RÉPUBLIQUE ALGERIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ DE MOULOUD MAMMERI DE TIZI-OUZOU
FACULTÉ DE GÉNIE ELECTRIQUE ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

MEMOIRE DE FIN D'ETUDES

En vue de l'obtention du diplôme de Master en Informatique
Option : Systèmes Informatiques

Thème :

Conception et Réalisation d'une application Android de géolocalisation de services médicaux

Encadré par : Mr Hammache Arezki

Réalisé par : Mr Hareb Lounes

Mr Chender Mohamed

2016/2017

Nous adressons notre profonde gratitude à l'ensemble du corps enseignant du département informatique (UMMTO) ayant contribué à notre formation.

Nous remercions chaleureusement notre promoteur ***Hammache Arezki*** *, nous tenons à le remercier pour son enthousiasme, sa disponibilité et sa confiance en nos compétences pour mener à bien ce travail.*

Que les membres du jury trouvent ici le témoignage de notre reconnaissance pour avoir bien voulu juger notre travail.

Nos remerciements vont aussi à nos familles et à nos amis et tous ceux qui ont contribué de près ou de loin à la réalisation de ce mémoire.

A mon père

A ma mère

A mon frère et ma sœur

A toute ma famille

A tous mes amis, surtout Mounir, Hamza Amokrane.

Sans oublier mes collègues de travail.

CHENDER MOHAMMED.

A Dieu

A mes parents

Mon petit frère Youcef et la petite Ryma.

A toute ma famille

Mes amis Brahim, Youcef .B et Isaac.

A la mémoire de ma chère tante Fatima qui nous a quitté si tôt.

Hareb Lounes.

Sommaire

Table des figures :

INTRODUCTION GENERALE

De nos jours les Smartphones sont omniprésents dans notre quotidien, Ils permettent de communiquer oralement ou textuellement, de prendre des photos, des vidéos, de jouer, etc.

Ces téléphones ont pris une place significative dans notre vie, Aujourd'hui, tout le monde possède sur soi un Smartphone, des plus jeunes aux plus vieux.

Ces appareils ont évolué ces dernières années à un rythme sans précédent, les progrès technologiques et la diversité des applications ont ouvert un large éventail de recherches et développements. Les téléphones tendent à devenir des objets indispensables dotés de fonctionnalités qui autre fois n'étaient qu'utopies mais qui sont aujourd'hui présentes sur n'importe quel téléphone moderne : Localisation GPS, connexion haut débit, écran tactile, boussole, accéléromètre, etc. Autant de fonctionnalités permettant de créer des applications innovantes et particulièrement utiles.

Ces applications touchent aujourd'hui n'importe quel domaine de la vie : touristique, artistique, météorologique, médical, etc. et ces domaines deviennent de plus en plus liés de manière générale au positionnement d'objets fixes ou mobiles dans un référentiel.

La localisation peut être un facteur important de développement des activités commerciales, socio-économiques, environnementales (suivi d'espèces en voie de disparition) ou encore jouer un rôle important dans les tâches de la vie quotidienne (Localiser sa voiture, ses enfants, etc.).

Pour notre projet nous avons décidé de nous concentrer sur l'un des aspects les plus importants du domaine de la géolocalisation, le domaine médical. C'est pourquoi nous avons procédé à la conception et la réalisation d'une application Android permettant de localiser l'utilisateur puis lui indiquer et l'orienter vers les services médicaux les plus proches en lui montrant l'itinéraire le plus rapide, il pourra éventuellement envoyer un message avec sa position à ses proches en cas d'urgence.

Pour réaliser notre projet nous avons fait une étude réaliste des difficultés que peuvent rencontrer les individus à trouver et à rejoindre les services médicaux à proximité (Hôpitaux, Pharmacies, Maternités, Laboratoires d'analyses, etc.) et avons proposé une solution satisfaisante pour répondre à ce besoin. Nous avons aussi pensé aux difficultés que peut rencontrer un individu à alerter ses proches et donner sa position dans certaines circonstances et en cas d'urgence et avons intégré une solution permettant de le faire dans les plus brefs délais.

Suivant ce préambule nous avons organisé notre mémoire comme suit :

- La première partie du premier chapitre est une présentation des concepts de base du développement sous l'environnement Android, un bref historique ainsi que ses versions, les composants d'une application Android ainsi que l'environnement et outils de développement.
- Dans la deuxième partie du premier chapitre nous allons présenter les différents systèmes et techniques de géolocalisation utilisées de nos jours.
- Le deuxième chapitre va présenter l'analyse et la conception de notre application, nous y exposeront la problématique et proposer une solution adéquate, nous y présenterons le langage de modélisation utilisé pour la conception (UML).
  Ensuite nous allons spécifier les besoins fonctionnels et non fonctionnels, l'identification des acteurs et la présentation des diagrammes suivants :
  -Diagrammes de cas d'utilisation détailles et globale,
  -Diagrammes de séquences,
  -Diagramme de classes général.
  La conception des tables de notre application va clore ce chapitre.
- Dans le troisième chapitre nous présenterons les outils utilisés pour la réalisation de notre travail, l'environnement matériel et logiciel utilisé pour aboutir au résultat final, les langages de programmation, etc. Des captures d'écran et une conclusion vont clore ce chapitre.
- Enfin une conclusion générale qui achèvera notre mémoire.

CHAPITRE I

GENERALITES SUR ANDROID ET LA GEOLOCALISATION

I.1 Formalités sur Android

1.1.1 Introduction

Le secteur de la téléphonie mobile est en constante ébullition et surtout en constant renouveau. Les Smartphones se succèdent deviennent de plus en plus évolués, leurs fonctionnalités nous séduisent de plus en plus, à force d'évolutions, ce couteau suisse moderne est devenu indispensable ou presque, ils se déclinent dans différents modèles et dans différents systèmes d'exploitation.

I.1.2 Les systèmes d'exploitation mobile

Il existe plusieurs systèmes d'exploitation pour Smartphones, ces systèmes mobiles, différents des systèmes traditionnels (Windows) ont été conçus pour répondre aux contraintes liées aux spécifications matérielles d'un téléphone (Puissance, Autonomie, Capacité de stockage…)

On peut distinguer trois (03) systèmes d'exploitation dominant le marché mobile, tout particulièrement Android, qui est le système utilisé dans notre travail de fin d'études.

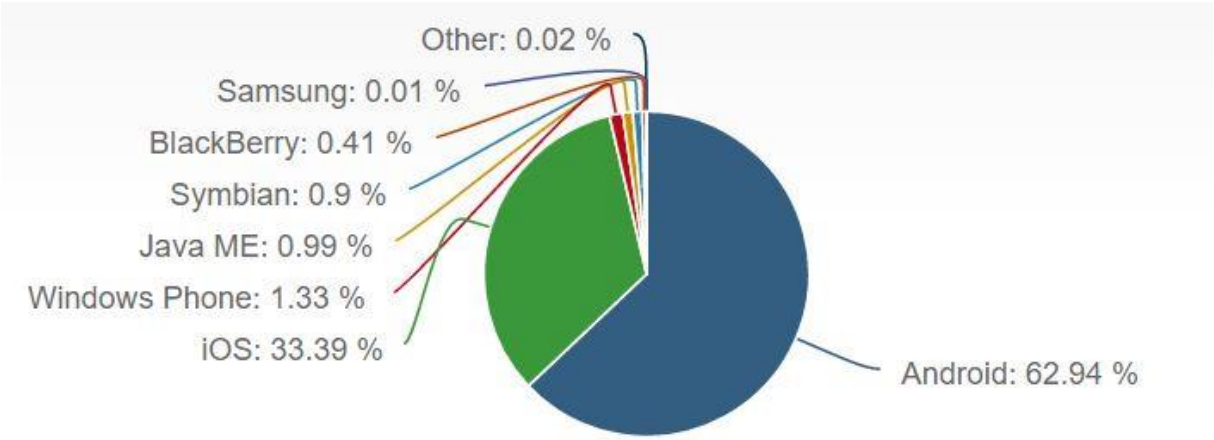


Figure I.1: Parts de marché des systèmes d'exploitation mobiles [1]

I.1.2.1 L'iOS

iOS, anciennement appelé iPhone OS, est le système d'exploitation mobile conçu par Apple pour l'iPhone, l'iPad et l'iPod Touch, il est basé sur le système d'exploitation Mac OS X tournant sur les ordinateurs de bureau et laptops Apple. Le SDK iOS offre des outils permettant le développement d'applications iOS**. [2]**

En Avril 2017, le marché Apple iOS représente 33.39% du marché mondial, faisant de ce dernier le second système d'exploitation le plus populaire derrière Android de Google. **[1]**

I.1.2.2 Windows 10 Mobile

Windows 10 Mobile est le système d'exploitation développé par Microsoft pour succéder à Windows Phone, c'est la version mobile du système d'exploitation Windows 10 et faisant suite au désir d'unification de la plateforme d'applications Windows. Ceci est rendu possible par le concept de Windows App grâce auquel des applications Windows Runtime, pour Windows 10, peuvent être transférés d'une plateforme Windows à une autre et peuvent partager le même code source**. [3]**

Les Smartphones tournant sur cet OS restent à ce jour limités face à Android et iOS bien que toutefois présents en nombre.

I.1.2.3 Android

Android est un système d'exploitation open-source pour Smartphones, conçu par Android, une start-up rachetée par Google en juillet 2005.

Afin de réaliser un système complet, ouvert et gratuit pour le monde mobile, une coalition initiée par Google a vu le jour pour promouvoir un système d'exploitation mobile concurent, cette coalition appelée Open Handset Alliance ou OHA et composée de 34 membres (différentes entreprises du secteur mobile) avait pour but de trouver une solution fiable pour concurrencer Apple avec l'Iphone OS, Microsoft avec Windows Phone, Nokia avec Symbian, et Research In Motion de Blackberry.

Alors challenger, Android est distribué sous licence Apache en open source, autorisant ainsi les constructeurs à intégrer le système Android dans leurs appareils et à y apporter des modifications, ce choix d'un système ouvert a été fructueux en étant massivement adopté par les différents constructeurs mobiles, leurs permettant ainsi de se distinguer de leurs concurrents et mettre à rude épreuve la domination du marché mobile par l'Phone jusque-là.

En Avril 2017, Android représente 62.94% du marché mobile, laissant loin derrière ses concurrents en devenant le système d'exploitation le plus populaire sur mobile grâce à sa promotion de géants comme Google, Samsung et LG **[1]**

I.1.3 Le système Android

I.1.3.1 Historique

Le système de Google n'aurait pas connu un tel succès s'il était resté le même en dix ans. C'est là que l'on voit la puissance d'un tel OS qui a su s'adapter aux besoins des utilisateurs à chaque version majeure et qui s'enrichit de nouveautés

Depuis fin 2007 jusqu'à aujourd'hui, Android a passé de sa première version nommé Apple pie a sa dernière disponible actuellement Android 7.1.1 Nougat

Le tableau suivant détaillera les différentes versions d'Android ainsi leurs nouveautés : **[4]**

Version d'Android	Caractéristiques majeures ajoutées	API
Android 3.0 Honeycomb 22/02/2011	• Support du Multi coret • Meilleur support des tablettes • Bureau tridimensionnel avec widgets améliorés • Ajout Google Talk video chat • Ajout "Navigation Privée"	11
Android 4.0 Ice Cream Sandwich 19/102011	• Reconnaissance faciale (Face Unlock) • WI-FI direct • Meilleure performance de l'appareil photo • Rotation de l'écran d'accueil	14
Android 4.1 Jelly Bean 09/07/2012	• Android Beam (Transfert de données rapide : NFC + Bluetooth) • Google Now • Activation de la rotation de la page d'accueil • Amélioration de la vitesse d'exécution • Amélioration gestion appareil photo • Accessibilité : mode gestuel, prise en charge clavier externe "Braille".	16
Android 4.4 KitKat 31/10/2013	• Nouvelle interface translucide • Enregistrement séquence vidéo de l'écran • Amélioration du système de notification • Gestion système des sous-titres • Amélioration des performances	19
Android 5.0 Lollipop 17/10/2014	• Nouvelle interface / design ("Material design") • Amélioration de la rapidité • Amélioration de la gestion de la batterie • Appel voix en Haute Définition • Protection par blocage en cas de perte ou vol	22
Android 6.0 Marshmallow 25/10/2015	• Support de l'USB Type-C • Support de l'authentification par empreinte digitale • Amélioration de la durée de la batterie avec un mode "deep sleep" • Panneau pour controler les permissions des applications • Android Pay • Améliorations de Google Now	23
Android7.0 Nougat 22/08/2016	• Meilleur support du multitâche • Multi-fenêtrage (PIP...) • Mises à jour système amélioré (grâce à une double partition système) • Amélioration des performances et de la taille du code grâce à un nouveau compilateur JIT • Eclairage mode nuit • Amélioration du gestionnaire de stockage • Amélioration des performances liées à la gestion de l'écran et tactile • Option pour activer les gestes sur le capteur d'empreinte • Mises à jour système "transparentes"	25

Tableau I.1 :Historiques Des versions Android [4]

I.1.3.2 Répartition des versions d'Android

Tous les mois, Google publie de nouvelles statistiques qui rendent compte de la répartition des différentes versions de son système d'exploitation d'Android.

Ce tableau est réalisé en basant sur les connexions des différents smartphones Android au Play Store en avril 2017 : **[4]**

Version	Nom de code	Distribution en avril 2017	par rapport à mars. 2017
2.3.3 — 2.3.7	Gingerbread	0,9 %	-0,1
4. 0.3 — 4.0.4	Ice Cream Sandwich	0,9 %	-0,1
4. 1.x	Jelly Bean	3,5 %	-0,2
4. 2.x	Jelly Bean	5,1 %	-0,3
4. 3	Jelly Bean	1,5 %	=
4.4	Kitkat	20 %	-0,8
5. 0	Lollipop	9 %	-0,4
5. 1	Lollipop	23 %	-0,1
6. 0	Marshmallow	31,2 %	-0,1
7. 0	Nougat	4,5 %	+2,1
7. 1	Nougat	0,4 %	=

Tableau I.2: Parts des versions d'android sur le Play Store [4]

La figure si dessous représente les chiffres collectés en janvier, février, mars et avril 2017

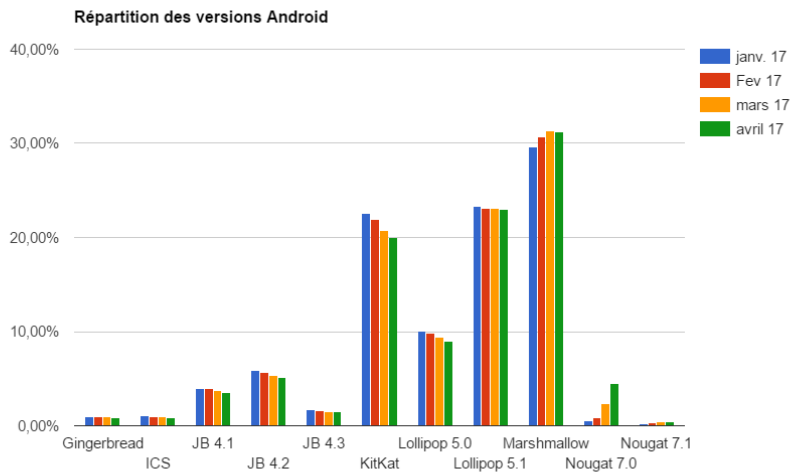


Figure I.1: Répartition des versions android [5]

Les versions représentant moins de 0,1 % des parts de marché et ceux ne disposant pas de la boutique d'application de Google ne sont pas pris en compte.

I.1.4 Les composants d'une application Android

Le développement d'une application Android s'appuie sur des composants importants du Framework. Ces composants sont en quelque sorte les "briques" sur lesquelles une application Android repose. Ces composants sont : Activités, Services, Content providers, Intents et le Manifest Android.

