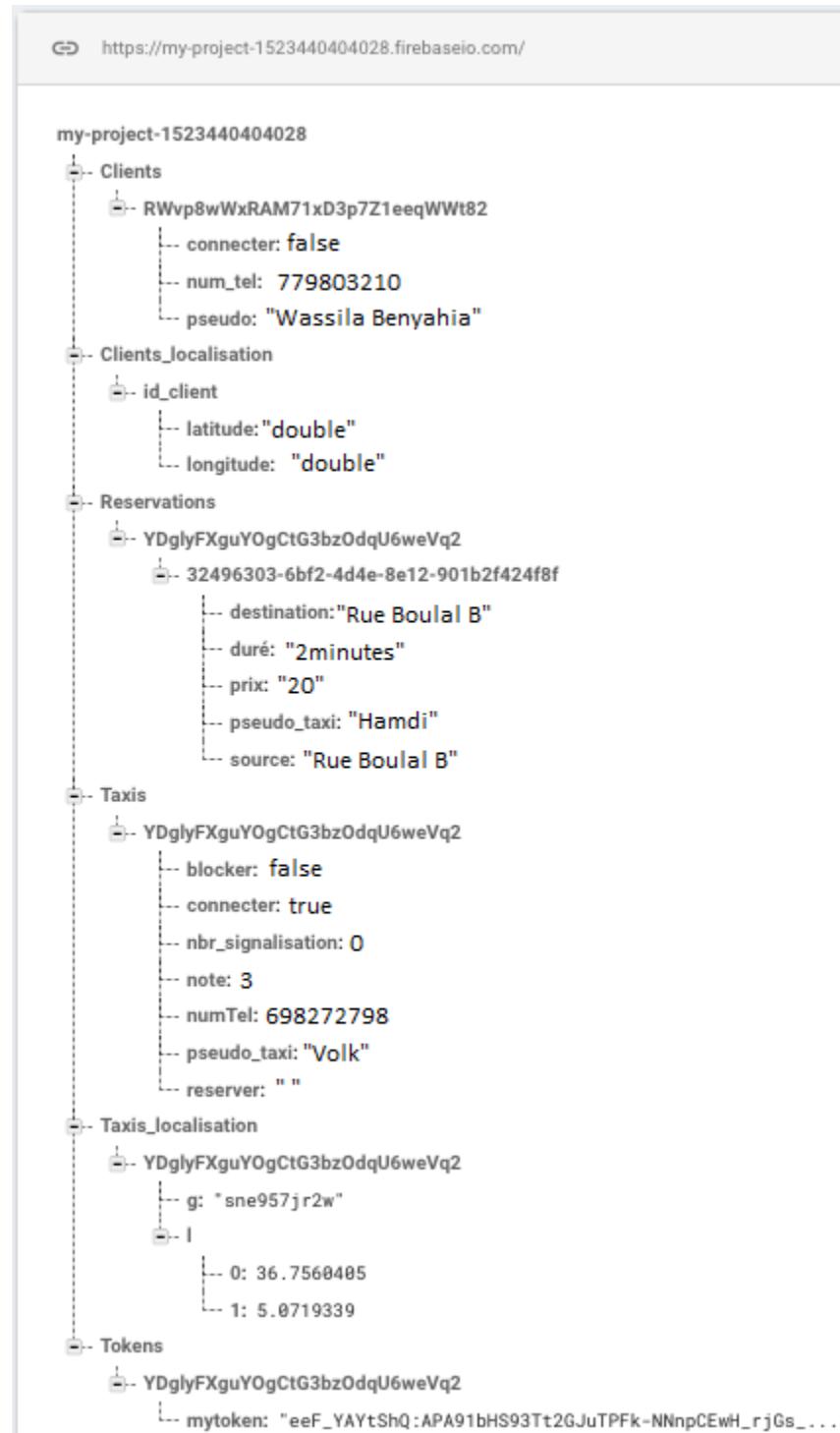


FIGURE 4.10 – Schéma de notre BDD.

4.6.6 Diagramme de déploiement

Le diagramme de déploiement illustré dans la figure 4.11 démontre le matériel nécessaire qu'on a utilisé dans notre projet et qui représente un terminal mobile, en plus des différentes API utilisées et des services de firebase auxquels on a eu recours, sans oublier l'utilisation de l'internet qui a été indispensable, de plus les échanges de données se sont fait en xml et json.

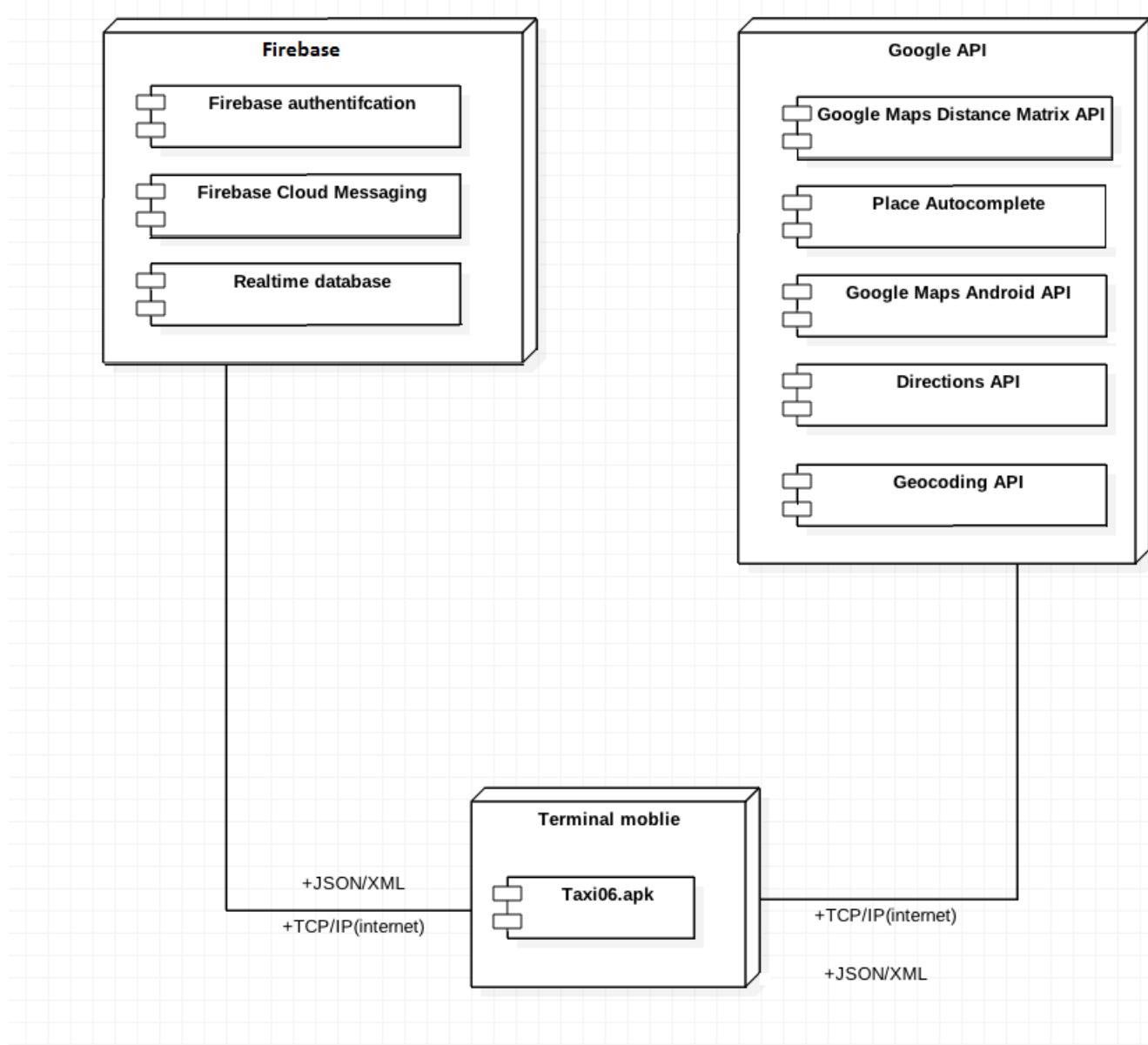


FIGURE 4.11 – Diagramme de déploiement

4.7 Conclusion

Nous avons présenté dans ce chapitre la phase d'analyse et de conception de notre système en utilisant la démarche du processus unifié et les diagrammes UML appropriés, ceux que nous avons jugé indispensable à notre travail.

Dans le chapitre suivant, nous entamerons la réalisation de notre système en présentant les langages de programmation, outils et environnement de développement utilisés pour la mise en œuvre de notre application. Nous présentons entre autre quelques interfaces graphiques explicatives de l'application réalisée.

Chapitre 5

Expérimentation

5.1 Introduction

C'est ainsi que nous poursuivons avec la dernière étape de notre travail qui se traduit par la phase de réalisation. Ceci en précisant l'environnement logiciel, les outils de développements utilisés ainsi que les langages de programmation. Ces derniers, nous ont permis d'arriver au résultat voulu, que nous présentons à travers les différentes interfaces réalisées.

5.2 Présentation de l'environnement de développement

Android Studio est un nouvel environnement pour développement et programmation entièrement intégré qui a été lancé par Google pour les systèmes Android. Il a été conçu pour fournir un environnement de développement et une alternative à Eclipse qui est l'IDE le plus utilisé [16].

Parmi ses avantages on peut citer les suivants :

- Un environnement de développement robuste .
- Permet de voir chacun des changements visuels effectué dans l'application en temps réel .
- Une manière simple pour tester les performances sur d'autres types d'appareils .
- Firebase est intégré à l'IDE .
- Un éditeur complet avec une panoplie d'outils pour accélérer le développement des applications.