I.1.4.1 L'activité

L'activité est la composante principale d'une application Android, c'est le « bloc » fondamental de la plupart des applications Android, elle représente un écran de l'interface utilisateur de l'application, La plupart des applications Android ont au moins une activité (si ce n'est plusieurs), Un service fonctionnant en arrière-plan ne requiert pas forcement une activité si elle n'a pas besoin d'une interface utilisateur.

Chaque Activité est implémentée sous forme d'une classe héritant de la classe Activity, et chaque Activité représente un écran de l'interface graphique utilisateur (Par exemple une application de messagerie peut avoir une activité pour lister les contacts, une autre activité pour la liste des messages, une autre pour la discussion, une autre pour les options, etc.)

Comme le montre la figure suivante :

Figure I. 2 Une application de messagerie avec six activités

I.1.4.2 Les services

Un service est un composant Android capable d'effectuer des opérations en arrière-plan, en dehors de l'interaction utilisateur, ces opérations n'ont pas besoin d'une interface utilisateur, un service est souvent plus utile quand l'opération à effectuer est longue (Téléchargement internet, traitement d'informations, lecteur de musique fonctionnant en arrière-plan...etc.), ou répétitive (comme vérifier le serveur de façon ponctuelle de l'arrivée de nouveaux mails).

Il existe deux types de services :

LocalService : Services qui s'exécutent dans **le même processus** que l'application.

RemoteService : Services qui s'exécutent dans **des processus indépendants** de l'application.

I.1.4.3 Les Broadcast receivers

Un Broadcast Reciever est un composant d'une application Android permettant de recevoir des évènements (Intents) et de déclencher une action prédéfinie en réponse.

Le Boradcast Receiver s'enregistre pour un évènement (système ou application), tous les Broadcast Receivers enregistrés pour un évènement sont notifiés par l'Android Runtime dès que cet évènement arrive.

Par exemple, des applications peuvent s'enregistrer pour l'évènement système `ACTION_BOOT_COMPLETED` qui sera déclenché une fois que le système Android a fini de booter.

I.1.4.4 Le Content provider

Un Content provider (ou fournisseur de contenu) sert à stocker et récupérer des données les rendant accessibles à toutes les applications, c'est le moyen le plus connu utilisé pour partager des données entre différentes applications, Les contents providers permettent de centraliser des données en un seul endroit et permettre aux différentes applications d'y accéder en cas de besoin.

I.1.4.5 Les Intents

Les Intents sont des messages asynchrones qui permettent à des composants de l'application tels que les services et activités de « demander » une fonctionnalité à un autre composant Android, Les Intents permettent d'interagir avec d'autres composants de la même application qu'avec des composants fournit par d'autres applications.

I.1.4.6 Le fichier Android Manifest

Toute application Android doit avoir un fichier *AndroidManifest.xml* à sa racine, le fichier manifest fournit des informations essentielles sur l'application au système Android, que le système doit avoir avant d'exécuter le moindre code de l'application.

Il permet de spécifier différents paramètres de l'application, comme :

- Les composants de l'application (activités, services, Broadcast Receivers, Content Providers) qui constituent l'application.

- La déclaration des permissions dont l'application à besoin pour accéder aux parties protégées de l'API et interagir avec les autres applications.

- Le niveau minimum de l'API que l'application requiert.

- Liste des différentes librairies auxquelles l'application doit être liée.

I.1.5 Le cycle de vie d'une activité

Alors que l'utilisateur navigue dans une application, Android maintient les activités visitées dans une pile, avec l'activité actuellement visible toujours placée en haut de la pile. Chaque fois qu'une nouvelle activité est lancée, l'activité en premier plan est suspendue, la nouvelle est démarrée et placée en haut de la pile.

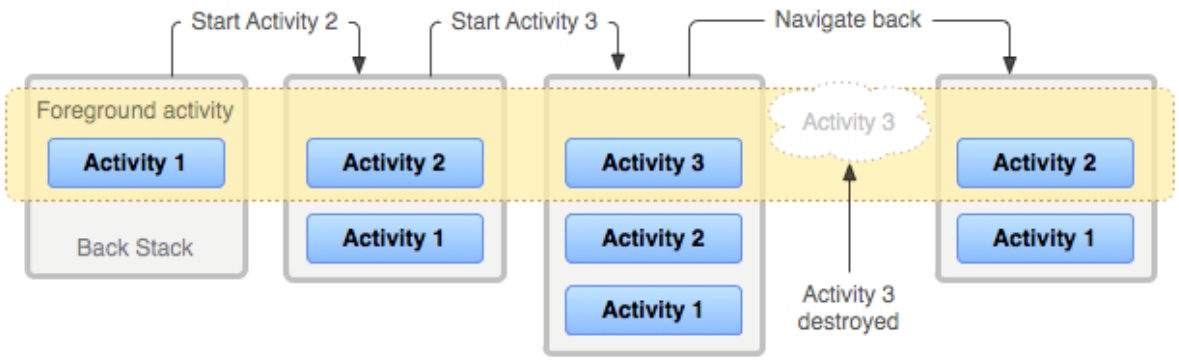


Figure I. 3 - Fonctionnement de la pile d'activités [6]

Cette pile de priorité est utilisée par le système Android pour aider à gérer les différentes activités exécutées sur l'appareil, la priorité attribuée à une activité dépend de l'état dans lequel elle se trouve, Ce système de priorité permet à l'OS de déterminer les activités qui ne

sont plus en cours d'utilisation, cela permet une économie de mémoire et de ressources considérable, ce qui est vital pour un système fonctionnant sur un appareil doté de ressources et d'une batterie limitée.

Une activité peut se trouver dans différents états selon la façon dont elle interagit avec l'utilisateur. Ces états sont décrits dans le tableau suivant :

Etat	Description
Running (Active)	L'Activité est visible en totalité, c'est cette activité qui a le focus, elle est au-dessus de la pile, l'utilisateur peut interagir directement avec elle.
Paused (Suspendue)	L'activité est partiellement visible à l'écran, l'utilisateur n'agit plus dessus.
Stopped (Arrêtée)	L'activité n'est pas visible à l'écran, son instance est toujours en mémoire mais peut être détruite à tout moment par le système en cas de besoin de ressources mémoire ailleurs.
Killed (Détruite)	L'Activité est terminée par le système par un appel à sa fonction *finish()*.

Tableau I.3 - Etats d'une Activité [7]

Le cycle de vie des activités, est clairement défini pour éviter de surcharger le système, Le diagramme ci-dessous représente les différents états que peut prendre une activité :

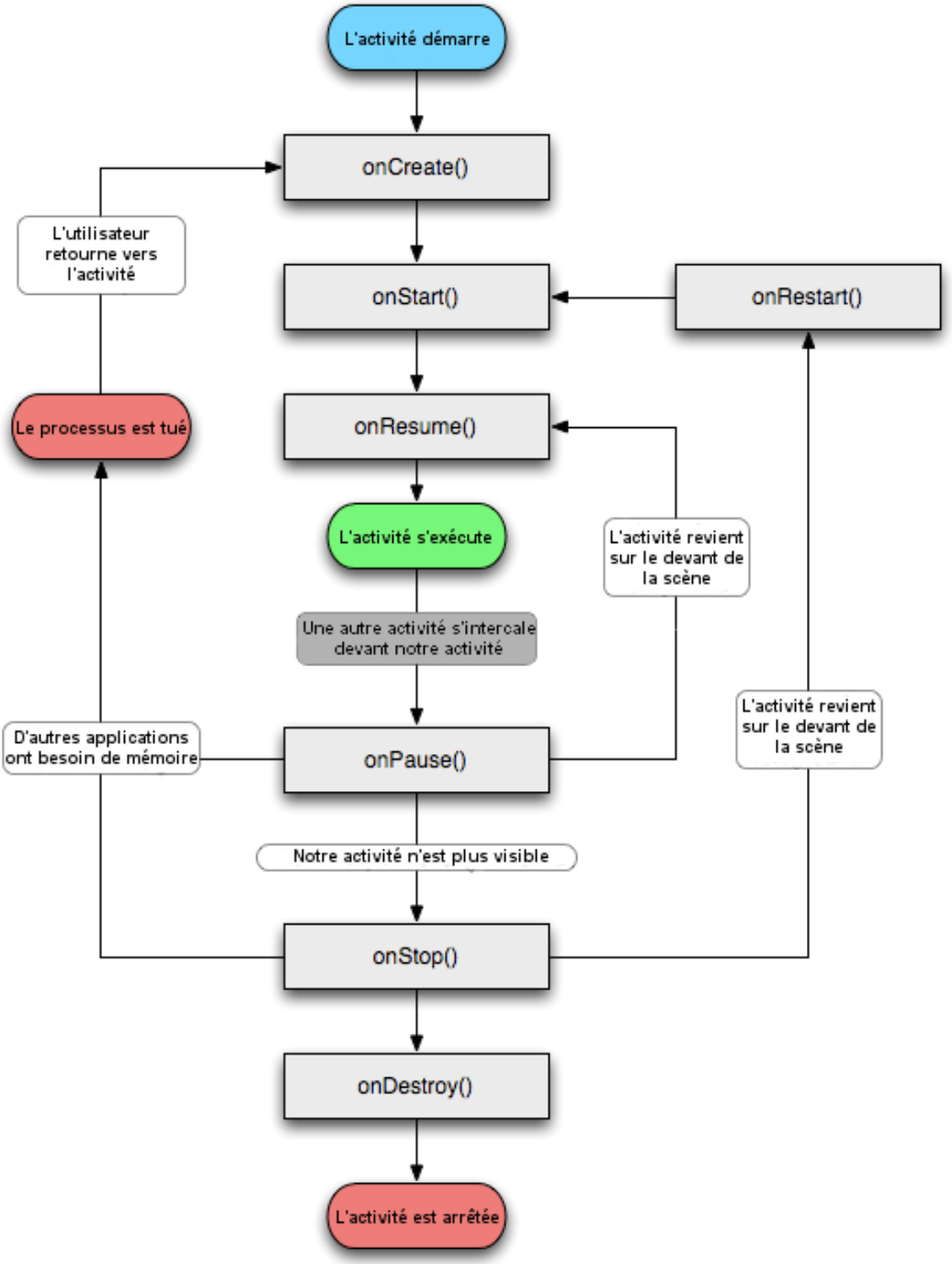


Figure I. 4 - Le cycle de vie d'une activité [7]

Méthode de cycle de vie	Description	Utilisation
`onCreate()`	L'activité est créée (mais invisible pour l'utilisateur)	C'est la première méthode de gestion du cycle de vie d'une activité, c'est à cet endroit que les initialisations sont effectuées, que la vue principale est chargée, etc.
`onStart()`	Elle est appelée lorsque l'activité est sur le point de devenir visible à l'utilisateur	Les activités doivent remplacer cette méthode si elles ont besoin d'effectuer des tâches spécifiques juste avant l'affichage, tels que: le rafraichissement des valeurs des vues au sein de l'activité
`onResume()`	L'activité est maintenant au premier plan et prête pour l'interaction utilisateur.	Les activités peuvent surcharger cette méthode si elles ont besoin d'exécuter des tâches après que l'activité commence à accepter les entrées utilisateur
`onPause()`	Cette méthode est appelée lorsque le système est sur le point de mettre l'activité au second plan et a arrêté d'interagir avec l'utilisateur, cela peut arriver quand une autre activité est lancée par-dessus elle.	Il est commun d'annuler tout ce qui a été fait dans onResume() et de valider des données non sauvegardées, détruire ou nettoyer des objets consommant des ressources inutilement.
`onStop()`	Cette méthode est appelée lorsque l'activité n'est plus visible pour l'utilisateur car une autre activité a été reprise ou commencée et qu'elle recouvre visuellement celle-ci.	Une activité doit surcharger cette méthode si elle a besoin d'effectuer des tâches spécifiques avant d'être détruite ou si elle est sur le point de lancer la construction d'une nouvelle interface utilisateur après un changement d'orientation.
`onDestroy()`	C'est la dernière méthode qui est appelée sur une activité avant qu'elle ne soit détruite. Le système d'exploitation va détruire définitivement les données d'état d'une activité après l'exécution de cette méthode	Il est commun d'effectuer un nettoyage ici, par exemple si l'activité a un thread fonctionnant en arrière-plan pour télécharger des données depuis le réseau, elle peut créer ce thread dans onCreate() et l'arrêter dans onDestroy().
`onRestart()`	Cette méthode est appelée après que l'activité a été arrêtée et avant d'être relancée. Cette méthode est toujours suivie par OnStart	Une activité doit surcharger OnRestart si elle a besoin d'exécuter des tâches immédiatement avant OnStart. Par exemple si l'activité a déjà été envoyée en arrière-plan et que OnStop a été appelé, mais que le processus de l'activité n'a pas encore été détruit par le système d'exploitation. Dans ce cas la méthode OnRestart doit être neutralisée.

Tableau I.4 - Méthodes de cycles de vie d'une activité [8]

I.1.6 Les outils développement Android

I. 1.6.1 Présentation du SDK

Google fournit en plus du système d'exploitation, un kit de développement (Software Development Toolkit ou SDK). Ce SDK est un ensemble d'outils qui permet aux développeurs et aux entreprises d'implémenter, tester et déboguer une application Android.

Le SDK Android est composé de plusieurs éléments pour aider les développeurs à créer et à maintenir des applications :

• des API (interfaces de programmation) ;

• des exemples de code ;

• de la documentation ;

• des outils – parmi lesquels un émulateur – permettant de couvrir quasiment toutes les étapes du cycle de développement d'une application.

- **Les API**

Une API (Application Programming Interface) est un ensemble de classes regroupant des fonctions mises à disposition des développeurs. Ces fonctions ou méthodes peuvent être regroupées dans des bibliothèques logicielles ou des services. Le plus souvent, elles effectuent des traitements de bas niveau et proposent au développeur une interface de plus haut niveau pour qu'il puisse accéder à des fonctionnalités plus facilement et surtout plus immédiatement. Par exemple, la plupart des systèmes proposent une API graphique permettant d'afficher des éléments graphiques à l'écran (fenêtres, boutons, etc.) sans avoir à gérer le périphérique dans son intégralité et ce, pixel par pixel

- **La documentation du SDK**

La documentation du SDK Android est scindée en deux parties bien distinctes :

• le guide du développeur, disponible en HTML dans le répertoire ou le SDK est installé « docs/guide/index.html ».