5.3 Outils de développements

Les outils de développements que nous avons utilisés sont les suivants :

5.3.1 SDK

L'outil le plus important est le SDK Android. Facile à installer, il permet de télécharger tous les outils indispensables au développement d'applications. Il permet d'abord de télécharger les différentes versions de SDK, mais aussi de télécharger les différentes versions de Google APIs [16].

5.3.2 JDK

Afin de développer des programmes Java, le programmeur doit disposer d'un compilateur Java et des bibliothèques de compilation, ainsi que des bibliothèques de support pour les tâches de programmation régulière, telles que le débogage, le kit de développement Java représente la collections de ces bibliothèques, y compris le « javac » compilateur Java. Le JDK contient le compilateur, des bibliothèques de programmes et le JRE afin de présenter aux programmeurs une plateforme permettant de compiler et d'exécuter leurs programmes [21].

5.3.3 AVD

L'Android Virtual Device est un émulateur de terminal sous Android, il permet de lancer sur la machine du développeur un terminal virtuel représentant à l'écran un téléphone embarquant Android. C'est bien évidemment un outil indispensable pour le développement mobile. A chaque version d'Android est associée une version de l'émulateur, permettant au développeur de voir exactement à quoi ressemblera son application sur un matériel réel [16].

5.4 Langages de développements

5.4.1 Java

Java est un langage de programmation orienté objet moderne développé par Sun Microsystems (aujourd'hui racheté par Oracle), son squelette principal est constitué du langage C++. Java n'a rien à voir avec JavaScript qui est utilisé principalement pour les sites. Une de ses plus grandes forces est son excellente portabilité : une fois votre programme créé, il fonctionnera automatiquement sous Windows, Mac, Linux, etc [21].

5.4.2 XML

Le XML est un langage de balisage un peu comme le HTML — le HTML est d'ailleurs indirectement un dérivé du XML. Le principe d'un langage de programmation (Java, C++, etc.) est de donner des ordres à l'ordinateur pour qu'il puisse effectuer des calculs, puis éventuellement de mettre en forme le résultat de ces calculs dans une interface graphique. À l'opposé, un langage de balisage (comme le XML) n'a pas pour objectif de donner des ordres à l'ordinateur, il se contente de définir une façon de mettre en forme l'information de façon à ce qu'on sache comment lire cette information [16].

5.4.3 JSON

JSON est un format léger d'échange de données. Il est facile à lire ou à écrire pour des humains. Il est aisément analysable ou générable par des machines. Il est basé sur un sous-ensemble du langage de programmation JavaScript . JSON est un format texte complètement indépendant de tout langage, mais les conventions qu'il utilise seront familières à tout programmeur habitué aux langages descendant du C, comme par exemple : C lui-même, C++, C#, Java, JavaScript, Perl, Python et bien d'autres. Ces propriétés font de JSON un langage d'échange de données idéal [22].

5.5 Présentation de Firebase

Les plateformes de Cloud ont révolutionné la conception des applications grâce aux produits internationaux basés sur le : calcul, stockage, base de données, analyse, mise en réseau, service mobile... etc. Parmi eux on peut distinguer « Firebase » qui est l'un des cloud les plus performants développés par Google [23].

5.5.1 Définition

Firebase est une plateforme mobile créée en 2011 par James Tamplin et Andrew Lee, puis rachetée par Google en 2014 pour être intégrée à leur offre de services Cloud (Google Cloud Platform).

L'objectif premier de Firebase est de vous libérer de la complexité de création et de la maintenance d'une architecture serveur, tout en vous garantissant une scalabilité à toute épreuve (plusieurs milliards d'utilisateurs) et une simplicité dans l'utilisation [23].

5.5.2 Utilité de Firebase

Lorsqu'on développe une application, qu'elle soit destinée au grand public ou réservée à un usage interne à l'entreprise, certaines fonctionnalités sont systématiquement requises, telles que la gestion des utilisateurs, de la connexion et des notifications. La gestion de ces fonctionnalités est fastidieuse, répétitive si votre SI se compose de plusieurs applications, et critiques en termes de sécurité, dans la mesure où l'on va stocker des mots de passe.

Firebase permet d'externaliser cette gestion, en même temps qu'il offre la possibilité de proposer de manière unifiée des connexions Facebook, Twitter, Google. De plus, Firebase offre des SDK pour chaque environnement de développement classique [23].

5.5.3 Les services utilisés

Firebase a été décomposée en plusieurs produits extrêmement riches et adaptés au monde du mobile, ceux que nous avons utilisés et qui sont considérés comme principaux sont les suivants :

1. **Realtime Database** : La base de données Firebase Realtime Database est une base de données NoSQL hébergée dans le cloud qui vous permet de stocker et de synchroniser des données entre vos utilisateurs en temps réel [23].
2. **Authentication** : Solution permettant de créer et gérer facilement des moyens d'authentification variés (Google, Facebook, Email, etc...) dans le but de sécuriser l'accès à une application mobile et authentifier les utilisateurs [23].
3. **Cloud Messaging** : Fournit un flux de communication fiable et économique en batterie entre le serveur (Firebase) et les appareils distants (où l'application est installée) dans l'objectif d'envoyer et recevoir des messages de notifications [23].

5.6 Les services web utilisés

Un Service Web est une fonctionnalité qu'on peut appeler à distance à travers un réseau (comme un intranet d'entreprise ou internet lui-même). Cette fonctionnalité a été réalisée par un tiers langage et qui peut être invoquée à travers des messages XML ou JSON.

Nous distinguons deux générations de Services Web :

- Service web 1.0 basé sur le protocole SOAP.
- Service web 2.0 basé sur le style architectural REST.

SOAP est un protocole tandis que REST est un style d'architecture.

La différence majeure entre ces 2 éléments est le degré de liaison entre le client et le serveur. Un client développé avec le protocole SOAP ressemble à un logiciel d'ordinateur, car il est étroitement lié au serveur. Si une modification est effectuée d'un côté ou de l'autre, l'ensemble peut ne plus fonctionner. Il faut effectuer des mises à jour du client s'il y a des changements sur le serveur et vice-versa.

Un client de type REST sait utiliser un protocole et des méthodes standardisées. Son application doit rentrer dans ce modèle. On ne crée pas de méthodes supplémentaires, on utilise les méthodes standardisées que l'on développe pour le type de media dont on a besoin. Il y a en conséquence beaucoup moins de couplage entre le client et le serveur : un client peut utiliser un service de type REST sans aucune connaissance de l'API. A l'inverse, un client SOAP doit tout savoir des éléments qu'il va utiliser pendant son interaction avec le serveur, sinon cela ne fonctionnera pas.

Par ailleurs, REST possède des caractéristiques spécifiques qui le différencient de SOAP. REST est indépendant d'un protocole. On peut utiliser aussi bien le protocole HTTP que FTP, tant qu'il s'agit d'un protocole possédant un schéma standard pour une URI.

Vu la simplicité du style REST, nous avons opté pour l’invocation des services web REST existants avec des messages JSON.

Dans le cadre de notre application, nous avons utilisé quelques services web existants.

5.6.1 Google Maps Distance Matrix

Le service Google Maps Distance Matrix API fournit les distances et les durées des trajets pour une matrice de points de départ et de destinations[24].

5.6.2 Place AutoComplete

Le service Place Autocomplete (saisie-semi-automatique de lieu) est un service Web qui renvoie des prédictions de lieu en réponse à une requête HTTP. La requête spécifie une chaîne de recherche textuelle et des limites géographiques facultatives. Le service permet de fournir la fonctionnalité de saisie semi-automatique pour les recherches géographiques textuelles et renvoie des lieux tels que des entreprises, des adresses et des points d’intérêt au fur et à mesure de la saisie par l’utilisateur[24].

5.6.3 Google Maps Android API

C'est un service web qui sert à ajouter Google Maps à votre application Android[24].

5.6.4 Direction API

Google Maps Directions API est un service qui calcule des itinéraires entre des points géographiques [24].

5.6.5 Geocoding API

Google Maps Geocoding API est un service qui effectue le géocodage et le géocodage inversé d’adresses[24].

Le géocodage : est le processus qui permet de convertir des adresses (comme une adresse postale) en coordonnées géographiques (comme la latitude et la longitude) que vous pouvez ensuite utiliser pour placer des marqueurs sur une carte ou pour positionner la carte.

Le géocodage inversé : est le processus de conversion de coordonnées géographiques en adresses lisibles. Le service de géocodage inversé de Google Maps Geocoding API vous permet également de retrouver l’adresse correspondant à un identifiant de lieu donné.

5.7 Geofire

GeoFire est un ensemble de bibliothèques open-source pour JavaScript, Objective-C et Java qui vous permettent de stocker et interroger un ensemble de clés en fonction de leur emplacement géographique. En son cœur, GeoFire stocke simplement les emplacements avec des clés de chaîne. Son principal avantage, cependant, est la possibilité de récupérer uniquement les clés dans une zone géographique donnée, le tout en temps réel.

GeoFire utilise la base de données Firebase pour le stockage des données, ce qui permet de mettre à jour les résultats de la requête en temps réel à mesure qu'ils changent.

GeoFire charge de manière sélective uniquement les données à proximité de certains emplacements, en gardant vos applications légères et réactives, même avec des ensembles de données extrêmement volumineux [25].

5.8 Logiciel de gestion de version

Il existe de nombreux logiciels de gestion de versions, comme SVN (Subversion), Mercurial et Git. Pour notre projet, nous avons utilisé Git (prononcez « guite ») qui est un des plus puissants logiciels de ce genre

5.8.1 Définition

Les logiciels de gestion de versions sont utilisés principalement par les développeurs . ce sont donc bel et bien des outils pour geeks. En effet, ils sont quasi exclusivement utilisés pour gérer des codes sources, car ils sont capables de suivre l'évolution d'un fichier texte ligne de code par ligne de code. Ces logiciels sont fortement conseillés pour gérer un projet informatique. Ces outils suivent l'évolution de vos fichiers source et gardent les anciennes versions de chacun d'eux [26].

5.8.2 Fonctionnalités

Les logiciels de gestion de versions proposent de nombreuses fonctionnalités qui vont vraiment vous être utiles tout au long de l'évolution de votre projet informatique [26].

- Ils retiennent qui a effectué chaque modification de chaque fichier et pourquoi. Ils sont par conséquent capables de dire qui a écrit chaque ligne de code de chaque fichier et dans quel but .
- Si deux personnes travaillent simultanément sur un même fichier, ils sont capables d'assembler (de fusionner) leurs modifications et d'éviter que le travail d'une de ces personnes ne soit écrasé.

Ces logiciels ont donc par conséquent deux utilités principales

- **suivre l'évolution d'un code source**, pour retenir les modifications effectuées sur chaque fichier et être ainsi capable de revenir en arrière en cas de problème .

- **travailler à plusieurs**, sans risquer de se marcher sur les pieds. Si deux personnes modifient un même fichier en même temps, leurs modifications doivent pouvoir être fusionnées sans perte d'information.

5.8.3 Les principaux types de logiciels de gestion de versions

Il existe deux types principaux de logiciels de gestion de versions [26].

- **Les logiciels centralisés :**

Un serveur conserve les anciennes versions des fichiers et les développeurs s'y connectent pour prendre connaissance des fichiers qui ont été modifiés par d'autres personnes et pour y envoyer leurs modifications. La figure 5.1 illustre ce type de logiciel.

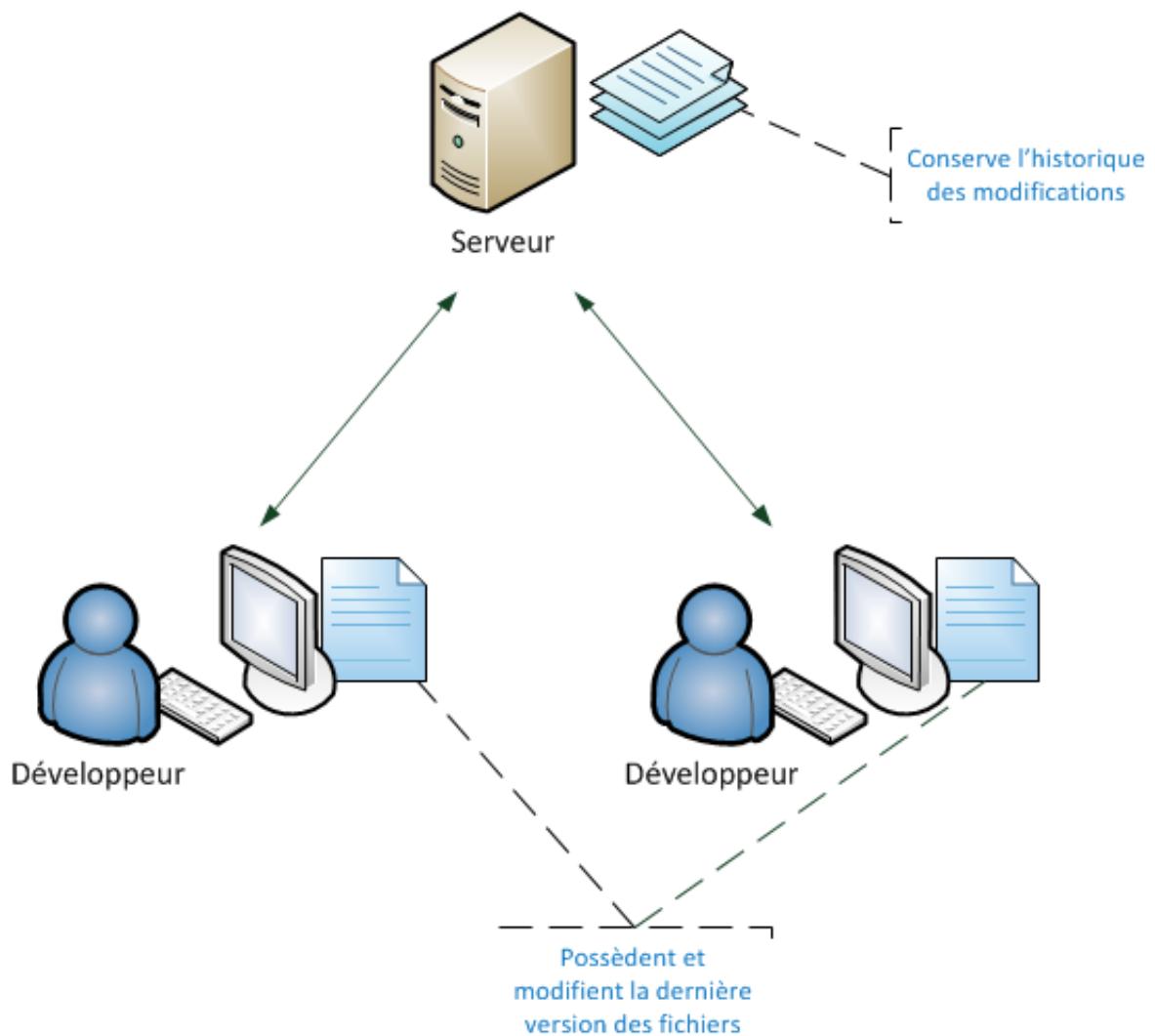


FIGURE 5.1 – Logiciel centralisé [26].

- **Les logiciels distribués :**

Il n'y a pas de serveur, chacun possède l'historique de l'évolution de chacun des fichiers. Les développeurs se transmettent directement entre eux les modifications, à la façon du peer-to-peer. La figure 5.2 illustre ce type de logiciel

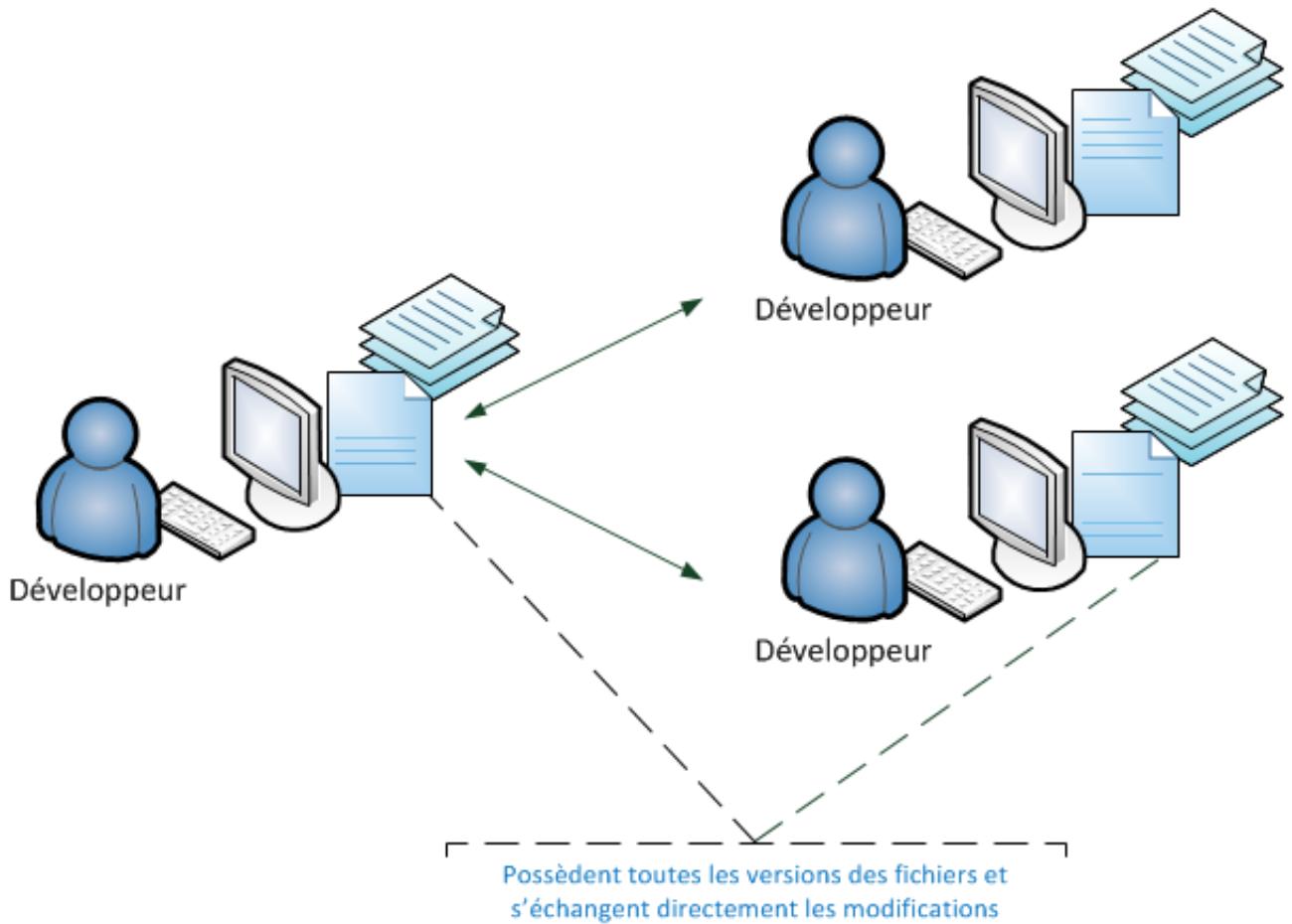


FIGURE 5.2 – Logiciel distribué [26].