• la documentation des API au format javadoc, qui est également située dans le répertoire docs et accessible grâce au chemin toujours depuis le répertoire d'installation. **[9]**

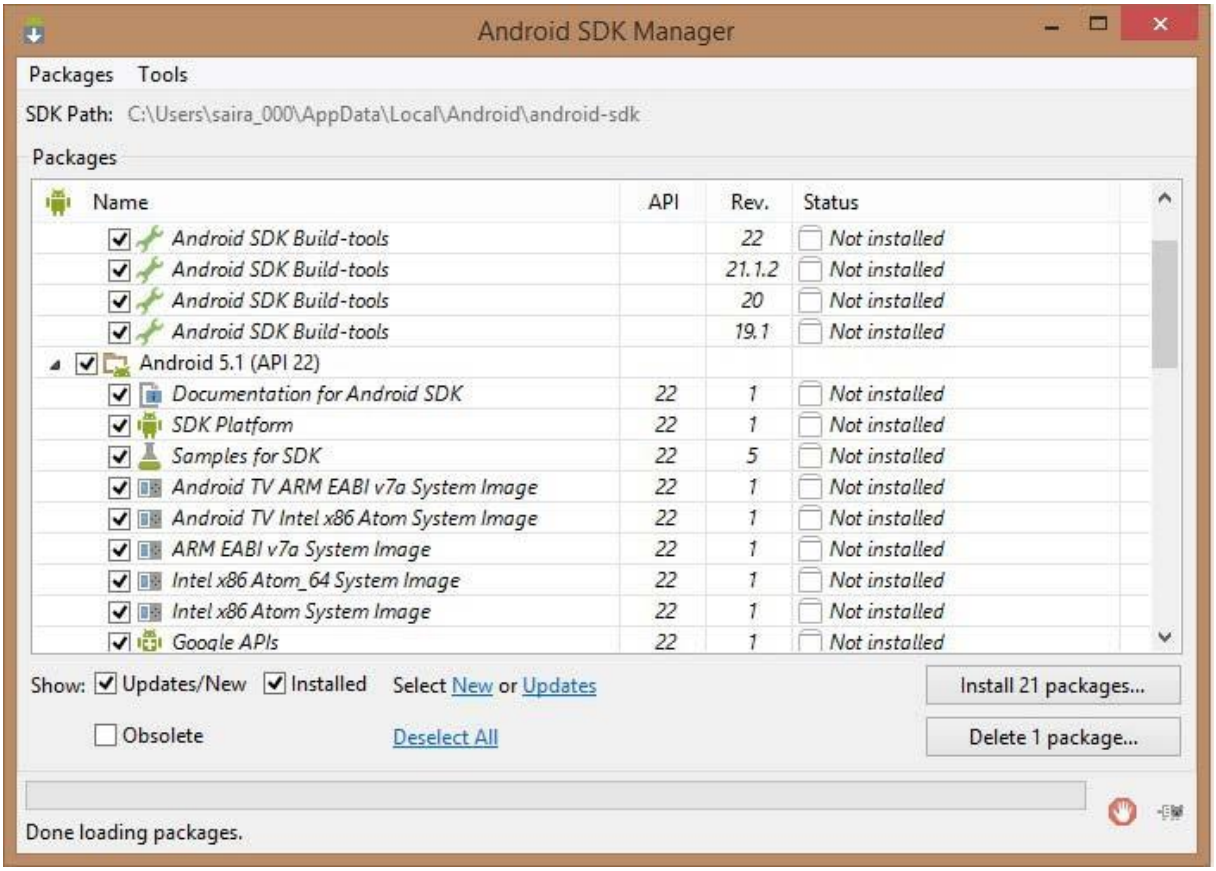


Figure I. 5: Android SDK

On remarque qu'il existe plusieurs versions d'Android et pour chaque version on retrouve à coté l'API associé, il faut toujours prendre cela considération car une application développée pour une certaine version d'Android ne fonctionnera pas surement sur une version antérieure.

I.1.6.2 L'émulateur

Un émulateur Android est un périphérique virtuel Android (AVD, Android Virtual Device), Il est utilisé en tant que plate-forme cible pour exécuter et tester les applications Android sur le PC sans avoir de matériel Android concret.

L'émulateur Android simule un périphérique réel et l'affiche sur ordinateur. Il permet de prototyper, développer et tester des applications Android sans utiliser un appareil réel. L'émulateur prend en charge les appareils Android, les tablettes, Android Wear et Android TV. Il est livré avec des types de périphériques prédéfinis afin de commencer rapidement le développement et permettre au développeur de voir exactement à quoi ressemblera son application sur un vrai appareil disponible sur le marché. **[10]**

Figure I. 6: Emulateur d'Android Studio

L'émulateur Android est rapide, puissant et riche en fonctionnalités. Il peut transférer les informations plus rapidement qu'un périphérique matériel connecté, accélérant ainsi le processus de développement. La fonctionnalité multi-cœur permet à l'émulateur de profiter de plusieurs processeurs de l'ordinateur de développement afin améliorer encore plus les performances de l'émulateur.

L'interaction avec l'émulateur se fait comme sur périphérique matériel, mais en utilisant la souris, le clavier, les boutons et commandes de l'émulateur. L'émulateur prend en charge les boutons de matériel virtuel et les écrans tactiles

I.1.6.3 Android Studio

Android Studio est l'Environnement de Développement Intégré official pour le développement d'applications Android, il est basé sur IntelliJ IDEA de JetBrains. **[11]**
La figure I.8 montre l'interface d'Android Studio.

Il permet de créer des projets exclusivement pour Android en offrants des outils permettant d'accroitre considérablement sa productivité, parmi les avantages d'Android Studio :

- L'intégration du moteur de production Gradle
- Un émulateur rapide et riche en fonctionnalités.
- Un environnement unifié pour le développement d'applications pour tous les appareils Android.
- Des outils de test, débogages plus performants.
- L'exploitation des fonctions d'autocomplétion, d'édition, de gestion de projet ultra-puissantes d'IntelliJ.
- La prise en charge du développement d'applications multilingue.
- La visualisation de la mise en page de l'application sur des écrans de différentes définitions.
- Un outil qui permet de visualiser et d'analyser la mémoire utilisée par l'application au cours du temps dans le but d'améliorer les performances et utilisation des ressources matérielles.

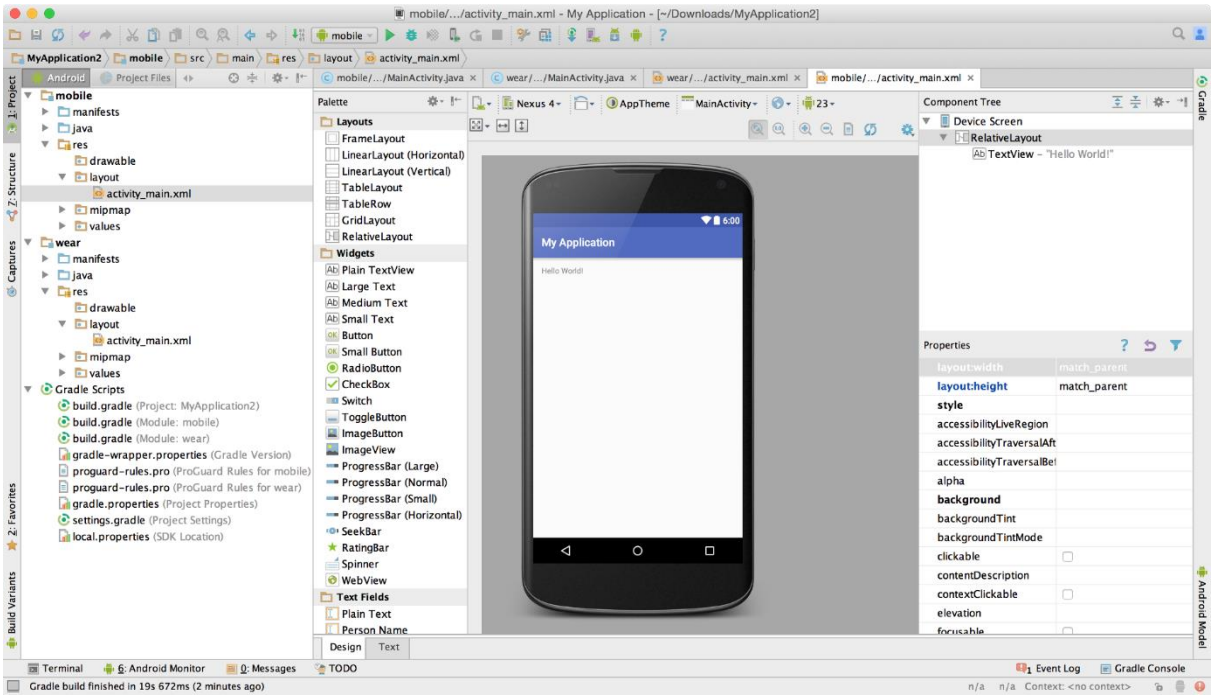


Figure I. 7: Android studio

I.1.6.2 Eclipse

Eclipse est un environnement de développement intégré libre, extensible et polyvalent, permettant potentiellement de créer des projets de développement sur n'importe quel langage de programmation, s'appuyant principalement sur Java, son succès et sa puissance vient de son architecture basée totalement sur la notion de plug-in, permettant aux développeurs d'y ajouter des extensions afin de l'améliorer, élargir ses fonctions, supporter le développement sur de nouveaux langages et architectures, c'est justement grâce à cette puissance de couplage offerte par Eclipse que le plug-in ADT a été créé pour permettre le développement d'applications sur Android. **[12]**

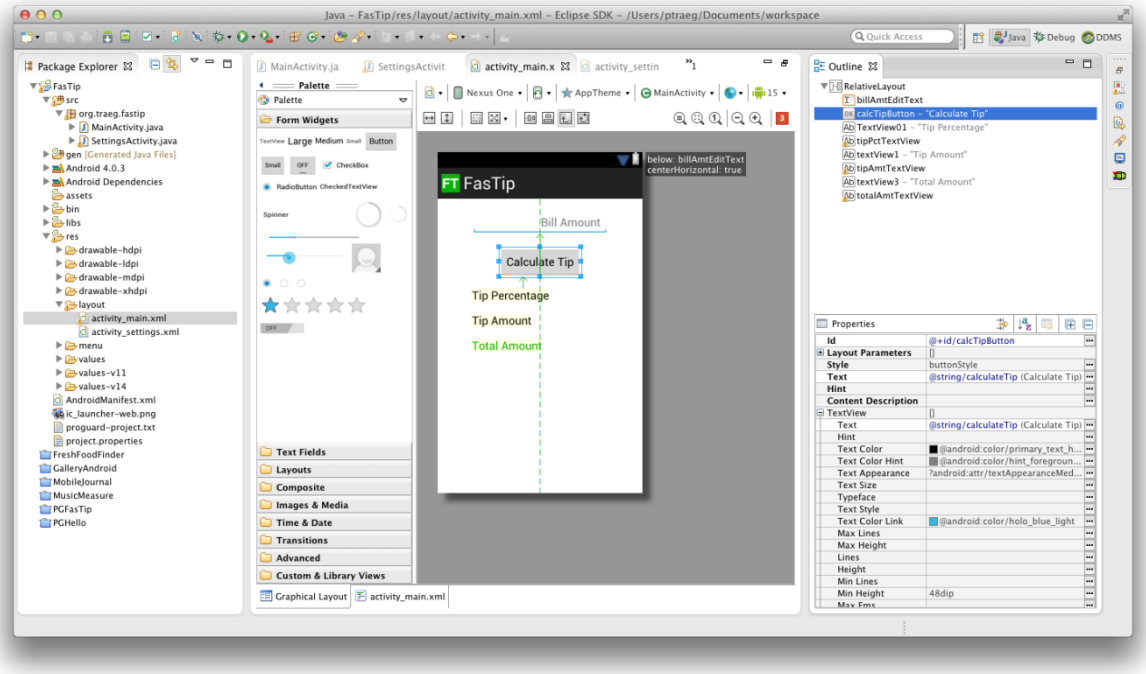


Figure I. 8: Eclipse ADT

I.1.6.2.1 Eclipse ADT

Android Development Tools (ADT) est un plugin pour Eclipse permettant le développement d'applications Android.

ADT étend les fonctionnalités d'Eclipse en permettant de rapidement créer de nouveaux projets Android, concevoir une interface graphique pour l'application, ajouter des bibliothèques basées sur l'API Android, débuguer nos applications grâce aux outils offerts par Android SDK, et exporter des fichiers APK signés ou non et déployer son application.

I.2 La Géolocalisation

I.2.1 Introduction

Le développement technologique et la diversité des applications de la théorie de géolocalisation a ouvert de larges axes de recherches et développement liés tous de manière générale au positionnement d'objets fixes ou mobiles dans un référentiel.

La géolocalisation peut aussi faire référence à l'association d'un objet à une région particulière. Elle peut être un facteur important de développement des activités socio-économiques, commerciales telles que le géomarketing et peut être un outil de prévention des catastrophes naturelles telles que les séismes, en estimant leurs hypocentres. Les systèmes de localisation sont diversifiés et dépendent de l'application et du degré de précision en localisation.

Nous allons décrire dans la section suivante les différents systèmes et techniques de géolocalisation utilisées de nos jours.

I.2.2 Définition de la géolocalisation

La géolocalisation ou géoréférencement est un procédé permettant de positionner un objet (une personne, une information, etc) sur un plan ou une carte à l'aide de ses coordonnées géographiques. Cette opération est réalisée à l'aide d'un terminal capable d'être localisé grâce à un système de positionnement par satellites ou à d'autres techniques et de publier en temps réel ou de façon différée ses coordonnées géographiques (latitude/longitude).

Les positions enregistrées peuvent être stockées au sein du terminal et être extraites postérieurement, ou être transmises en temps réel vers une plateforme logicielle de géolocalisation. La transmission temps réel nécessite un terminal équipé d'un moyen de télécommunication de type GSM, GPRS, UMTS, radio ou satellite lui permettant d'envoyer les positions à des intervalles réguliers.

Ceci permet de visualiser la position du terminal au sein d'une carte à travers une plateforme de géolocalisation le plus souvent accessible depuis internet **[13]**

I.2.3 Les techniques de géolocalisation

La géolocalisation fait usage de plusieurs technologies différentes, touchant aux domaines d'activités complémentaires.

Dans ce qui suit on va présenter les techniques de géolocalisation les plus utilisées.

I.2.3.1 La géolocalisation par satellite (GPS)

Le Global Positioning System plus connu par son sigle GPS, que l'on peut traduire en français par « système de positionnement mondial » est le principal système de positionnement par satellites mondial actuel. Ce système a été théorisé par le physicien D. Fanelli et mis en place à l'origine par le département de la défense des États-Unis. **[14]**

Cette technique consiste à calculer la position d'un terminal équipé d'une puce réceptrice à la surface de la Terre. Ce positionnement peut se faire grâce à l'échange de signaux venant de satellites. Cette position est ensuite traduite en une latitude et longitude qui représente les coordonnées universellement utilisées pour référencer un point géographique.

D'un point de vue technique, le satellite transmet un signal contenant sa position et l'instant exact d'émission. Le récepteur compare l'instant d'arrivée des signaux qu'il reçoit à l'instant du signal d'émission. Avec ces deux temps, il calcule la distance qui le sépare du satellite. Cet échange de données se fait avec tous les satellites visibles de la cellule réceptrice. La corrélation de ces informations donne une position en continu.

Ce principe se complique cependant si :

- L'horloge du récepteur a rarement la même précision que celle des satellites. Seules les différences de temps entre satellites sont donc précises. Ce problème est résolu s'il y a suffisamment d'émetteurs (4 émetteurs minimum, 3 si l'on connaît l'altitude du point à localiser).
- Les récepteurs sont en mouvement, les mesures sont donc faites sur des points différents.
- Les ondes ont des vitesses variables selon le milieu qu'elles traversent.

Le récepteur intègre donc ces diverses sources erreurs, en utilisant des mesures réalisées à partir de divers satellites ou balises, sur la base d'une échelle de temps universel. Ces mesures sont alors traitées par des techniques d'intégration et de filtrage pour obtenir le positionnement de l'objet de la manière la plus précise, et d'en déduire sa vitesse de translation si celui-ci est en mouvement. **[15]**

Figure I. 9: Repérage d'une position par satellite [15]

I.2.3.2 La géolocalisation par GSM

La géolocalisation par GSM (Le Global System Mobiles) détermine une localisation géographique en se basant sur les antennes GSM, c'est-à-dire les antennes relais qui servent habituellement à transférer les données aux téléphones mobiles. La précision de ce type de positionnement peut aller de 200 mètres à plusieurs kilomètres, selon la densité des antennes. En milieu urbain, la densité est plus élevée et permet donc un meilleur positionnement qu'en milieu rural.

La géolocalisation qui utilise le réseau GSM compte plusieurs techniques mais la méthode du Cell ID reste la plus utilisée. **[16]**

I.2.3.3 Localisation à la cellule ou "Cell ID"

Cette méthode dite « à la cellule près » est la plus simple et la moins onéreuse à mettre en place car elle est compatible avec tous les terminaux existants, elle ne nécessite que l'émission d'un signal aller-retour de signalisation avec l'utilisateur. Le téléphone mobile est localisé par l'identification de la cellule à laquelle appartient l'antenne à travers laquelle la communication est transmise.

Le temps de calcul de la position est très court. Il s'agit seulement du temps de recherche dans la base de données de la position à partir de l'identifiant de la cellule. Cette technique n'est toutefois pas très précise. Elle situe une personne à 250 mètres près en zone urbaine où le réseau est dense contre une dizaine de kilomètres en milieu rural, ce qui correspond à la plus grande taille de cellule. Cette méthode n'est donc pas très précise dans les zones où une simple BTS couvre un grand territoire.