5.9 Gestion de projets

Pour la gestion du projet nous avons utilisé le site web Trello et la méthode Kanban

5.9.1 Trello

Créé en 2011 par Fog Creek Software puis repris en 2014 par Trello Inc, Trello est un outil de gestion de projet collaboratif en ligne permettant à ses utilisateurs d'organiser leur projet selon la méthode Kanban. Cet outil est disponible sur le site web, sur google play store et sur l'apple store, ce qui vous offre la possibilité de consulter vos tableaux et vos cartes où vous voulez et quand vous voulez.

Trello est disponible en version gratuite et payante, ce qui permet aux particuliers comme aux entreprises de l'utiliser. La version payante offre à ses utilisateurs des fonctionnalités supplémentaires à la version gratuite [27]

5.9.2 La méthode Kanban

Il s'agit d'une méthode agile très utilisée des services informatiques pour la gestion des tickets du support mais aussi pour la gestion du développement et du déploiement de logiciels. Cette méthode permet de visualiser le flux de travail, de limiter le nombre de tâches en cours et de prioriser les tâches. Dans Trello, les tableaux représentent des projets, les cartes correspondent à des tâches et les listes sont les colonnes qui servent à organiser les tâches. Dans la plupart des cas, chaque liste représente le statut des tâches. Mieux qu'un simple post-it, les cartes de Trello peuvent également contenir des checklists, des commentaires, des pièces-jointes [27]. La figure 5.3 illustre un exemple d'un tableau dans Trello.

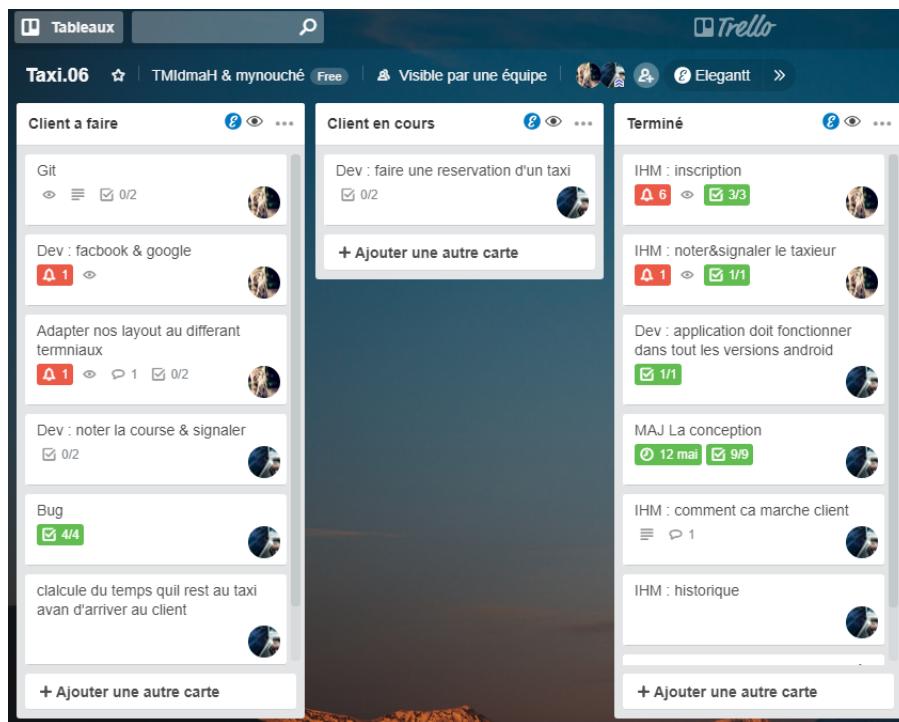


FIGURE 5.3 – Exemple d'un tableau Trello

5.10 Présentation des interfaces graphiques

Le volet technique de ce chapitre étant terminé, nous allons désormais consacrer cette partie du chapitre à la présentation des interfaces graphiques de l'application « Taxi06»

Notre application contient deux espaces utilisateurs, un espace réservé au client et un autre espace réservé au taxieur.

5.10.1 Interface de choix de l'utilisateur

La première étape avant d'accéder à l'interface d'inscription est de choisir le type de l'utilisateur de l'application, comme illustré dans la figure 5.4.

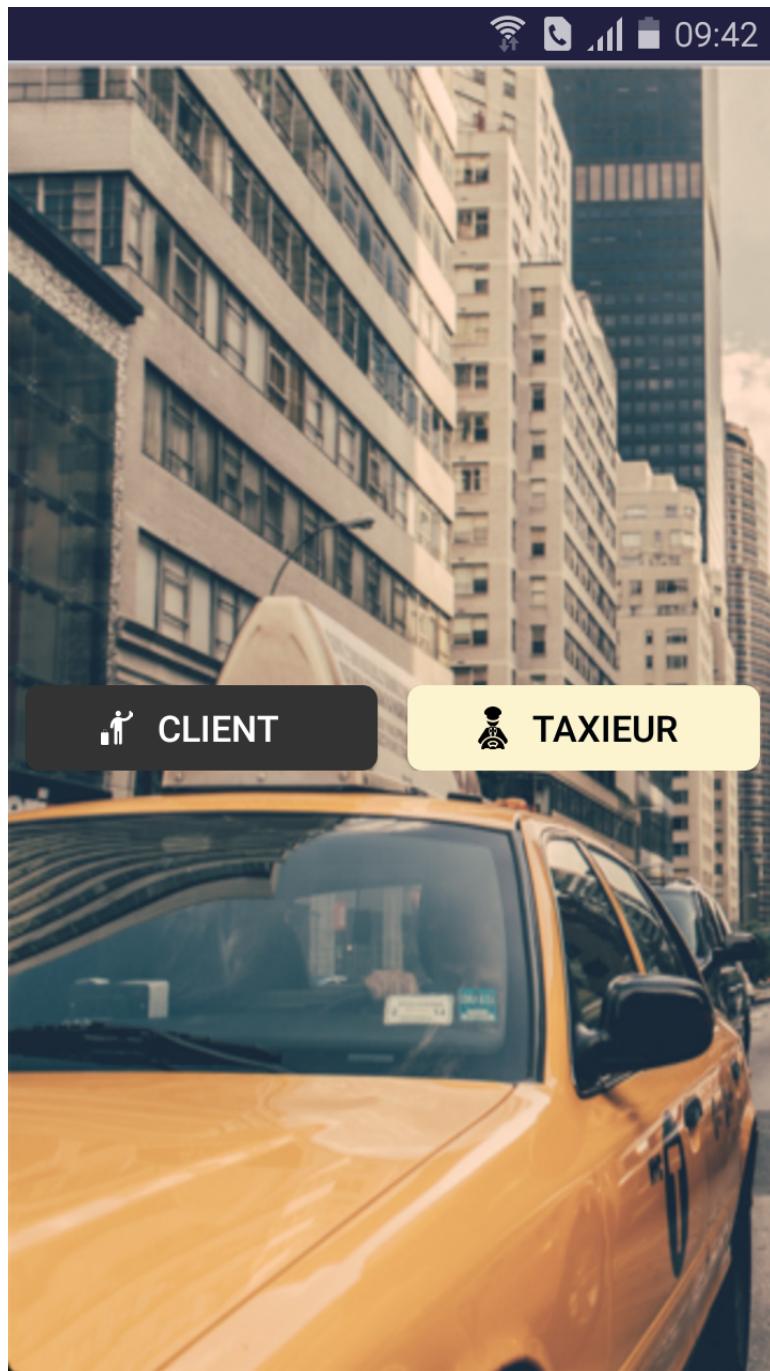


FIGURE 5.4 – Interface de choix de l'utilisateur.

5.10.2 Interface d'inscription

Avant toute action, un nouvel utilisateur devra s'inscrire avant de bénéficier de l'utilisation de l'application, il a le choix de s'inscrire soit par numéro de téléphone et cela en saisissant son numéro de téléphone et son pseudo et recevra un code de vérification par sms, soit par Facebook ou bien en utilisant son compte Google, cette étape est illustrée dans la figure suivante :

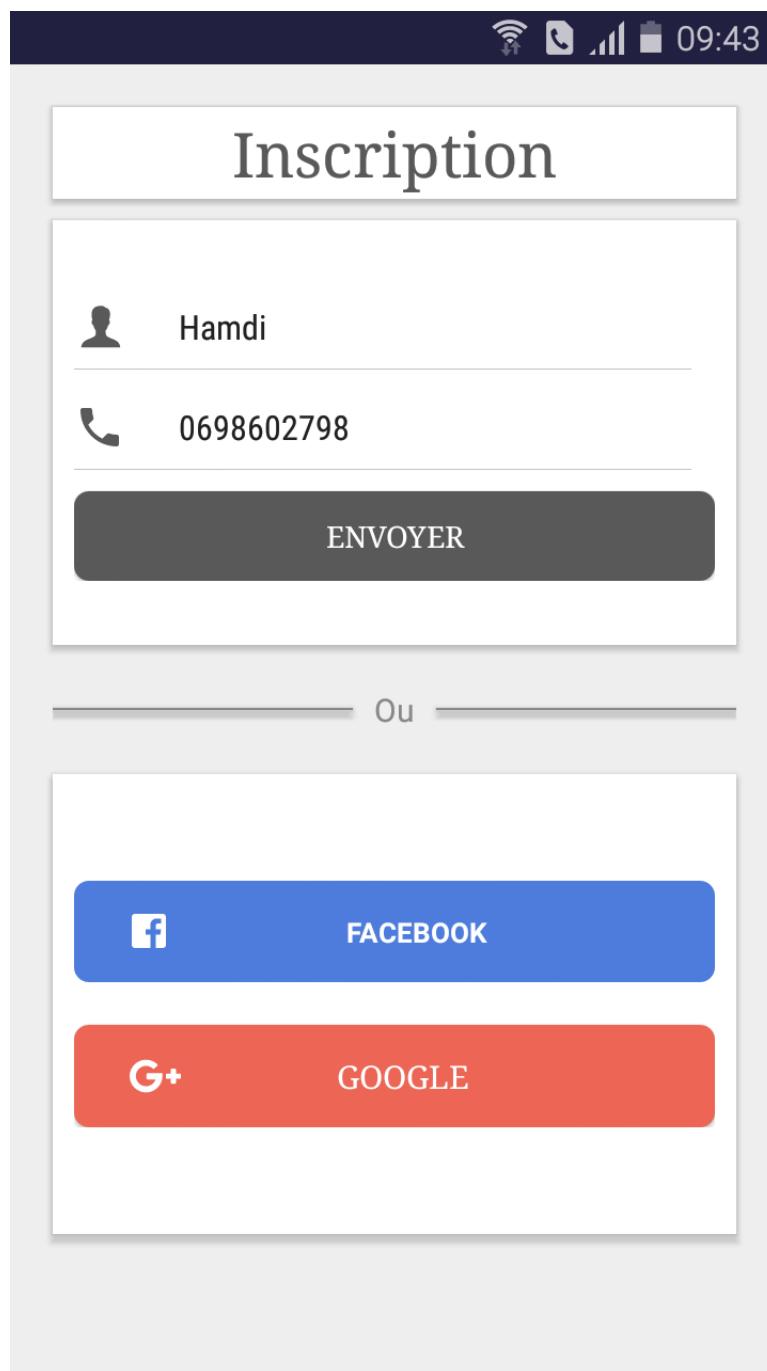


FIGURE 5.5 – Etape d'inscription

5.10.3 Interface principal du client

Lorsque le client accède à l'application, il pourra consulter son menu comme le montre la figure suivante :

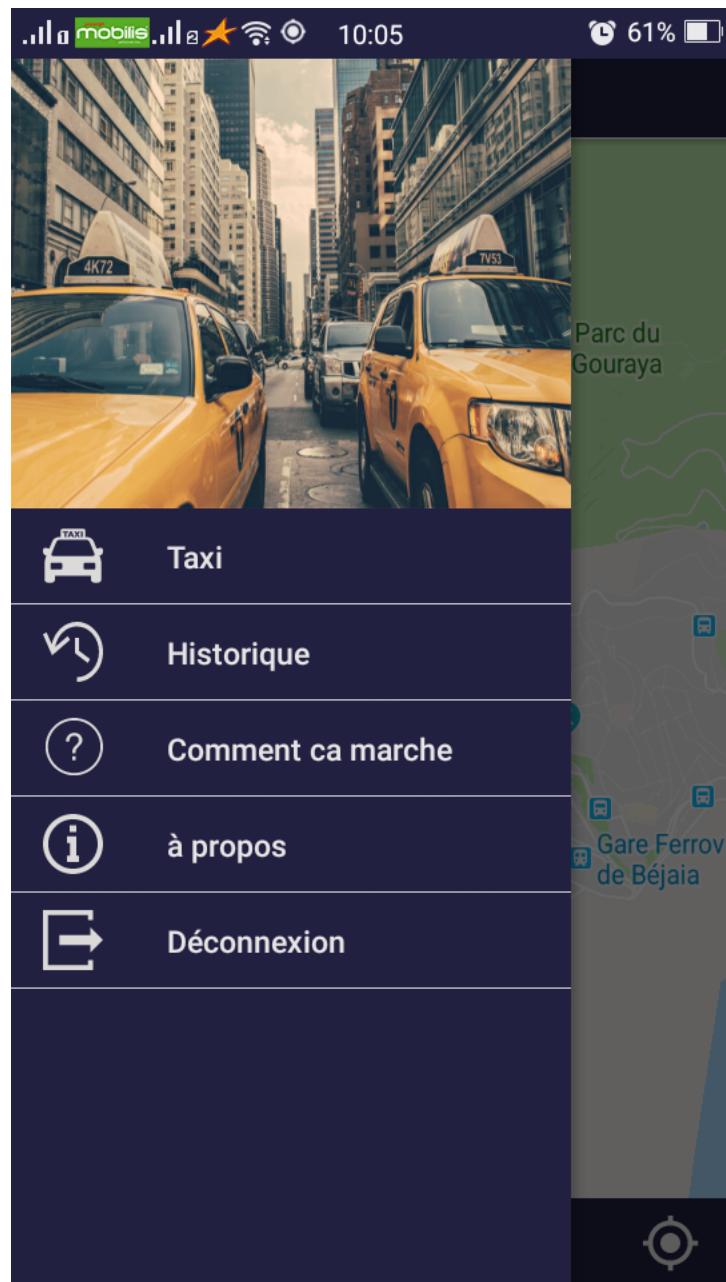
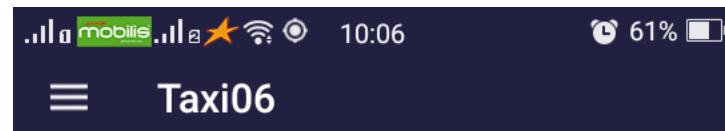


FIGURE 5.6 – Menu du client.

5.10.4 Interface "à propos"

La figure suivante illustre l'interface «à propos» du menu de l'utilisateur.



The image shows a detailed view of the 'About' screen for the Taxi06 app. The screen has a white background with a thin gray border. It contains four main sections separated by horizontal lines:

- Version**
Version 1.0
- Contactez-nous**
Taxi06@gmail.com
- Réalisé par :**
Mr Mohamed Amine HAMDI
Mlle Wassila BENYAHIA
- Encadré par**
Dr EL BOUHSSI BRAHAMI Houda

FIGURE 5.7 – Interface à propos

5.10.5 Interface de l'historique

La figure 5.8 illustre l'interface "Historique" qui se trouve dans le menu du client et qui sert à consulter les réservations de taxi qu'il a effectué

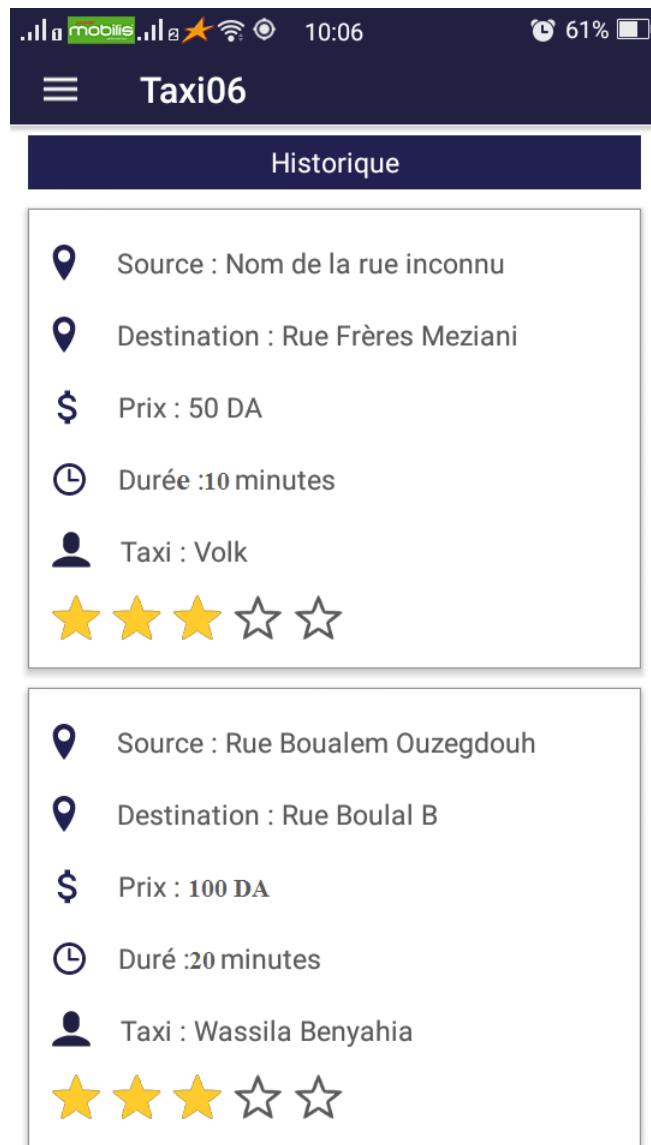


FIGURE 5.8 – Interface de l'historique

5.10.6 Etapes de Réservation

Lorsque le client se connecte à l'application, il aura un aperçu de tous les taxieurs qui se trouvent à un rayon de 5 Km , la première étape sera d'indiquer la destination où il veut se rendre, ensuite il choisira un taxi comme le montre les figures suivantes