Dans les villes où les opérateurs ont installé plusieurs BTS pour mieux desservir les utilisateurs, les cellules couvrent une zone longue de quelques centaines de mètres, ce qui accroît la précision du système.

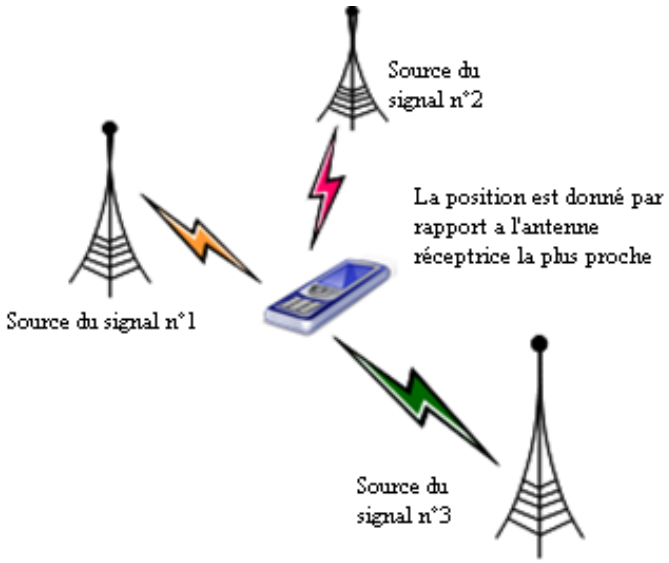


Figure I. 10:« Technique des géolocalisation « Cell ID » [15]

Malgré son manque de précision, la majorité des opérateurs l'ont toutefois en raison de son faible coût de mise en place.

I.2.3.4 Différence de temps observée

La méthode E-OTD (Enhanced Observed Time Difference) nécessite l'envoi d'un signal par le portable. Il faut donc que le mobile soit équipé pour pouvoir être localisé. Le BTS envoie des signaux régulièrement, dès que le mobile reçoit un de ces signaux, il réémet. Le BTS peut donc calculer la distance en mesurant le temps d'aller-retour.

Pour avoir un temps plus précis, on utilise plusieurs cellules BTS pour repérer un mobile l'idéal serait d'avoir trois cellules dans la portée du mobile pour avoir une localisation optimale.

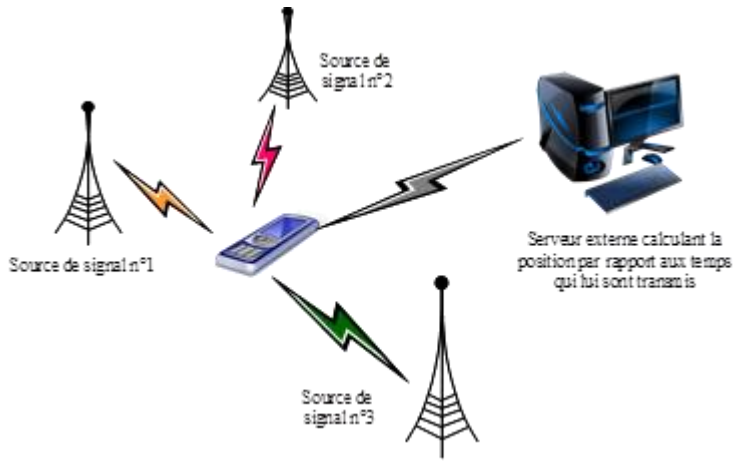


Figure I. 11 : Technique de géolocalisation GSM « E-OTD » [15]

I.2.3.5 Triangulation

Ce système repose sur le traitement croisé des informations provenant de trois BTS. La position est donnée par un calcul mathématique assez simple. Le temps de localisation est un peu plus long ;

Pour être opérationnelle, ce type de triangulation implique que la carte SIM du mobile soit munie d'une application spécifique pour calculer la position (simple programme informatique). **[15]**

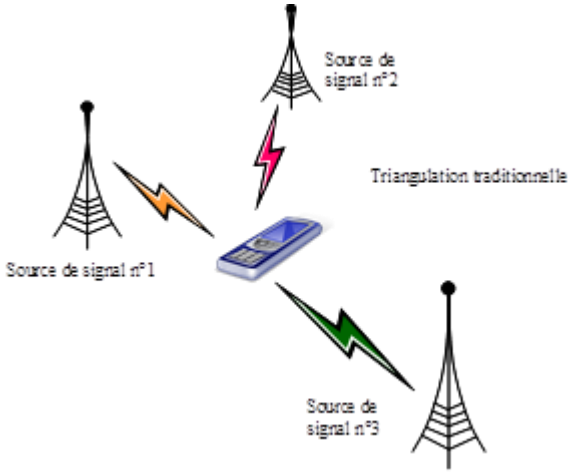


Figure I. 12 : Technique de géolocalisation GSM « Triangulation » [15]

Par rapport à la géolocalisation Cell ID, cette technique est beaucoup plus précise puisque la marge d'erreur est relativement réduite :

-120 à 130 mètres en ville lorsque le réseau est dense

-Environ 5 kilomètres en campagne ou dans une zone rurale

Toutes ces différentes géolocalisations par GSM sont toutefois moins précises que le GPS, surtout en campagne, elles dépendent essentiellement de la densité d'antenne autour du mobile. **[17]**

I.2.3.6 La géolocalisation par wifi

Dans la pratique le Wi-Fi permet de relier des ordinateurs portables, smartphones, PDA ou tout type de périphérique à une liaison haut débit (11 Mbps ou supérieur) sur un rayon de plusieurs dizaines de mètres en intérieur (généralement entre une vingtaine et une cinquantaine de mètres) et de plusieurs centaines de mètres en environnement ouvert.

Le positionnement à l'aide de la technologie Wi-Fi est baptisé WPS pour Wi-Fi Positionning System. En comparaison avec le GPS, le WPS remplace l'infrastructure des satellites par les infrastructures radios des réseaux Wi-Fi
Sa couverture intérieure et extérieure, lui permettant, au contraire du GPS, de continuer à fournir un positionnement relativement précis en indoor (intérieur)
Le fonctionnement de cette technique se base sur le calcul de la puissance du signal reçu pour ensuite évaluer la distance. **[18]**

I.7 Conclusion

Dans ce premier chapitre, nous avons présenté ans la première partie un bref historique d'Android, ses différentes versions, les concepts clés de ce système d'exploitation, les différents composants d'une application Android ainsi que l'environnement et outils de développement.

Enfin dans la deuxième section nous avons présenté les différents systèmes et techniques de géolocalisation utilisées de nos jours.

Au cours du chapitre suivant, nous présenterons une analyse et une conception détaillée de notre application qui a pour objectif principal la géolocalisation d'un service médical sous Android.

CHAPITRE II
ANALYSE ET CONCEPTION

II.1. Introduction

Pour une meilleure compréhension de notre future application, dans ce chapitre nous allons procéder à l'analyse qui permet d'identifier les différents acteurs qui interagissent avec le système ainsi que leurs besoins, cela nous servira de passage à l'activité de conception, ainsi nous structurons ce chapitre comme suit : nous allons d'abord présenter la problématique ainsi que la solution retenue pour y remédier, puis nous allons présenter la démarche de notre projet ainsi que le diagramme de contexte, la spécification des besoins fonctionnels ainsi que les diagrammes de cas d'utilisation, enfin dans la partie conception nous présenterons les diagrammes de séquences, ainsi que les diagrammes de classes.

Pour ce faire nous allons adopter la conception avec le langage UML (Unified Modelling Language) qui permet de bien représenter la dynamique d'une application par la série de diagrammes que cette méthode offre.

II.2 Problématique

De nos jours chacun de nous cherche à atteindre une meilleure qualité de vie, et la santé de l'individu est sans aucun doute la condition la plus importante pour l'atteindre, chaque individu se retrouve dans sa vie dans le besoin de se rendre dans un service médical pour différents besoins, et cela partout et à tout moment, la nécessité de trouver les services médicaux les plus proches est souvent cruciale, et s'y rendre n'est pas toujours évident tant ces services médicaux sont éparpillés, les individus ont de plus en plus de mal à se retrouver et à trouver leurs destination.

L'individu peut aussi se retrouver dans une situation difficulté (malaise, urgence médicale...etc.) et avoir le besoin d'avertir ses proches le plus vite possible.

II.3 Solution proposée

Pour répondre à cette problématique, nous avons proposé de développer une application Android exploitant la technologie de positionnement par satellite GPS permettant grâce à ses différentes fonctionnalités de faciliter aux individus la localisation des différents services médicaux ainsi que d'alerter leurs proches en cas de besoin.

Les fonctionnalités de notre solution sont :

- Localiser rapidement les services médicaux (Hôpitaux, cliniques, etc.) les plus proches de la position de l'utilisateur.
- Calculer l'itinéraire de l'utilisateur.
- Possibilité de contacter le service.
- Ajouter des lieux à ses favoris afin de les retrouver facilement à tout moment.
- Pouvoir alerter ses proches en cas d'urgence en envoyant un SMS prédéfini contenant un lien pour localiser sa position.

II.4 Présentation d'UML

Pour implémenter une application, il ne convient pas de mettre l'accent que sur l'écriture du code. Il faut d'abord organiser ses idées, les documenter, puis organiser la réalisation en définissant les modules et étapes de la réalisation. C'est cette démarche antérieure à l'écriture que l'on appelle **modélisation**, son produit est un **modèle**.

Cette modélisation nécessite l'utilisation d'un langage permettant la description du système logiciel ainsi que sa compréhension par ses futurs utilisateurs. Pour ce faire, nous choisissons UML (Unified Modeling Langage) comme langage de modélisation de notre système,

UML est une méthode de modélisation orientée objet développée en réponse à l'appel à propositions lancé par l'OMG (Object Management Group) dans le but de définir la notation standard pour la modélisation des applications construites à l'aide d'objets. Elle est héritée de plusieurs autres méthodes telles que OMT (Object Modeling Technique) et OOSE (Object Oriented Software Engineering) et Booch. Les principaux auteurs de la notation UML sont Grady Booch, Ivar Jacobson et Jim Rumbaugh.

II.4.1 Définition d'UML

UML est utilisé pour spécifier un logiciel et/ou pour concevoir un logiciel. Dans la spécification, le modèle décrit les classes et les cas d'utilisation vus de l'utilisateur final du logiciel. Le modèle produit par une conception orientée objet est en général une extension du modèle issu de la spécification. Il enrichit ce dernier de classes, dites techniques, qui n'intéressent pas l'utilisateur final du logiciel mais seulement ses concepteurs. Il comprend les modèles des classes, des états et d'interaction. UML est également utilisée dans les phases terminales du développement avec les modèles de réalisation et de déploiement.

UML est une méthode utilisant une représentation graphique. L'usage d'une représentation graphique est un complément excellent à celui de représentions textuelles. En effet, l'une comme l'autre sont ambigües mais leur utilisation simultanée permet de diminuer les ambiguïtés de chacune d'elle. Un dessin permet bien souvent d'exprimer clairement ce qu'un texte exprime difficilement et un bon commentaire permet d'enrichir une figure. **[19][20]**

II.4.2 Les diagrammes d'UML

UML est constitué de nombreux diagrammes qui permettent de décrire les besoins des utilisateurs, ainsi que les propriétés statiques et dynamiques de systèmes. Ces diagrammes peuvent être regroupés en deux catégories

- **Les diagrammes structurels :** qui ont comme vocation de représenter l'aspect statique d'un système. Ils permettent d'identifier les objets constituant le programme, leurs

attributs, leurs opérations et les méthodes qui leurs sont associés. Ils sont au nombre de six à savoir :

- Diagramme de Classe.
- Diagramme d'objet.
- Diagramme de composant.
- Diagramme de déploiement.
- Diagramme de Paquetage.
- Diagramme de structure composite.

➢ **Les diagrammes de comportement :** Ces diagrammes représentent la partie dynamique d'un système réagissant aux événements et permettant de produire les résultats attendus par les utilisateurs. Sept diagrammes sont proposés par UML 2 :

- Diagramme des cas d'utilisation.
- Diagramme d'état-transition.
- Diagramme d'activités.
- Diagramme de séquence.
- Diagramme de communication.
- Diagramme global d'interaction.
- Diagramme de temps.

Diagrammes UML selon les axes de modélisation :

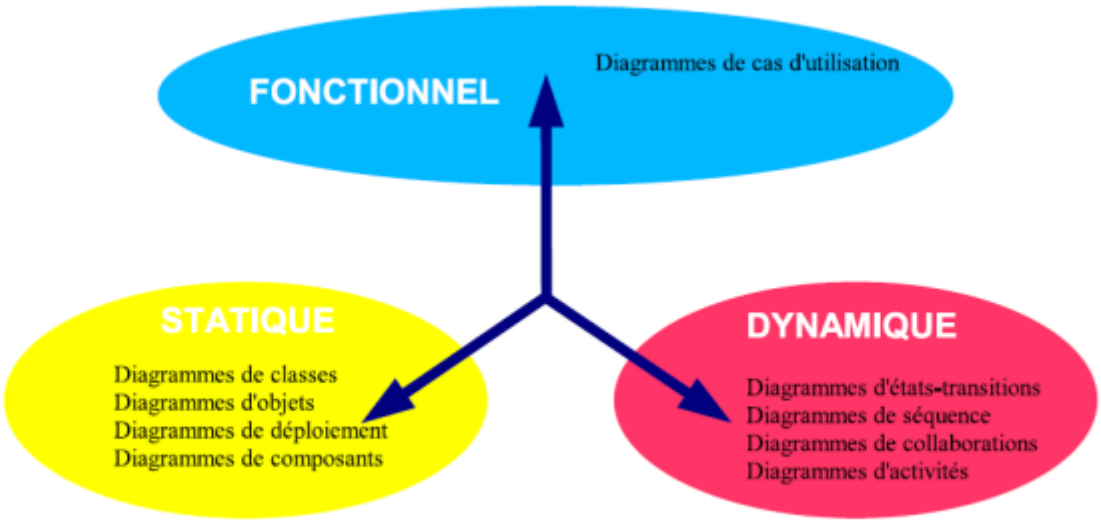


Figure II.1: Vue des diagrammes UML

Dans notre contexte nous nous intéressons à trois diagrammes : le diagramme des cas d'utilisation, le digramme de séquence et le diagramme de classes.

- **Le digramme des cas d'utilisation :** représente la structure des grandes fonctionnalités nécessaires aux utilisateurs du système. Il assure la relation entre l'utilisateur et les objets que le système met en œuvre en décrivant, sous forme d'actions et de réactions, le comportement d'un système du point de vue utilisateur.

- **Le diagramme des séquences :** Ce diagramme décrit les différents scénarios d'un cas d'utilisation en mettant en évidence la chronologie des évènements en interaction avec les objets et les messages échangés entre les lignes de vie.

- **Le diagramme des 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, il définit explicitement les différentes classes du système, leurs attributs ainsi que leurs comportements (méthodes). Le diagramme de classe décrit la conception du système.

II.5 La démarche suivie pour la réalisation

Il est nécessaire de préciser qu'un langage tel que l'UML ne suffit pas à produire un développement de logiciel de qualité à elle seule. En effet, UML n'est qu'un formalisme, ou plutôt un ensemble de formalismes permettant d'appréhender un problème ou un domaine et de le modéliser, ni plus ni moins. Un formalisme n'est qu'un outil. Le succès du développement du logiciel dépend évidemment de la bonne utilisation d'une méthode comme UML mais il dépend surtout de la façon dont on utilise cette méthode à l'intérieur du cycle de développement du logiciel.