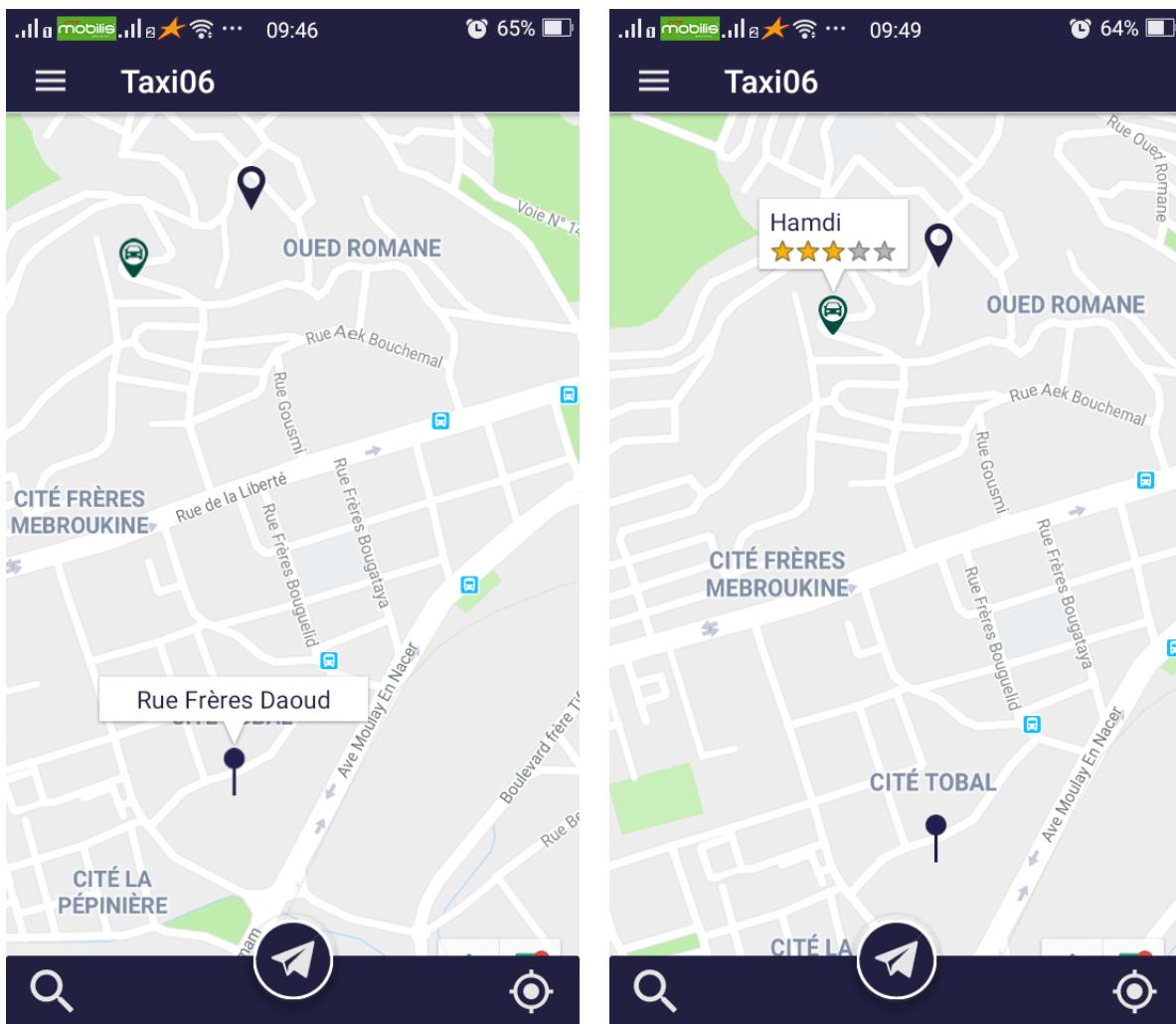


FIGURE 5.9 – Le client indique la destination/Il choisi un taxi

Dans le cas où le client souhaite visualiser les informations de la demande de réservation avant de les envoyer, il clique sur recherche, il pourra aussi saisir sa destination manuellement s'il n'a pas encore choisi sa destination sur la map, comme le montre les figures suivantes :

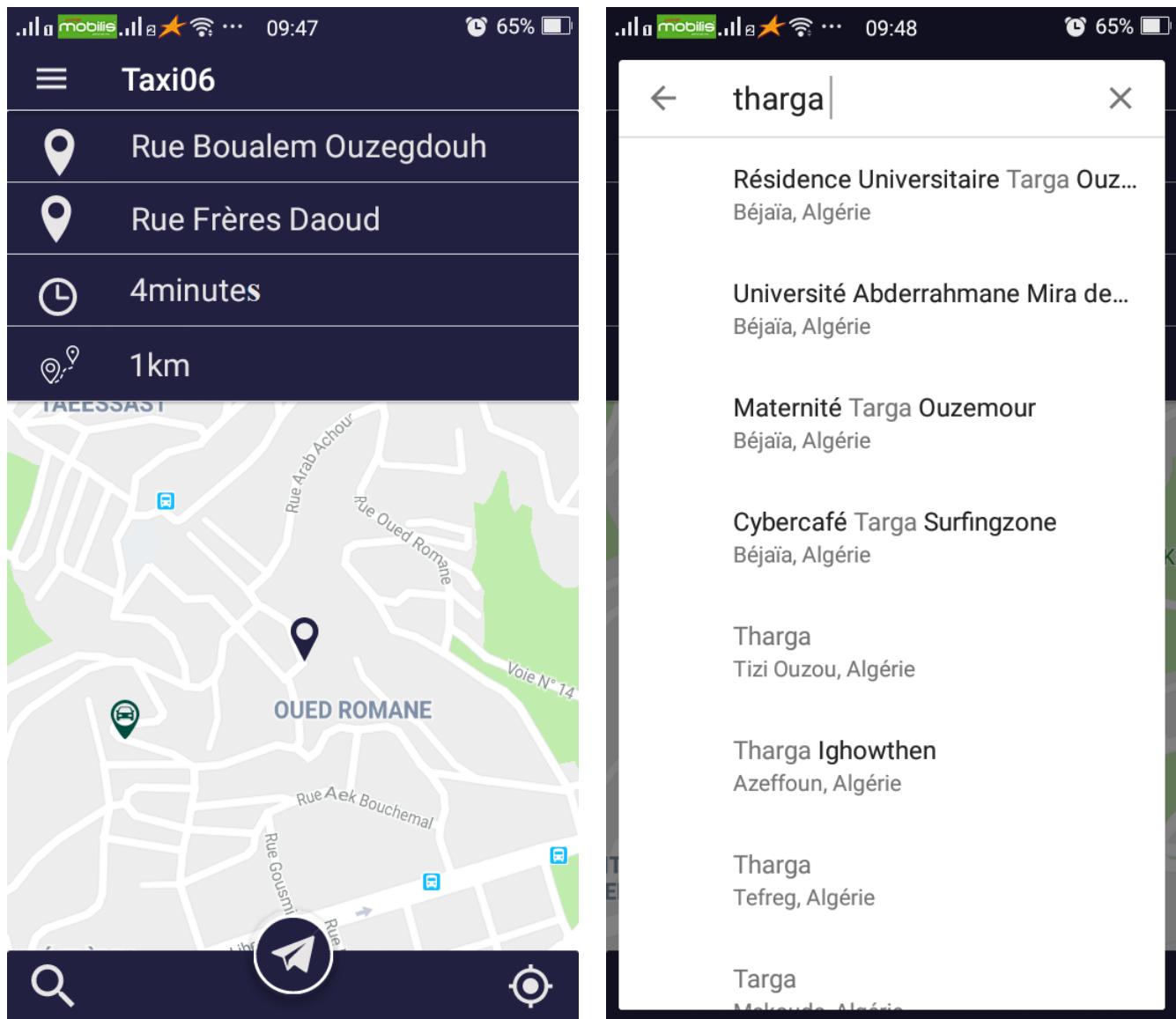


FIGURE 5.10 – Cliquer sur recherche/modifier la destination

Après avoir sélectionné la destination et choisi le taxieur, le client doit confirmer la demande de réservation avant de l'envoyer au taxieur comme le montre la figure suivante :

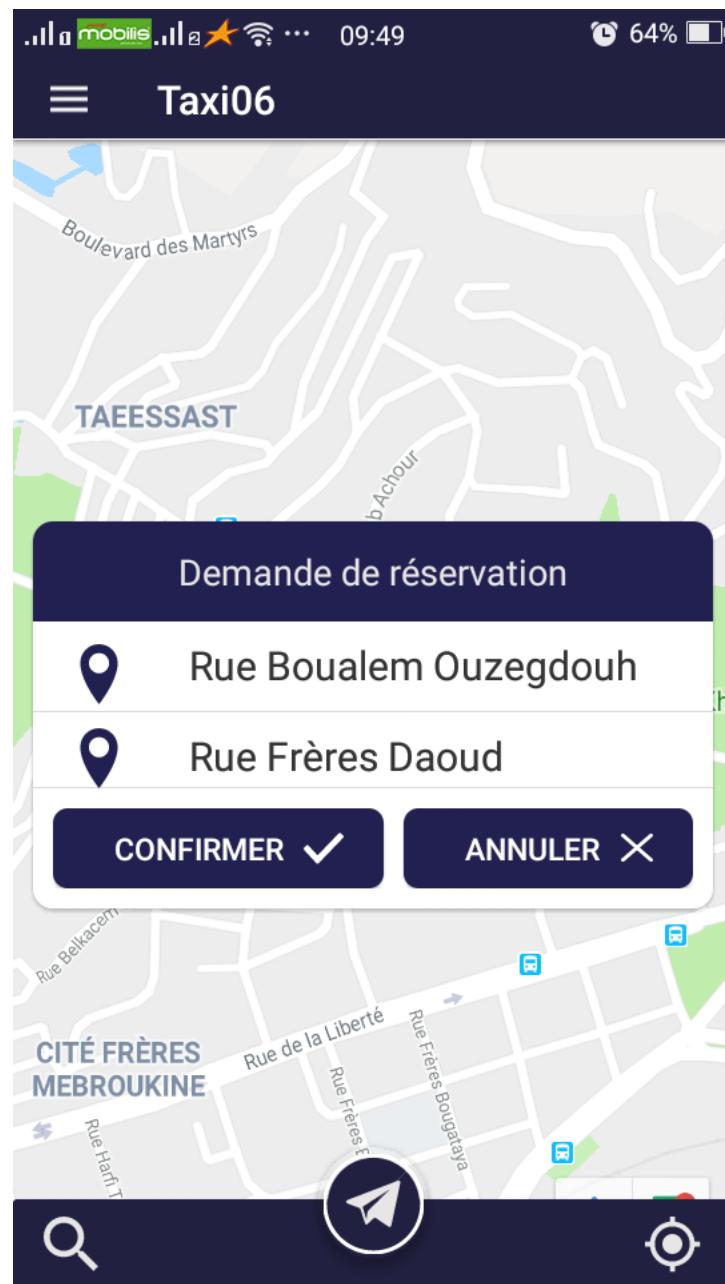


FIGURE 5.11 – Le client confirme l'envoi de la demande

A son tour le taxieur reçoit une notification de la demande de réservation qui provient du client et l'itinéraire sera affiché dans sa map comme suit : L'itinéraire entre le taxieur et le client sera affiché en vert, et celui du client vers la destination choisi sera affiché en bleu.