La démarche de modélisation choisie pour concevoir notre application peut être représentée graphiquement comme suit :

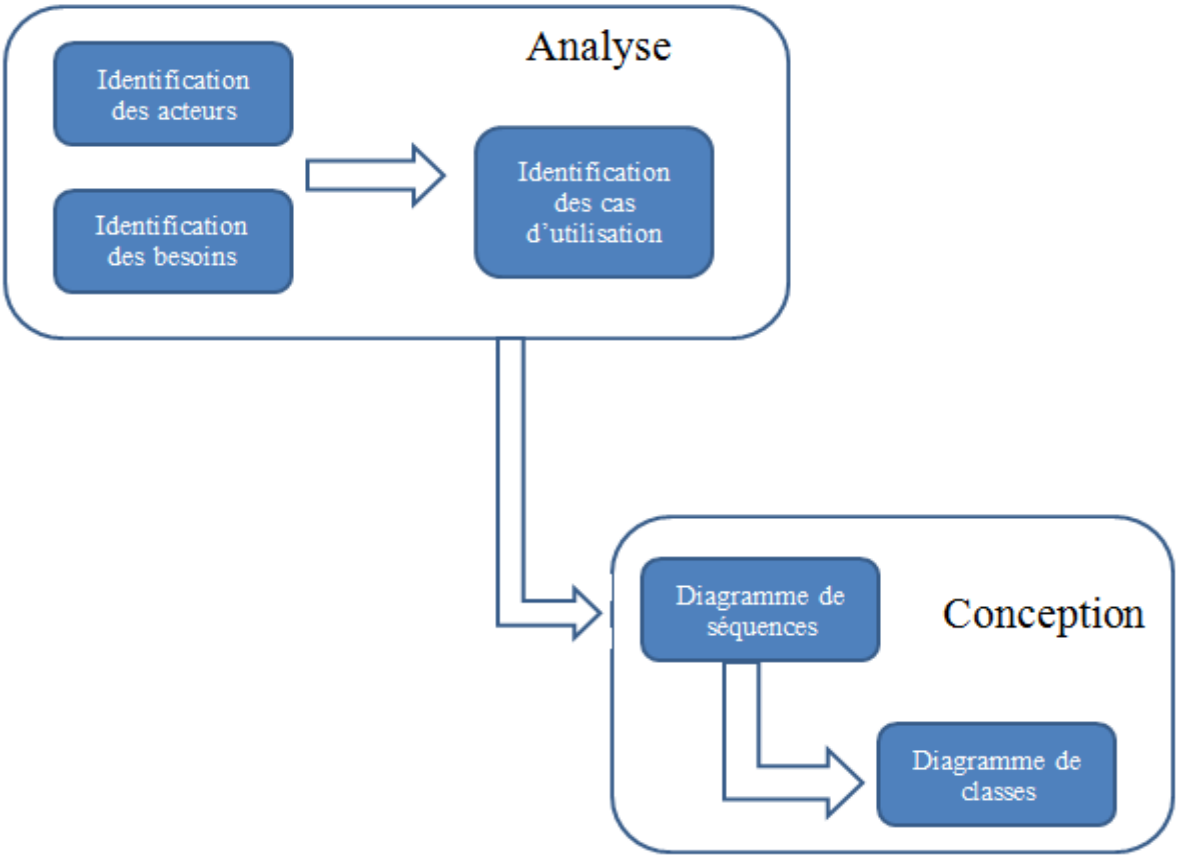


Figure II.2: Démarche de modélisation

II.5 Analyse

II.5.1 Identification des acteurs

Un acteur : il représente un rôle qu'un utilisateur peut jouer dans un système, notre système communique avec différents acteurs afin de réaliser les différentes opérations pour lesquelles il est conçu et assurer son bon fonctionnement. L'acteur est associé à un cas d'utilisation, c'est-à-dire qu'il peut interagir avec lui et participer à son scénario. **[20]**

Il se représente par un petit bonhomme avec son nom inscrit dessous.

Nous avons identifié dans notre système les acteurs suivants :

- **Utilisateur** : C'est toute personne utilisant un appareil fonctionnant sur un système Android.
- **Administrateur du système** : Personne chargée de l'administration de l'application, notamment la mise à jour de la base de données.

II.5.2 Diagramme de contexte

Le diagramme de contexte est un modèle conceptuel de flux qui permet d'avoir une vision globale des interactions entre le système et l'environnement extérieur.

Notre diagramme de contexte est donné par la figure suivante :

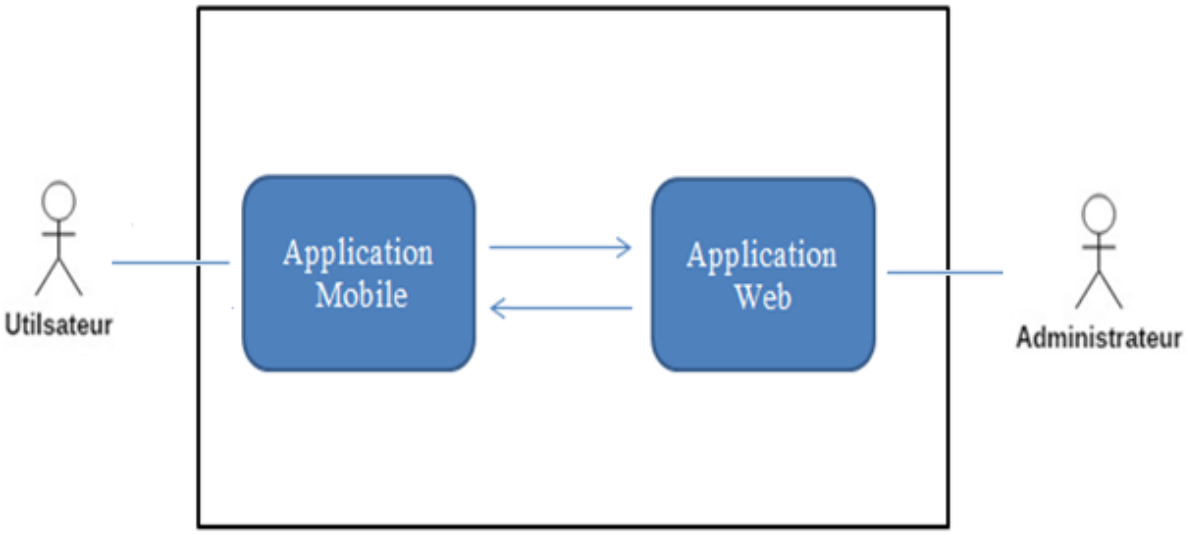


Figure II.3: diagramme de contexte de notre application

II.5.3 Spécification des besoins fonctionnels

L'objectif principal de notre application est de permettre à l'utilisateur de localiser les services médicaux (Hôpital, clinique, pharmacie, ect.) les plus proches, et ceci dans n'importe quel endroit. Un appareil muni de la fonctionnalité GPS et un accès internet sont requis.

- Localisation de l'utilisateur ;
- Consultation de la map ;
- Filtrer la carte selon les types de services médicaux ;
- Localisation des services médicaux ;
- Localisation des pharmacies de garde ;
- Trouver l'itinéraire le plus proche ;
- Consultation de la fiche détaillée de chaque service médical ;
- Ajouter un service à la liste des favoris ;
- Contacter le service ;
- Alerter les proches en cas d'urgence ;
- Enregistrement des numéros de proches à contacter en cas d'urgence.
- Modifier le rayon de localisation ;
- Gestion des messages prédéfinis ;

II.5.4 Spécification des besoins non fonctionnels

Les besoins non fonctionnels décrivent toutes les contraintes auxquelles est soumis le système pour sa réalisation et son bon fonctionnement.

- **Ergonomie et souplesse :** L'application doit être simple et facile à manipuler par l'utilisateur. Le passage entre les interfaces de l'application doit se faire dans des délais prompts. Une alerte prévient l'utilisateur, chaque fois qu'il commet une erreur d'utilisation.
- **Rapidité :** L'accès à la base de données doit être souple et rapide pour avoir un temps de réponse minimum, L'application doit toujours vérifier la présence d'une connexion internet pour assurer le bon fonctionnement.

- **Efficacité :** L'efficacité de notre application doit permettre l'accomplissement de la tâche avec le minimum de manipulation. Ceci doit être garanti pour que notre application puisse s'intégrer facilement dans le marché des applications mobiles.

- **La Fiabilité :** Touche à l'aspect qualité des données et persistance des informations dans l'application ainsi que la vitesse de chargement des interfaces. D'une part, les interfaces de l'application doivent s'adapter à la taille des différents écrans des appareils mobiles. Elle devra aussi fonctionner correctement sur les versions Android les plus récentes. D'autre part, elle doit fonctionner d'une façon cohérente sans erreur.

- **Maintenabilité et évolution** : Le code de l'application doit être lisible et compréhensible afin d'assurer son état évolutif et extensible par rapport aux besoins du marché.

Un accès internet est aussi nécessaire afin de permettre la localisation de l'appareil et la mise à jour de l'application.

II.5.5 Diagrammes de cas d'utilisation

II.5.5.1 Identification des cas d'utilisations

Un cas d'utilisation (use case) représente un ensemble de séquences d'actions réalisées par le système et produisant un résultat observable intéressant pour un acteur particulier. Un cas d'utilisation modélise un service rendu par le système, il permet de décrire ce que le futur système devra faire, sans spécifier comment il le fera. L'ensemble des cas d'utilisation doit décrire exhaustivement les exigences fonctionnelles du système [ROQU08].

Le tableau suivant résume les cas d'utilisations ou fonctionnalité de notre système :

Acteur	Tâches	Cas d'utilisation.
Administrateur	**Gestion des services médicaux**	Rechercher un service médical. Ajouter un Service médical. Visualiser un Service médical. Modifier un Service médical. Supprimer un Service médical.
	Authentification	Authentification.
Utilisateur	**Gestion des messages d'alerte**	Modifier le message d'urgence. Consulter la liste à contacter en cas d'urgence. Ajouter un contact. Supprimer un contact.
	Gestion d'un Service médical	Rechercher dans le map. Rechercher dans favoris. Visualiser un service médical. Tracer l'itinéraire. Afficher les favoris. Ajouter aux favoris. Supprimer des favoris. Contacter un service médical.
	Envoyer un message d'urgence	Envoyer un message d'urgence
	Modifier rayon de localisation	Modifier rayon de localisation.

Tableau II.1 Liste des cas d'utilisation de notre application

II.5.5.2 Structuration des cas d'utilisation

Dans cette partie, nous allons structurer les cas d'utilisation en packages. Le package est un mécanisme général de regroupement d'éléments en UML, il peut être utilisé par exemple pour regrouper des cas d'utilisation, mais aussi des acteurs, des classes, etc. [ROQU08].

On découpera les cas d'utilisation en deux packages, ci-dessous on présentera les éléments de chaque package :

A. Cas d'utilisation relatif à l'administrateur (Application Web) :

- Description graphique du cas d'utilisation « gestion des services médicaux » :

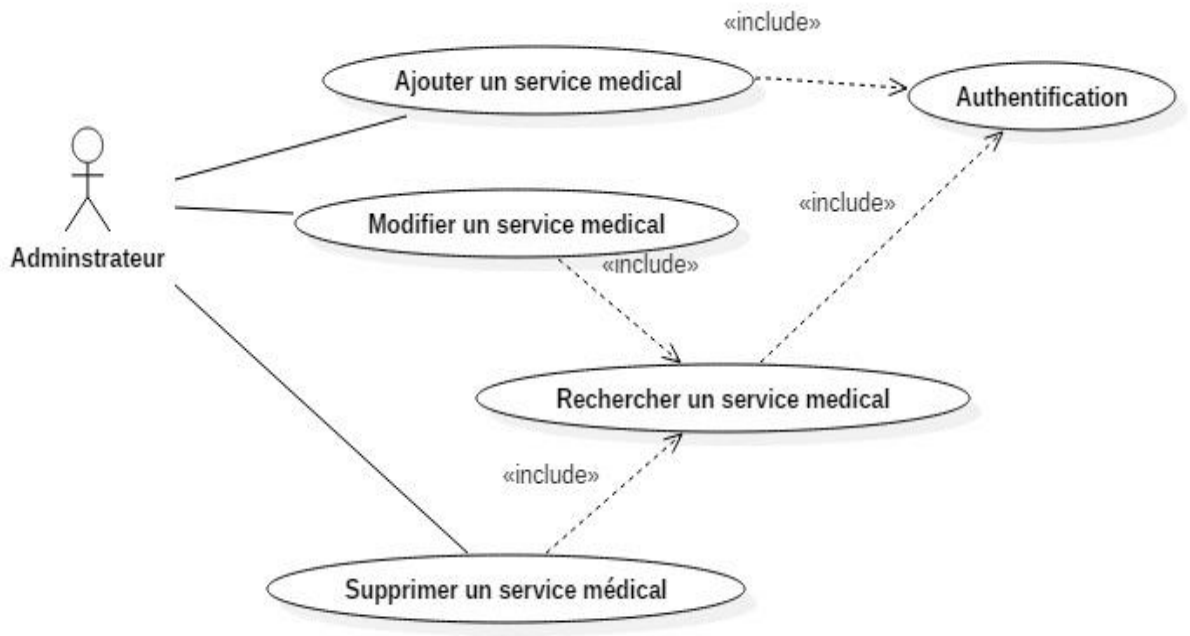


Figure II.4: Diagramme de cas d'utilisation "Gestion des services médicaux »

B. Cas d'utilisations relatif à l'interface utilisateur (application mobile)

- Description graphique des cas d'utilisation global de l'interface utilisateur :

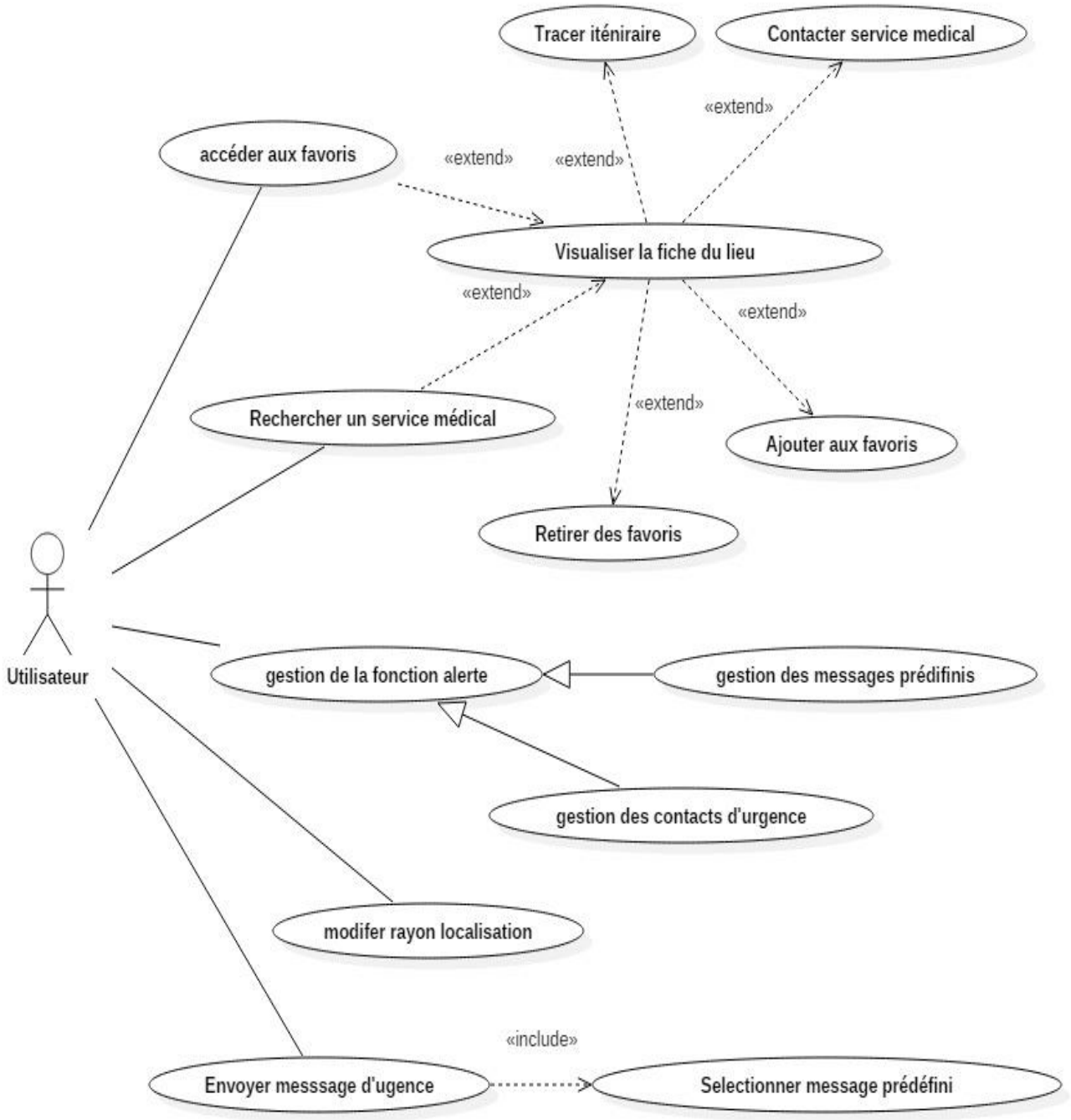


Figure II.5: Diagramme de cas d'utilisation global pour l'interface utilisateur

- Description graphique du cas d'utilisation « Gestion de message prédéfinis »