Si le taxieur décide d'accepter la demande, il devra d'abord indiquer le prix ensuite cliquer sur Accepter. Les étapes ci-dessous sont illustrées dans les figures suivantes :

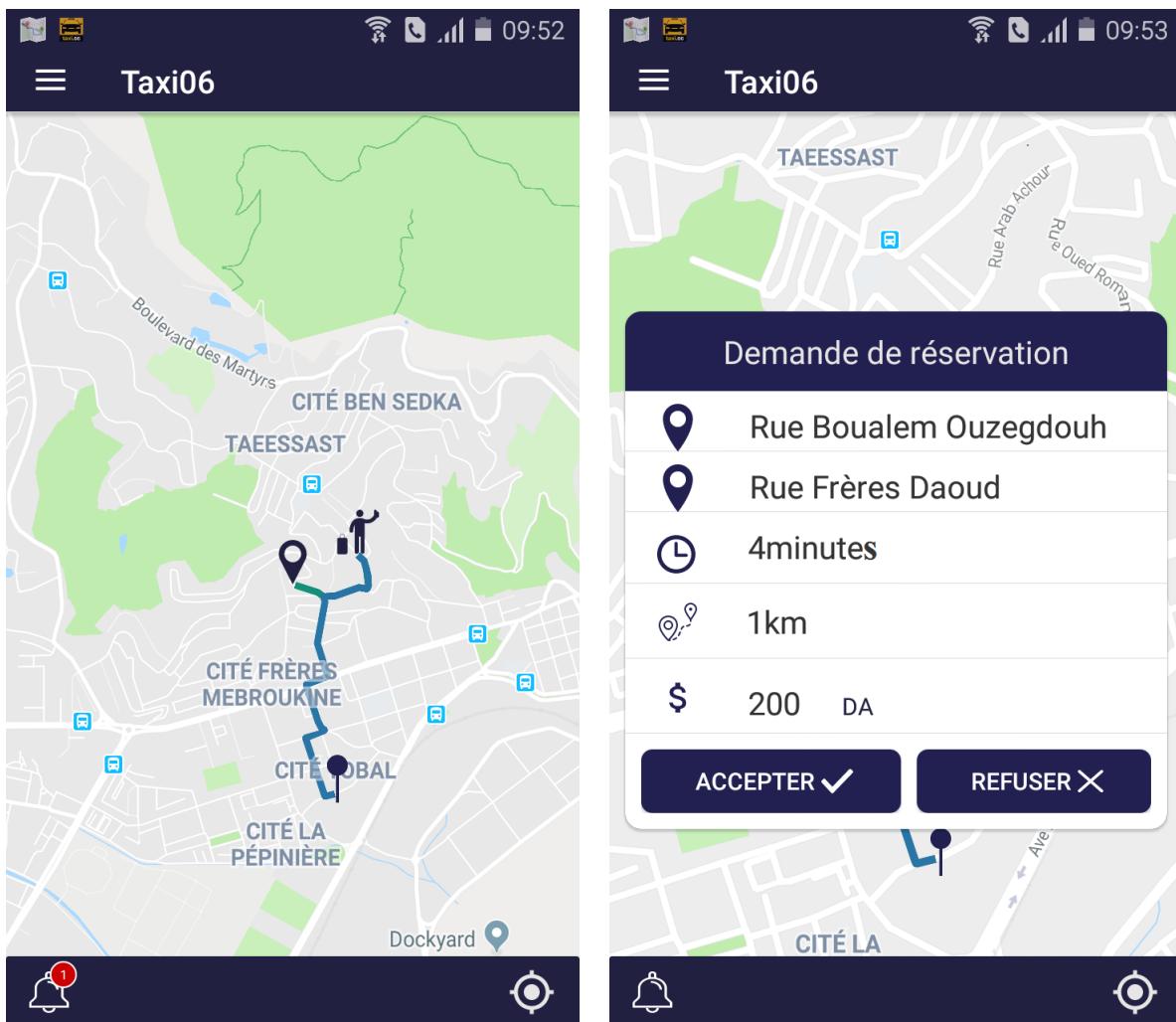


FIGURE 5.12 – Etapes de réception et d'acceptation d'une demande de réservation

Le client reçoit le prix indiqué par le chauffeur de taxi, et pourra accepter ou refuser le prix comme le montre la figure 5.13, s'il décide d'accepter le prix le taxieur recevra un message indiquant que la course a été confirmée.

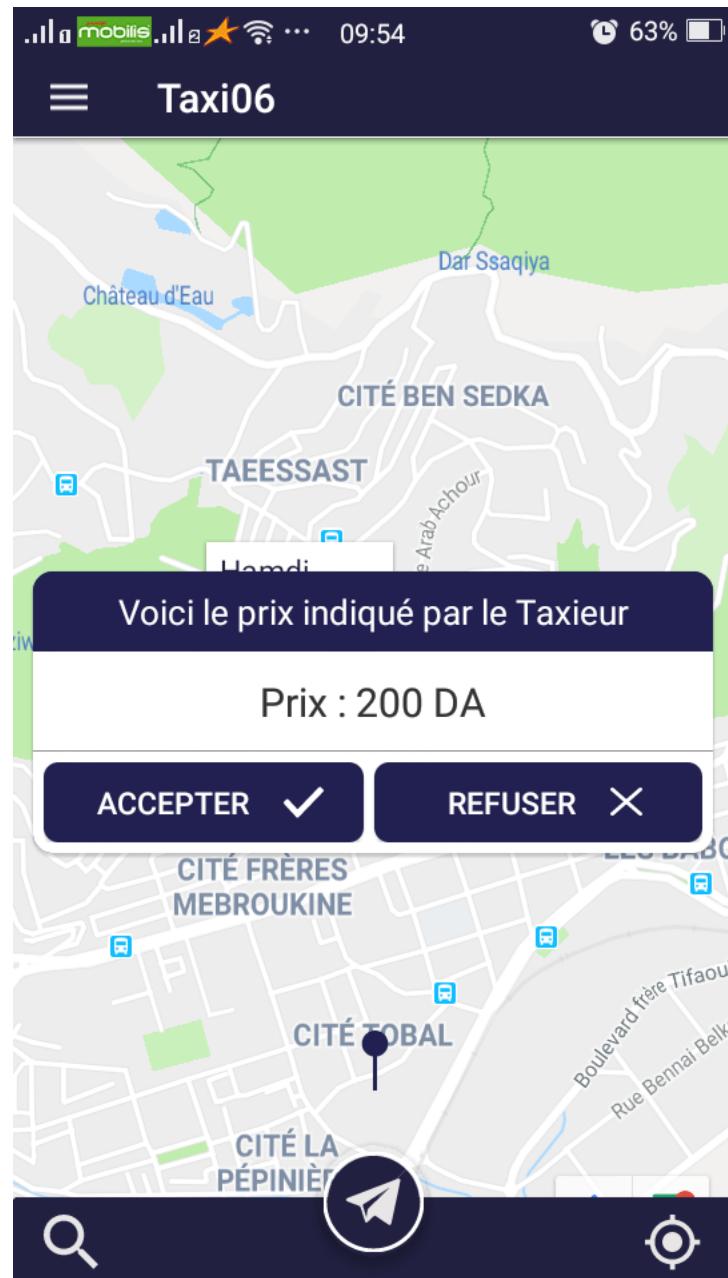


FIGURE 5.13 – Le client accepte le prix

5.11 Conclusion

Dans ce chapitre, nous avons décrit les aspects pratiques liés à la réalisation de notre application « Taxi06 », à savoir l'environnement, les outils de développement, ainsi que les langages utilisés suivi de l'explication de l'intérêt de Firebase sans oublié les services web auxquels on a eu recours, ensuite nous avons présenté les principales interfaces de notre application mobile.

Chapitre 6

Conclusion générale

6.1 Introduction

La géolocalisation des taxis présente de nos jours un atout considérable pour les personnes ne voulant plus gaspiller leur temps en attendant la disponibilité du transport public, surtout aux moments des embûches, c'est aussi un avantage pour les personnes qui veulent bénéficier de l'utilisation des taxis à tout moment (24h/24 7j/7). C'est dans ce contexte qu'il nous a été confié de concevoir et de réaliser un système de géolocalisation et de réservation de taxi destiné aux périphériques mobiles Android.

Tout au long de ce mémoire, nous avons présenté les différentes technologies nécessaires pour proposer une approche basée sur la géolocalisation dynamique .Nous nous sommes intéressés à la technologie mobile et à son utilisation dans un environnement de géolocalisation dynamique.

Nous avons, à cet effet, essayé d'adopter les meilleures solutions, techniques et méthodes de développement.

Nous avons entamé notre mémoire avec la présentation du sujet, la problématique, les solutions existantes, ainsi que les objectifs de notre travail.

Ensuite, nous avons présenté les applications mobiles, leurs types et les systèmes d'exploitation existants, et nous nous sommes focalisés sur le système d'exploitation choisi qui est l'Android.

Nous avons ensuite enchaîné avec le chapitre de l'analyse et la conception de notre mémoire, nous avons choisis de modéliser notre système avec le formalisme UML suivant une démarche du processus unifié.