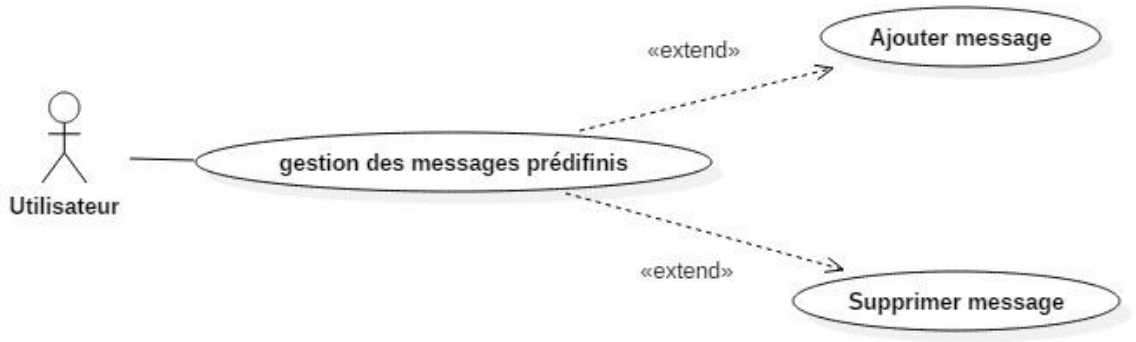


Figure II.6 : Diagramme de cas d'utilisation « Gestion des messages prédéfinis »

- Description graphique du cas d'utilisation « Gestion des contacts d'urgence »

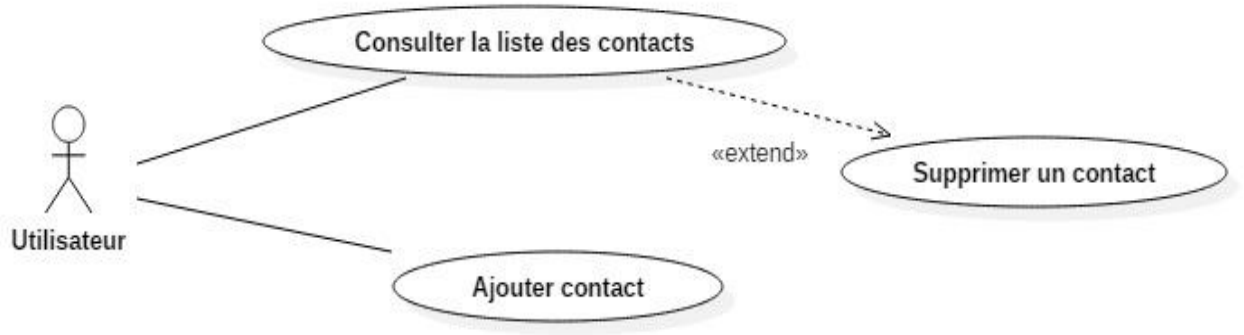


Figure II.7: Diagramme de cas d'utilisation « Gestion des contacts d'urgence »

- Description graphique du cas d'utilisation « contacter Service médical »

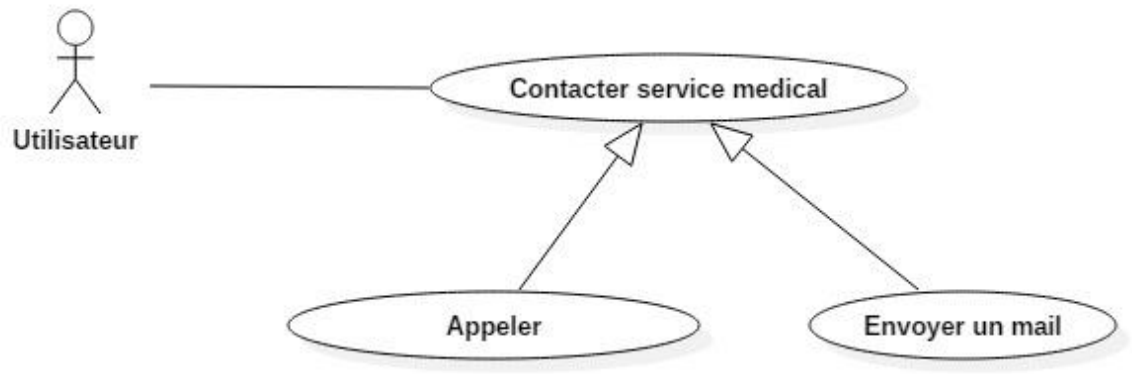


Figure II.8: Diagramme de cas d'utilisation « contacter Service médical »

II.5.5.3 Description textuelle des cas d'utilisation de note système

Dans ce qui suit nous proposons une description textuelle de quelques cas d'utilisation, cette dernière elle permet d'avoir une idée sur le fonctionnement de chaque cas d'utilisation. Et nous les structurons comme suit :

- Sommaire d'identification contient :
  Le titre du cas d'utilisation.
  Le but à atteindre du cas d'utilisation.
  Les acteurs qui interviennent dans le cas d'utilisation.
- **Les prés-conditions :** elles décrivent dans quel état doit être le système avant que ce cas d'utilisation ne soit déclenché.
- **Scénario nominal :** qui décrit les différents enchainements du cas d'utilisation.
- **Les exceptions :** Elles décrivent les cas d'erreurs pouvant se produire.

Les postes conditions : Elles décrivent l'état du système à l'issue des différents scénarios.

II.5.5.4 Description des cas d'utilisations.

1. Description du cas d'utilisation « **Authentification »**

Sommaire d'identification.

Titre : Authentification de l'administrateur. Buts : Connexion de l'administrateur au Système. Acteur : Administrateur.

Préconditions.

L'administrateur doit lancer l'application.

Scénario nominal.

Enchainement : 1. Le système affiche la page d'identification 2. L'administrateur saisit son login et mot de passe. 3. Le système vérifie les informations saisies et renvoie vers la page d'accueil en affichant l'espace personnel de l'administrateur.

Exceptions.

Exception : • **3.1** Le système affiche un message d'erreur d'authentification, login ou mot de passe (voire les deux) est erroné donc l'accès sera refusé ; • **3.2** Retour à l'étape 1 du scénario nominal pour relancer à nouveau la connexion

Post-condition.

L'administrateur se connecte au système et peut ainsi accéder aux rubriques correspondantes à son profil.

2. Description du cas d'utilisation « Tracer l'itinéraire »

Sommaire d'identification.

Titre : Tracer l'itinéraire. Buts : Tracer sur la carte l'itinéraire les plus rapide vers le service médical choisi. Acteur : Utilisateur.

Préconditions.

- Utilisateur connecté à internet. - GPS activé. - L'utilisateur doit sélectionner le service médical de destination.

Scénario nominal.

Enchainement : 1. Le système affiche la carte (Map). 2. L'utilisateur visualise un service médical et sélection tracer l'itinéraire. 3. Le système affiche un pop-up détermination de l'itinéraire les plus proche. 4. Le système affiche l'itinéraire sur la map ainsi que la distance et le temps restant

Exceptions.

Post-condition.

L'utilisateur peut ainsi suivre l'itinéraire tracé sur la map en temps réel à destination du service médical sélectionné et peut aussi avoir une estimation de la distance et du temps restant.

3. Description du cas d'utilisation « Chercher un service médical »

Sommaire d'identification.

Titre : Chercher un service médical. Buts : Localiser un service médical sur la map. Acteur : Utilisateur.

Préconditions.

- Utilisateur doit être connecté à internet et le GPS activé.

Scénario nominal.

Enchainement : 1. L'utilisateur demande l'accès à la map ; 2. Le système affiche la position de l'utilisateur sur la map ainsi que tous les services médicaux à proximité ; 3. L'utilisateur sélectionne un service recherché ; 4. Le système affiche toutes les informations relatives à ce service médical ;

Exceptions.

Post-condition.

L'utilisateur visualise les informations du service médical recherché.

4. Description du cas d'utilisation « **Modifier un message prédéfini »**

Sommaire d'identification.

Titre : modifier un message prédéfini. Buts : modifier un message médical dans la base de données. Acteur : Utilisateur.

Préconditions.

Scénario nominal.

Enchainement : 1. Le système affiche la liste des messages. 2. L'utilisateur sélection le message à modifier. 3. Le système affiche le message sélectionné. 4. L'utilisateur saisit les modifications. 5. L'utilisateur valide la modification. 6. Le système enregistre les modifications.

Exceptions.

Exception : • **6.1** Le système affiche un message d'erreur d'enregistrement « Champ vide » • **6.2** Retour à l'étape 4 du scénario nominal

Post-condition.

Modifier le texte d'un message prédéfini

II.6 CONCEPTION

L'étude dynamique est une étape importante dans la définition des objets et la compréhension de leur fonctionnement dans le système, elle se base sur plusieurs modèles.

Relativement à notre système nous allons nous baser sur un modèle dynamique : Le diagramme de séquences (les scénarios).

II.6.1 Définition du diagramme de séquence

Le diagramme de séquence décrit la dynamique du système. À moins de modéliser un très petit système, il est difficile de représenter toute la dynamique d'un système sur un seul diagramme. Aussi la dynamique globale sera représentée par un ensemble de diagrammes de séquence, chacun étant généralement lié à une sous fonction du système.

Le diagramme de séquence décrit les interactions entre un groupe d'objets en montrant, de façon séquentielle, les envois de message qui interviennent entre les objets. Le diagramme peut également montrer les flux de données échangées lors des envois de message. [UML 2.0]

II.6.2 Diagrammes de séquence détaillés de notre système

Nous présenterons dans cette partie, les diagrammes de séquences détaillés appelés aussi diagramme d'interaction. Par rapport au diagramme de séquence système le changement de niveau d'abstraction est sensible, dans la mesure où nous avons remplacé le système vu comme une boite noire par un ensemble d'objets en interaction. Pour cela nous utiliserons les trois types de classes d'analyse, à savoir les dialogues, les contrôles et les entités. Nous respecterons également les règles d'interaction entre ces trois types d'objets, ainsi

- ✓ Les acteurs ne peuvent interagir qu'avec les interfaces graphiques ;
- ✓ Les Interfaces graphiques peuvent interagir qu'avec les contrôles ;
- ✓ Les contrôles peuvent interagir avec des interfaces graphiques, les entités ou d'autres contrôles.

II.6.3 Diagrammes des séquences

1. Diagramme de séquence « Authentification »

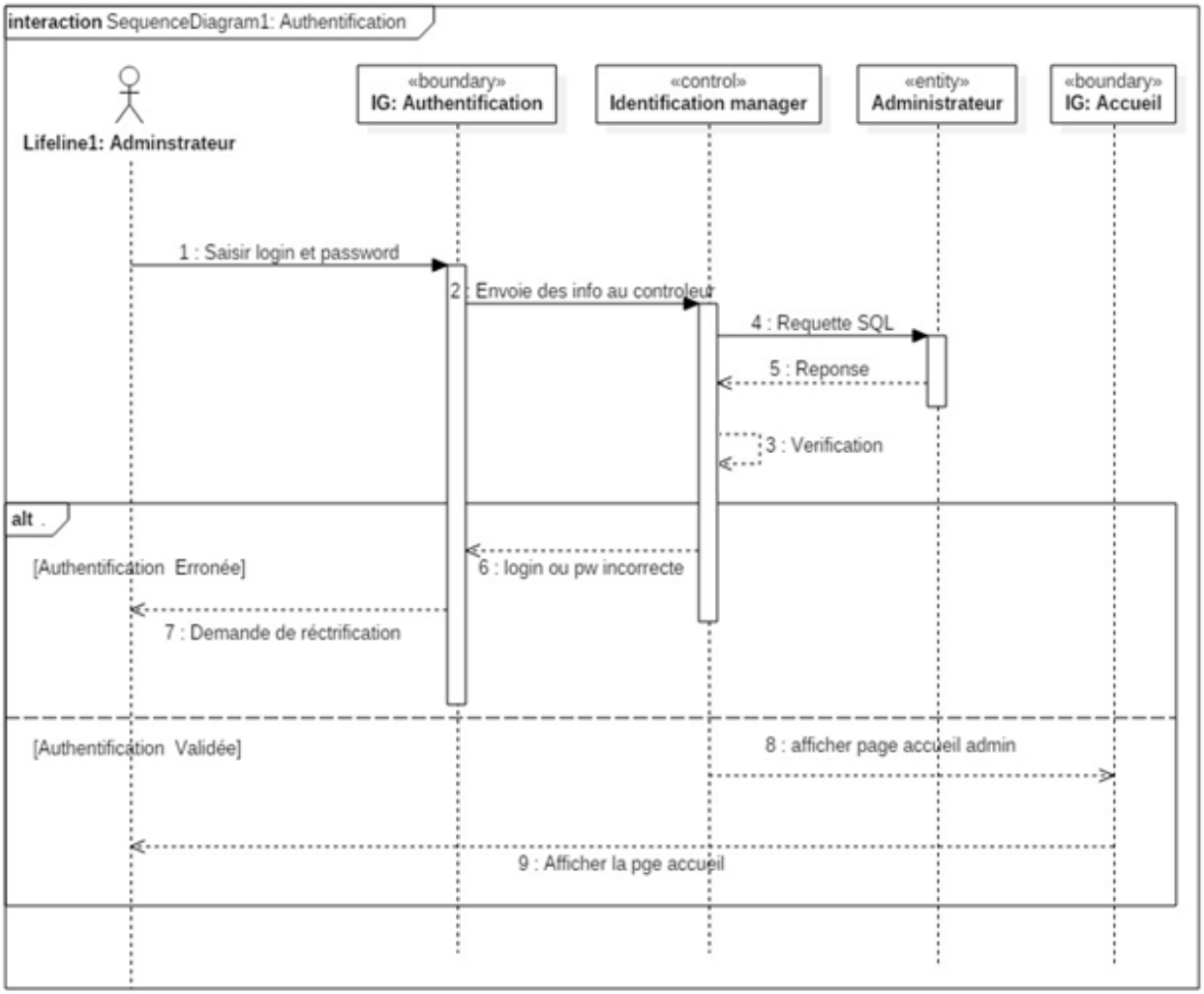


Figure II.9: Diagramme de séquence détaillé du cas d'utilisation "Authentification"

Objet	Description
« Boundary » IG : Authentification (web)	C'est l'interface qui permet à l'administrateur de saisir son login et mot de passe pour se connecter au système
« Boundary » IG : Accueil (web)	
« Control » Identification Manager	C'est le contrôleur qui vérifie le login et le mot de passe pour l'authentifier de l'administrateur
« Entity » Administrateur	Ensemble des données Administrateurs.

Tableau II.2 :Description du diagramme de séquence "Authentification"

2. **Diagramme de séquence « Ajouter un service médical »**

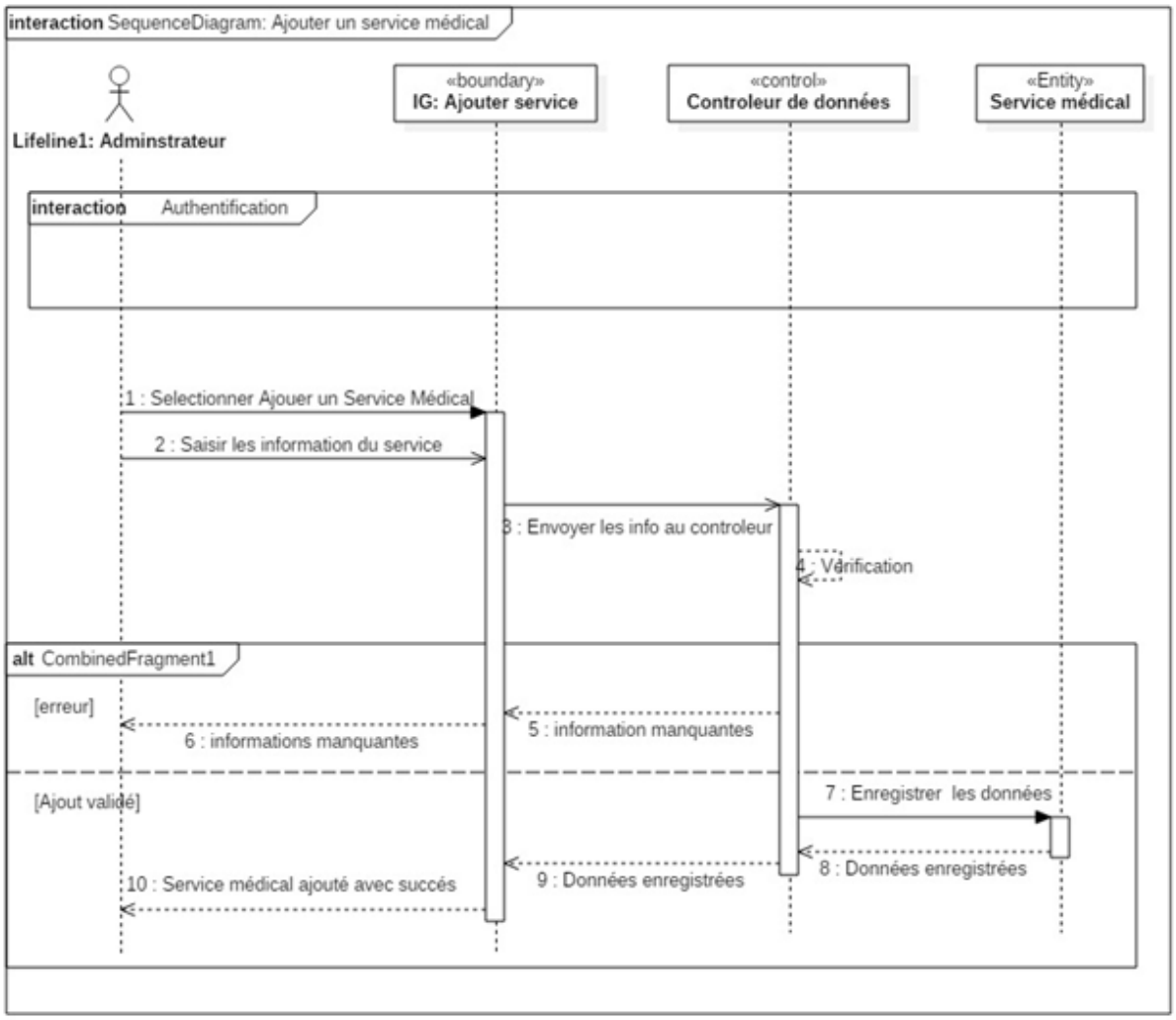


Figure II.10: Diagramme de séquence détaillé du CU "Ajouter un service médical"

Objet	Description
« Boundary » IG: « Ajouter un service médical »	C'est l'interface qui permet à l'administrateur de saisir les informations d'un service médical.
« Control » Contrôleur de données	Ensemble de programmes permettant le contrôle des informations saisit par l'administrateur
« Entity» Service médical	Ensemble de données sur les services médicaux.

Tableau II.3 : Description du diagramme de séquence « Ajouter un service médical »

3. Diagramme de séquence « Modifier un service médical »

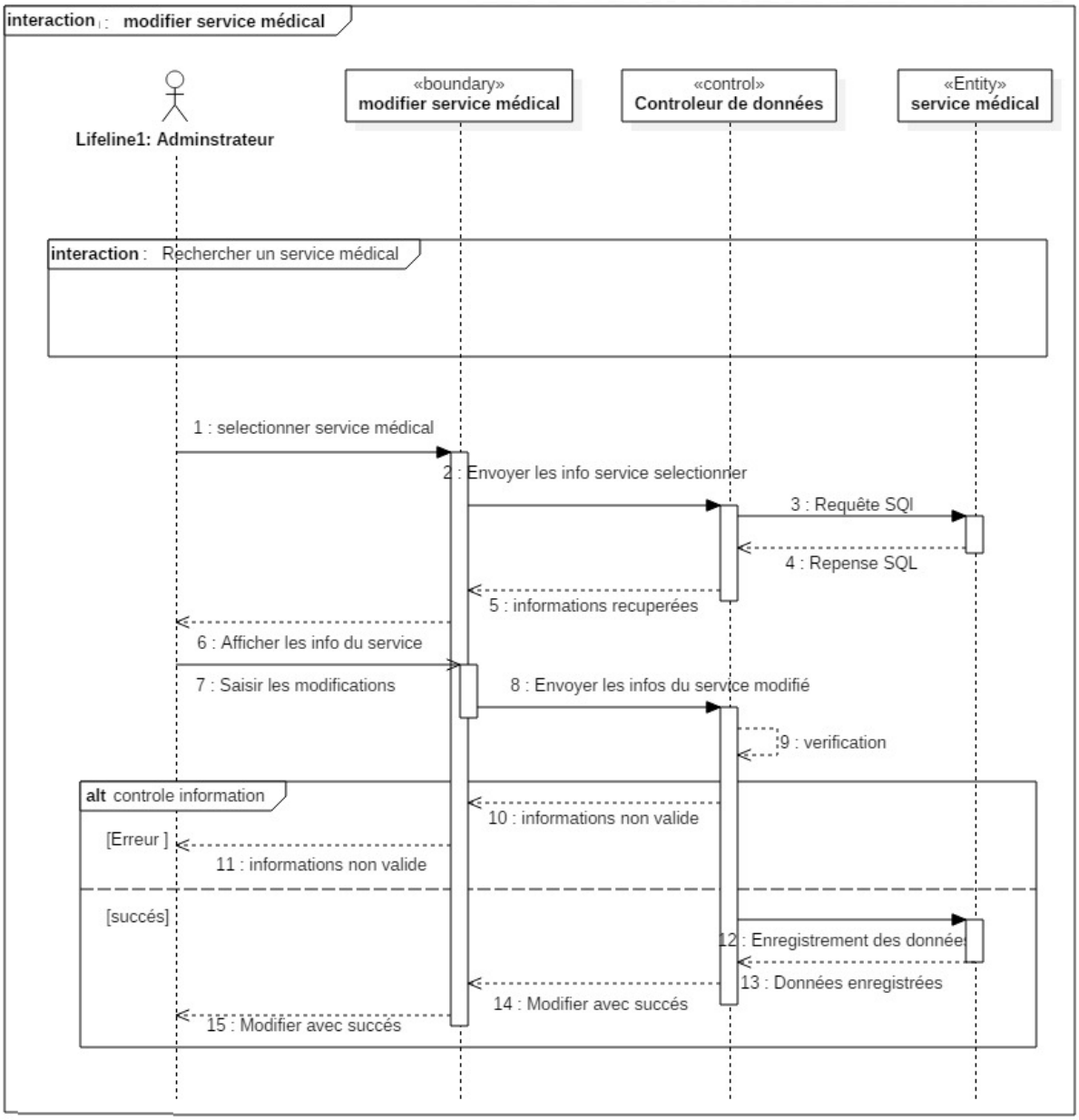


Figure II.11: Diagramme de séquence détaillé du CU "Modifier un service médical"

Objet	Description
« Boundary » IG : « Modifier un service médical »	C'est l'interface qui permet à l'administrateur d'afficher les informations du service médical à modifier et de changer ses informations.
« Control » Contrôleur des données	Ensemble de programme permettant le contrôle des informations du service médical a modifié
« Entity » Service médical	Ensemble de données sur les services médicaux.

Tableau II.4 : Description du diagramme de séquence « Modifier un service médical »

4. Diagramme de séquence « gestion des favoris »

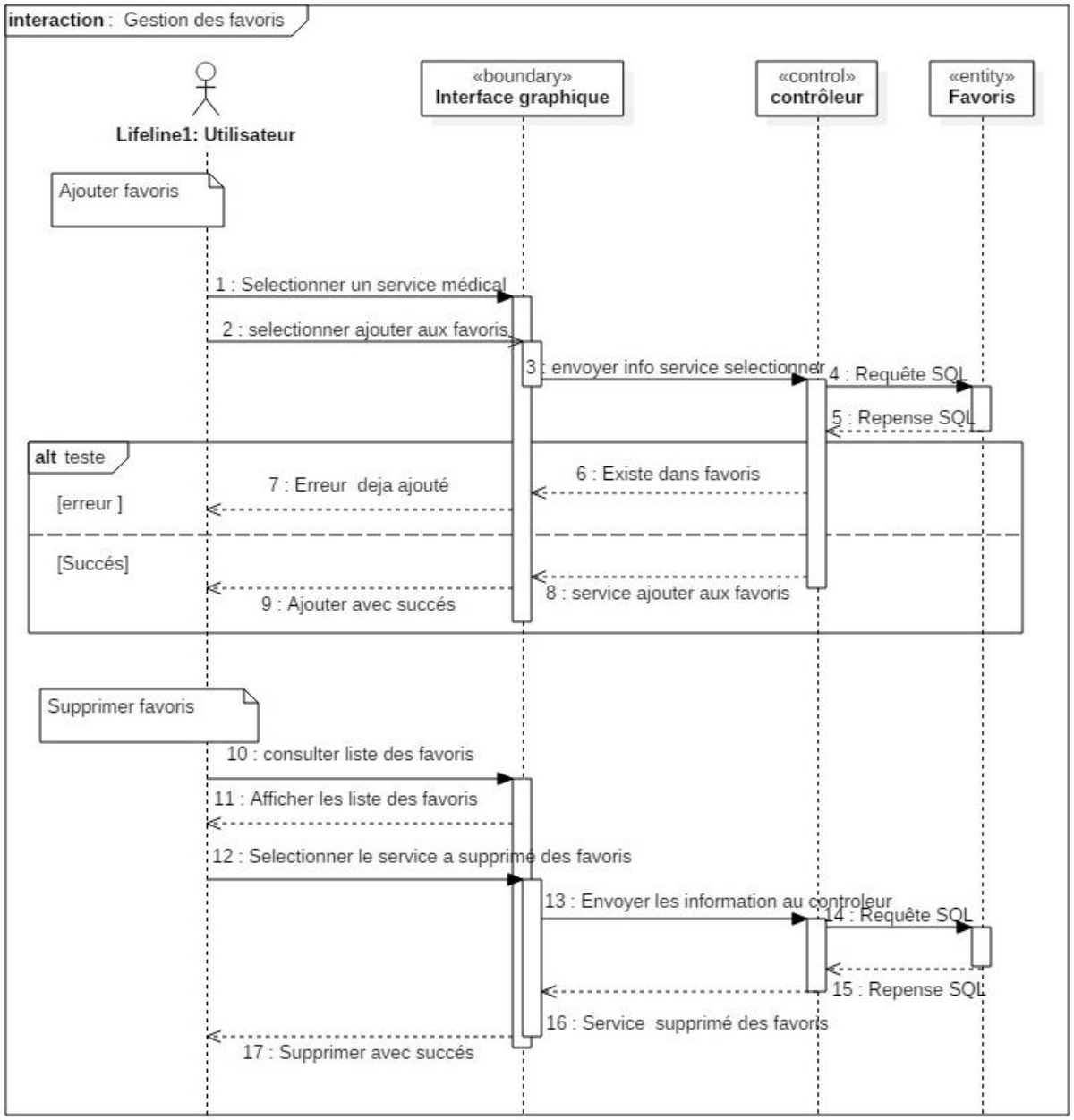


Tableau II.12: Diagramme de séquence "Gestion des favoris"

Objet	Description
« Boundary » Interface graphique « gestion des favoris (Afficher/ajouter/supprimer »	C'est l'interface qui permet à l'utilisateur de consulter la liste des favoris et de la modifier
« Control » Contrôleur d'information	Ensemble de fonctions permettant le contrôle de l'ajout et la suppression des services médicaux aux favoris
« Entity » Favoris	L'ensemble des services médicaux enregistré en favoris

Tableau II.5: Description du diagramme de séquence "Gestion des favoris"

5. Diagramme de séquence du CU « Rechercher un service médical »

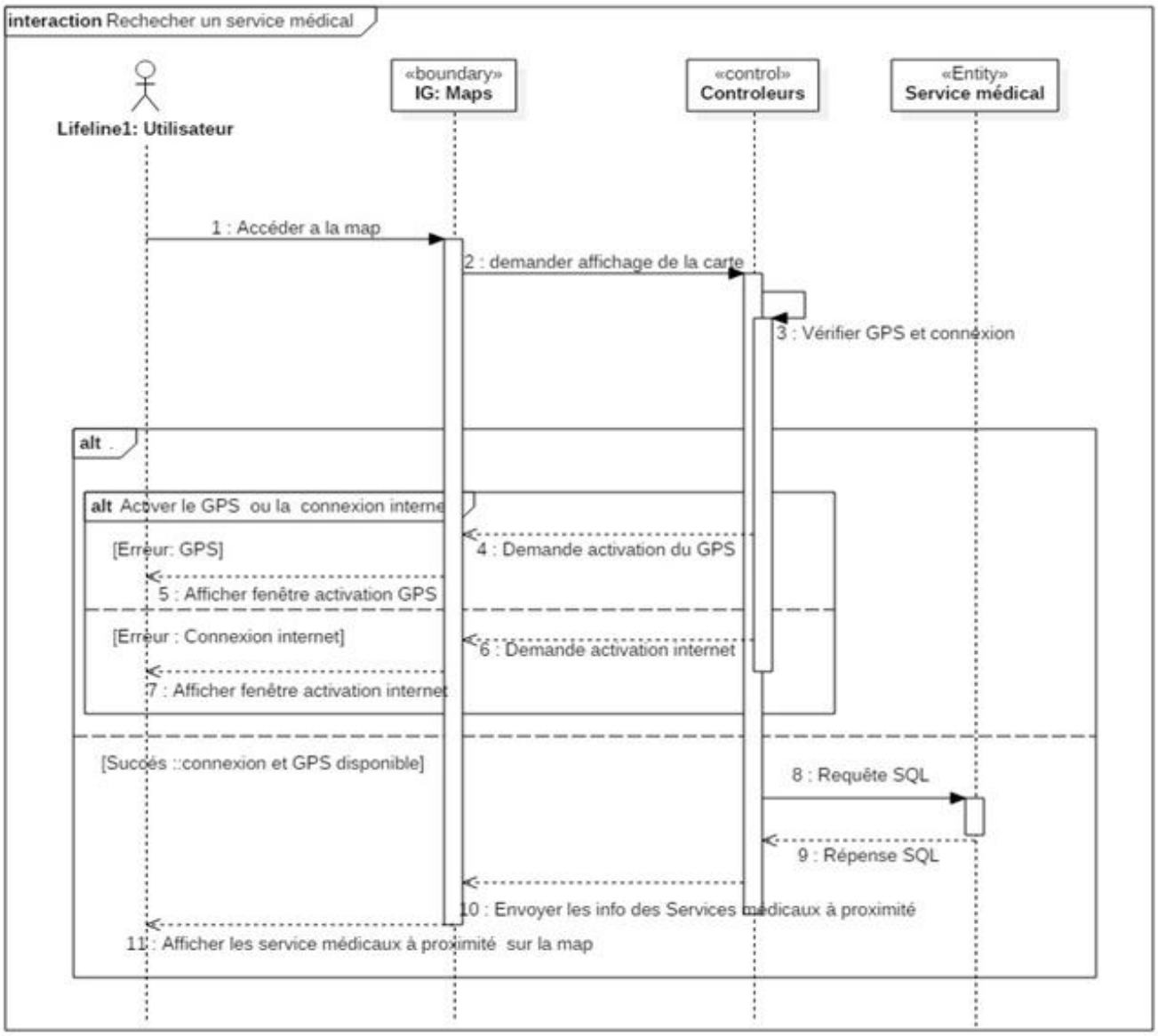


Figure II.13: Digramme de séquence détaillé du cas d'utilisation "Rechercher un service médical"

Objet	Description
« Boundary » IG : « MAPS »	C'est l'interface qui permet d'afficher la position d'un utilisateur sur la carte ainsi que la localisation des services médicaux à proximité.
« Control » Contrôleur d'information	Ensemble de fonctions qui vérifient l'activation du GPS et internet
« Entity » Service médical	Ensemble de données sur les services médicaux.