Par ailleurs, nous avons établi les spécifications des besoins fonctionnels à travers le diagramme de cas d'utilisation avec la description textuelle de chaque cas d'utilisation.

Une fois les fonctionnalités du système recensées, nous avons présenté la conception de notre application. Plus précisément, en utilisant le diagramme d'activité, les diagrammes de séquence, ainsi que le

diagramme de classe suivi du diagramme de déploiement qui décrit le déploiement physique sur des composants matériels des informations générées par le logiciel. La dernière étape a fait figure d'inventaire des outils utilisés pour la concrétisation de notre application, suivi d'une présentation des interfaces principales de notre application «Taxi06».

6.2 Contribution

A travers ce travail, nous avons d'abord essayé d'étudier et de comprendre les systèmes existants en matière de réservation de taxis puis nous avons tenté d'apporter un certain nombre d'améliorations, notamment sur les plans suivants :

- Facilité et simplicité d'utilisation de l'interface de l'application, que ce soit pour le client ou pour le chauffeur de taxi.
- Plusieurs moyens d'inscription par Facebook, Google ou par numéro de téléphone.
- Lors d'une demande de réservation, la source du client sera localisée automatiquement et dans le cas où le client décide d'indiquer un lieu de rendez-vous il l'a modifié
- Le chauffeur de taxi pourra accepter ou refuser une demande de réservation.
- Le chauffeur de taxi indique le prix s'il décide d'accepter une demande de réservation.
- Possibilité au client d'accepter une demande de réservation après avoir reçu le prix indiqué par le chauffeur qui a accepté la demande ou bien de l'annuler.
- Si le taxieur ne répond pas à une demande de réservation envoyée par le client au bout de 3 minutes, la demande de ce dernier sera expiré. De même pour le client, s'il ne confirme pas le prix indiqué par le taxieur.
- Possibilité de consulter un guide d'utilisation « Comment ça marche » qui est utile lors de la première utilisation de l'application.
- Le client ne peut pas envoyer une demande de réservation à un autre taxieur s'il en a déjà envoyé une.
- Possibilité de récupérer l'historique d'une course.
- Le client sera notifié lorsque le chauffeur de taxi arrive à destination en plus de suivre sa position sur la carte.
- un utilisateur bloqué sera sanctionné d'une durée de 6 mois.

Enfin, ce projet a fait l'objet d'une expérience très intéressante et instructive, il nous a été très bénéfique car il nous a permis de mettre en avant nos connaissances théoriques acquises durant notre cursus jusqu'alors et d'accroître nos compétences en modélisation UML, ainsi qu'en matière de programmation et de développement mobile, il nous a aussi permis pour la première fois de mettre en œuvre un mémoire sur la géolocalisation qui nous a été d'une richesse incommensurable. Bien évidemment nous avons appris à travailler en binôme et collaborer pour la réalisation d'une tâche commune.

6.3 Perspectives et travaux futurs

Ce mémoire constitue une base de travail à partir de laquelle, de nouvelles activités peuvent être lancées afin d'améliorer le travail présenté. Les perspectives que nous proposons peuvent donc s'orienter vers les directions suivantes :

- Améliorer l'application pour qu'elle soit compatible avec le système d'exploitation iOS.
- Ajouter une administration pour officialiser et fiabiliser d'autant plus notre application « Taxi06 » et pouvoir choisir les chauffeurs de taxis qui doivent bénéficier de l'utilisation de l'application.

Bibliographie

- [1] <https://fr.softonic.com/articles/uber-vtc-cest-quoi>, consulté le 10 novembre 2017.
- [2] Yasmine Taouint, "Tout sur Yassir, le clone algérien d'Uber", HuffPost Algérie, 8 novembre 2017.
- [3] <http://o-maroc.com/itaxi-contre-exemple-uber>, consulté le 10 novembre 2017.
- [4] www.ticmag.net/cardispo-premiere-application-web-et-mobile-de-reservation-de-taxi-au-cameroun/, consulté le 10 novembre 2017.
- [5] Badr Benmammar et Asma Amraoui, « Réseaux de radio cognitive : Allocation des ressources radio et accès dynamique au spectre », 2012.
- [6] www.taktilcommunication.com/blog/applications-mobile/definition-typologie-applications-mobiles.html, consulté le 13 novembre 2017.
- [7] [https://www.appmobile.paris/criteres-pour-une-bonne-application-mobile](http://www.appmobile.paris/criteres-pour-une-bonne-application-mobile), consulté le 15 novembre 2017.
- [8] blog.clever-age.com/fr/2010/04/22/comment-aborder-un-projet-de-mobilite/, consulté le 14 novembre 2017.
- [9] <http://www.latreebu.com/blog/application-mobile-native-ou-web-3-solutions/>, consulté le 30 novembre 2017.
- [10] [https://www.monpetitmobile.com/choisir-mobile/systemes-exploitation-smartphones](http://www.monpetitmobile.com/choisir-mobile/systemes-exploitation-smartphones), consulté le 30 novembre 2017.
- [11] GILLES BLANC, "Linux embarqué : comprendre, développer, réussir", PEARSON, 2011.
- [12] <http://www.futura-sciences.com/tech/definitions/> consulté le 15 novembre 2017
- [13] [https://socialcompare.com/fr/comparison/mobile-os-comparison-developer-view](http://socialcompare.com/fr/comparison/mobile-os-comparison-developer-view), consulté le 16 novembre 2017.
- [14] www.phonandroid.com/toute-l-histoire-et-la-chronologie-d-android-dossier.html, consulté le 16 novembre 2017.
- [15] <http://www.numerama.com/tech/132165-les-versions-dandroid-les-plus-utilisees.html>, consulté le 16 novembre 2017.
- [16] FREDERIC ESPIAU, « Créez des applications pour Android », Openclassroom, 2012.
- [17] JOSEF GABAY, DAVID GABAY, « UML2 Analyse et Conception », Université de Québec, 1^{re} édition, 2009.

- [18] PASCAL ROQUES, «Les cahiers du programmeurs UML2 modéliser une application web », EYROLLES, 4^e édition, 2008.
- [19] RUDI BRUCHEZ, « Les base de données NoSQL et le Big Data comprendre et mettre en œuvre », EYROLLES, 2^e édition, 2015.
- [20] LAURENT DEBRAUWER, « Design patterns en java », ENI, 3^e édition, 2013.
- [21] CYRILLE HERBY, « Apprenez à programmer en Java », OpenClassrooms - ex-Site du Zéro ,2^e édition, 2012.
- [22] DOUGLAS CROCKFORD, « JavaScript gardez pour le meilleur ! », PEARSON, 2009.
- [23] openclassrooms.com/courses/creez-un-backend-scalable-et-performant-sur-firebase, consulté le 20 Mai 2018
- [24] <https://developers.google.com>, consulté le 20 Mai 2018.
- [25] <https://github.com/firebase/geofire>, consulté le 20 Mai 2018.
- [26] <https://openclassrooms.com/courses/gerez-vos-codes-source-avec-git>, consulté le 21 Mai 2018
- [27] Dominic Wolf, « Get things done with Trello », CreateSpace Independent Publishing Platform, 2014.

RÉSUMÉ

Le monde entier fait face actuellement à un problème de transport dû à l'urbanisation. Depuis quelques années, nous vivons une forte concentration de la population en milieu urbain. C'est la raison pour laquelle, nous assistons à une augmentation de la demande du transport au niveau des villes. Donc, la question de transport représente un maillon fort dans le processus de développement économique et social d'une ville.

Aussi, le rôle du taxi a radicalement évolué dans la mobilité urbaine avec d'une part des besoins croissants pour des services personnalisés que les taxis sont les plus aptes à offrir, et d'autre part, la perspective de voir se desserrer les deux freins à leur développement : la disponibilité et le coût relativement élevé.

En outre, avec l'avènement du web mobile et la généralisation des smartphones, les applications mobiles sont devenues un point central de notre vie numérique. Elles se sont multipliées sur nos différents devices mobiles, et ont même tendance à devenir la norme sur desktop.

Notre travail consiste en « la conception et la réalisation d'une application mobile sous Android pour la réservation de taxis ». Il s'agit de mettre en place un système pour permettre à un client de rechercher un taxi de façon conviviale, facile et fiable.

Pour mettre en œuvre notre système, nous avons utilisé le processus Unifié (UP), qui se base sur l'UML comme langage de modélisation conçu pour fournir une méthode normalisée pour la conception.

L'application que nous avons réalisée utilise un ensemble de technologies d'actualité et elle est pourvue de fonctionnalités nécessaires et appropriées aux besoins des clients et des chauffeurs de taxi.

Mots clés : Application mobile, Android, NoSQL, Firebase, Service Web et XML.

ABSTRACT

The whole world is currently facing a transport problem due to urbanization. In recent years, we have a high concentration of the population in urban areas. That's why we are seeing an increase in the demand for transport at the city level. Therefore, the transportation issue represents a strong link in the process of economic and social development of a city.

Also, the role of the taxi has changed radically in urban mobility with, on one hand, growing needs for personalized services that taxis are best able to offer, and on the other hand, the prospect of loosening the two brakes to their development : availability and relatively high cost.

In addition, with the advent of the mobile web and the spread of smartphones, mobile apps have become a focal point of our digital life. They have multiplied on our different mobile devices, and even tend to become the standard on desktop.

Our work is "the design and implementation of an Android mobile application for booking taxis". It is about setting up a system to allow a customer to search for a taxi in a user-friendly, easy and reliable way.

To implement our system, we have used the Unified Process (UP), which is based on UML as a modeling language designed to provide a standardized method for design.

The application we have made uses a set of topical technologies and is equipped with the necessary features that are appropriate to the needs of customers and taxi drivers.

Key words : Mobile application, Android, NoSQL, Firebase, Web service and XML.