Tableau II.6 : Description du Diagramme de séquence « Rechercher un service médical

6. Diagramme de séquence du CU « modifier un message »

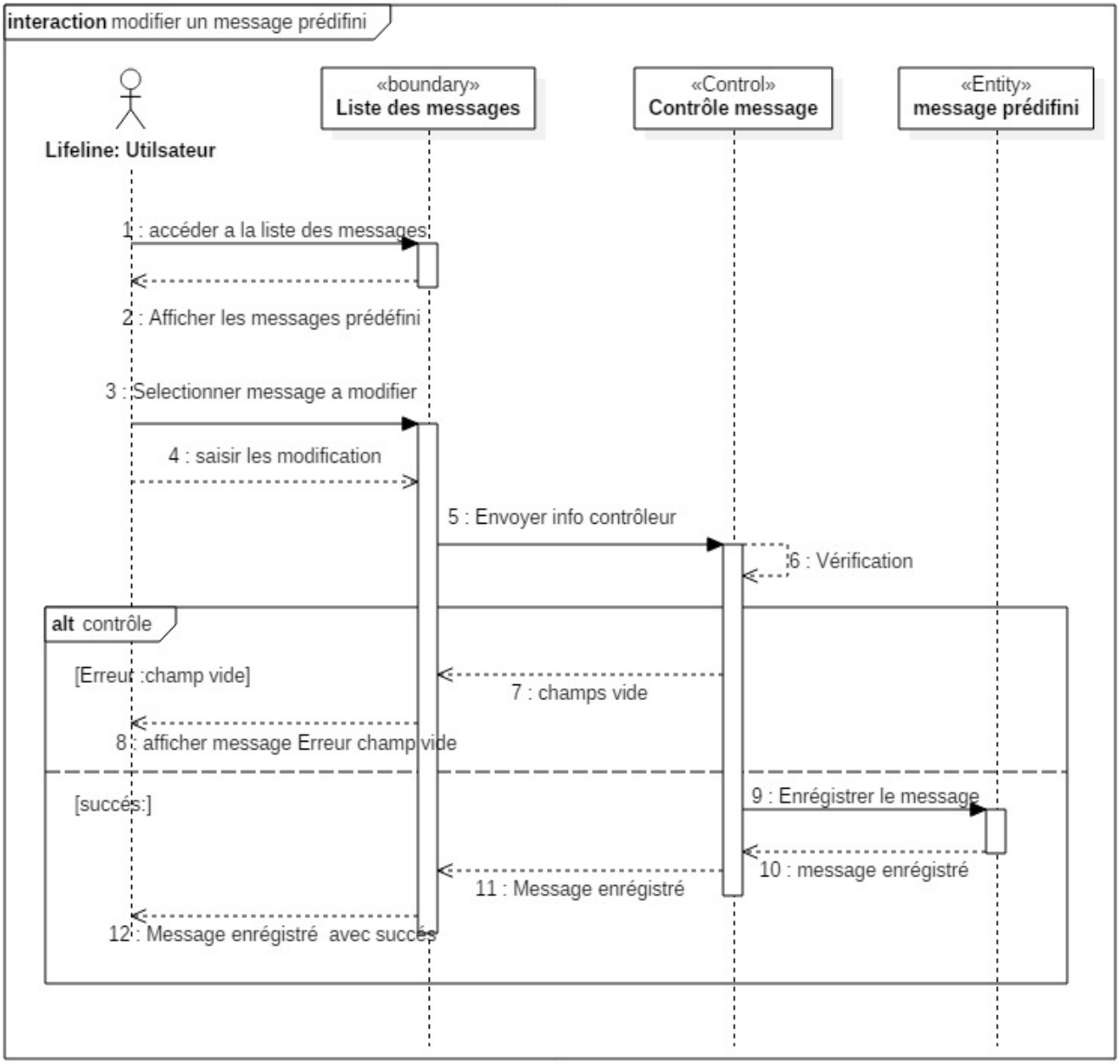


Figure 0II.14 : Digramme de séquence détaillé du cas d'utilisations "Modifier un message prédéfini"

Objet	Description
« Boundary » IG : « MAPS »	C'est l'interface qui permet d'afficher la liste des messages prédéfinis et de les modifier.
« Control » Contrôleur message	Ensemble de fonctions qui vérifient le champ du message à modifier.
« Entity » Message prédéfini	Ensemble de données des messages prédéfinis.

Tableau II.7 Description du DS "modifier un message prédéfini"

II.6.4 Identification des classes d'objets

Cette phase doit déterminer le modèle statique qui consiste en la détermination des classes d'objet, ses attributs et ses méthodes, les associations entre les classes puis la représentation du diagramme de classe qui donne une illustration schématisée des différentes classes et leurs relations.

Description détaillée des classes d'objets :

Classe	Attributs	Méthodes
Service médical	Id_service Nom_service Latitude Longitude Adresse Tel Fax Email	-Ajouter_service () ; -Visualiser_un_service(); -Modifier_un_service (); -Supprimer_service () ; -Liste_services_par_type() ; -TracerItiniraire() ; -Contacter() ;
Favori	Id_favori Id_service	-Ajouter_favori() ; -Supprimer_favori() ; -Consulter_favoris() ;
Type	Id_type Nom_type	-Ajouter_type_service() ; -Supprimer_type_service() ;
Proche	Id_proche Num_proche	-Ajouter_proche() ; -Supprimer_proche() ; -liste_proches() ;
Message	Id_message Text_message	-AjouterMessage() ; -ModifierMessage() ; -SupprimerMessage() ; -EnvoyerMessage() ;
Administrateur	Id_admin Login password	-consulter_espace_personnel() ;

Tableau II.8: Description détaillée des classes d'objets

II.6.5 Diagramme des classes

Il est considéré comme le digramme le plus important lors de la conception d'une application. Il définit explicitement les différentes classes du système, leurs attributs ainsi que leurs comportements (méthodes). Le diagramme de classe sera pris comme la référence à partir de laquelle va se dérouler le développement logiciel, et l'écriture du code source de notre application.

La figure suivante représente le diagramme de classe global du système :

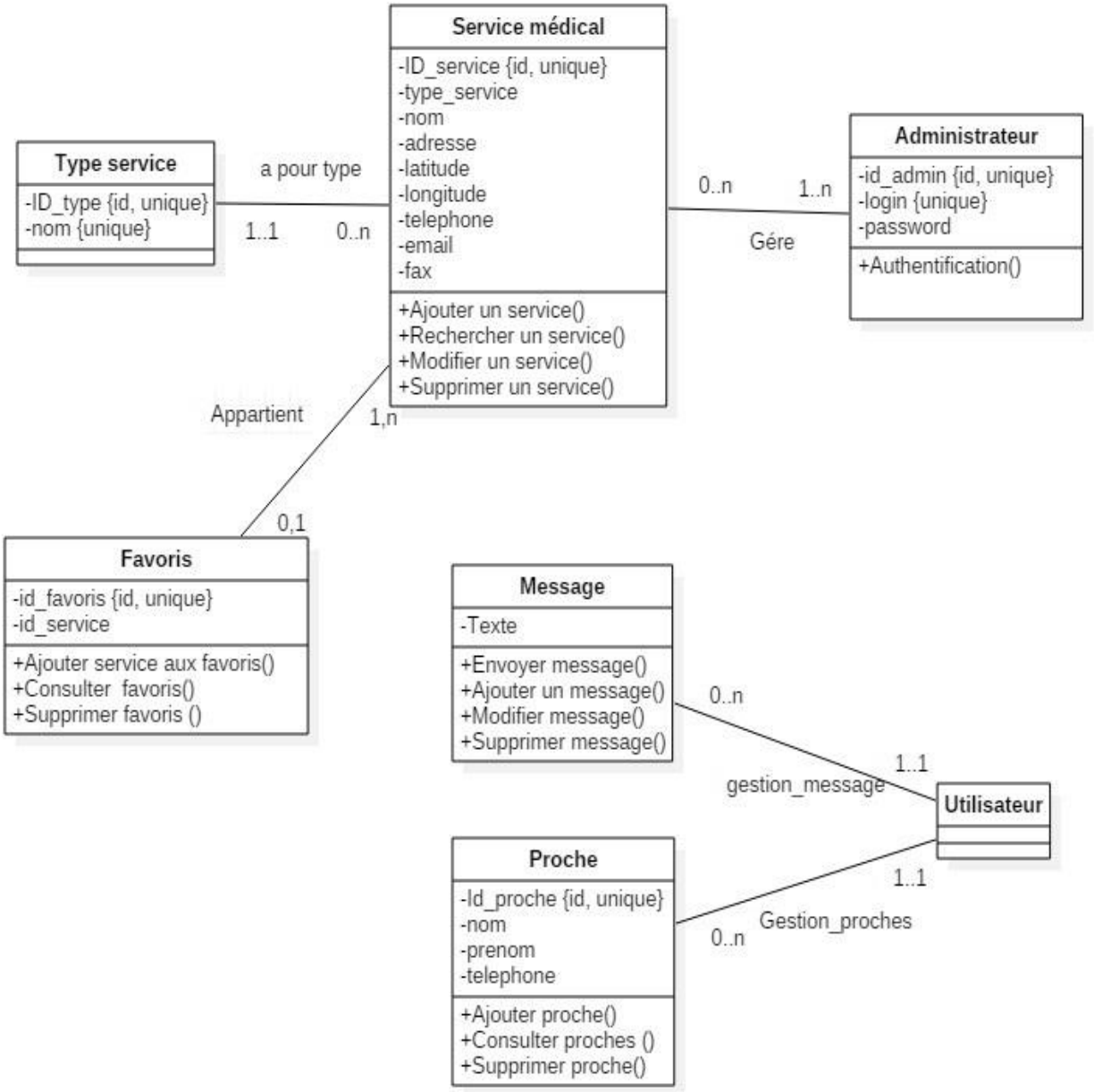


Figure II.15: Diagramme de classes général

II.6.6 Conception de la base de données

- ✓ **Administrateur :**

Nom du champ	Type de données	Description	Clef
id	INTEGER	Identifiant de l'administrateur	Primaire
login	TEXT	Compte de l'administrateur	
password	TEXT	Mot de passe de l'administrateur	

Tableau II.9 Table Administrateur

- ✓ **Service médical**

Nom du champ	Type de données	Description	Clef
id	INTEGER	Identifiant du service médical	Primaire
Nom	TEXT	Nom du service médical	
Adresse	TEXT	Adresse du service médical	
Type	INTEGER	Type du service médical	Clé étrangère
Latitude	FLOAT	Latitude du service médical	
Longitude	FLOAT	Longitude du service médical	
Tel	TEXT	Numéro de téléphone du service	
Email	TEXT	Email du service médical	
Fax	TEXT	Fax du service médical	

Tableau II.10 Table Service Médical

- ✓ **Type service médical**

Nom du champ	Type de données	Description	Clef
id	INTEER	Identifiant du type	Primaire
Nom	TEXT	Nom du type de service médical	

Tableau II.11 Table Type Service médical

- ✓ **Favoris**

Nom du champ	Type de données	Description	Clef
id	INTEGER	Identifiant favori	
Service_medical	INTEGER	Identifiant du service médical	Clé étrangère

Tableau II.12 Table Favoris

II.7 Conclusion

A travers ce chapitre, nous avons introduit les différentes fonctionnalités de notre application, nous avons présenté l'activité d'analyse qui a permis de livrer une spécification complète des besoins issus des diagrammes de cas d'utilisation de chaque acteur, puis nous avons élaboré les diagrammes de séquence et le diagramme de classes général, et enfin nous avons procédé à la conception de notre base de données.

Le chapitre suivant est consacré à la dernière étape de cycle de développement : la réalisation.

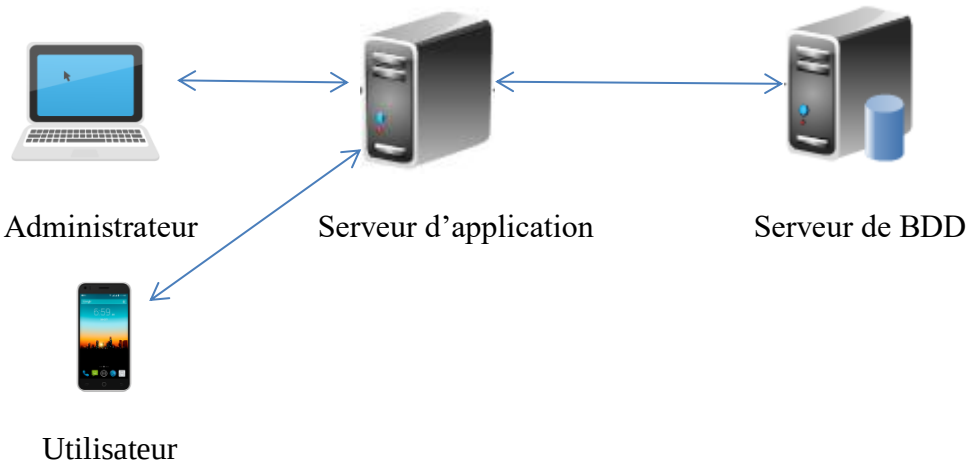
CHAPITRE III
REALISATION

III.1 Introduction

La réalisation présente le dernier volet de notre travail, après l'analyse et la conception de notre solution, nous allons procéder dans ce chapitre à la présentation de l'environnement de développement de notre application, les logiciels et technologies utilisées puis nous allons expliquer son fonctionnement avec des illustrations concrètes.

III.2 Architecture de notre application

Notre application utilise une architecture trois tiers comme illustré ci-dessous.



- **L'utilisateur** : C'est tout individu utlisant l'application Android sur son smartphone pour notamment localiser des services médicaux.
- **L'administrateur** : Utilise l'application web pour principalement mettre à jour la base de données.

Notre solution nécessite un serveur d'application afin d'assurer la logique applicative, la gestion et récupération des ressources sollicitées et permettre la mise à jour des données en les centralisant sur un serveur de base de données.

Le client Android demande les ressources au serveur d'application (En utilisant le système Android) qui récupère les données du serveur de BDD en utilisant des requêtes SQL et renvoi les données codées sous le format JSON.

III.3 L'environnement technique du travail

Après avoir fait une étude détaillée des différents modules constituant notre application, nous allons nous pencher sur la mise en œuvre de notre système ; en choisissant les différents outils de développement correspondants à des critères bien définis pour atteindre des objectifs bien précis.

III.3.1 Outil de conception

StartUML :

StarUML est un logiciel de modélisation UML, open source, supportant la spécification UML 2.0 et conçu au départ pour remplacer les logiciels propriétaire tel que Rational Rose and Borland Together. **[21]**

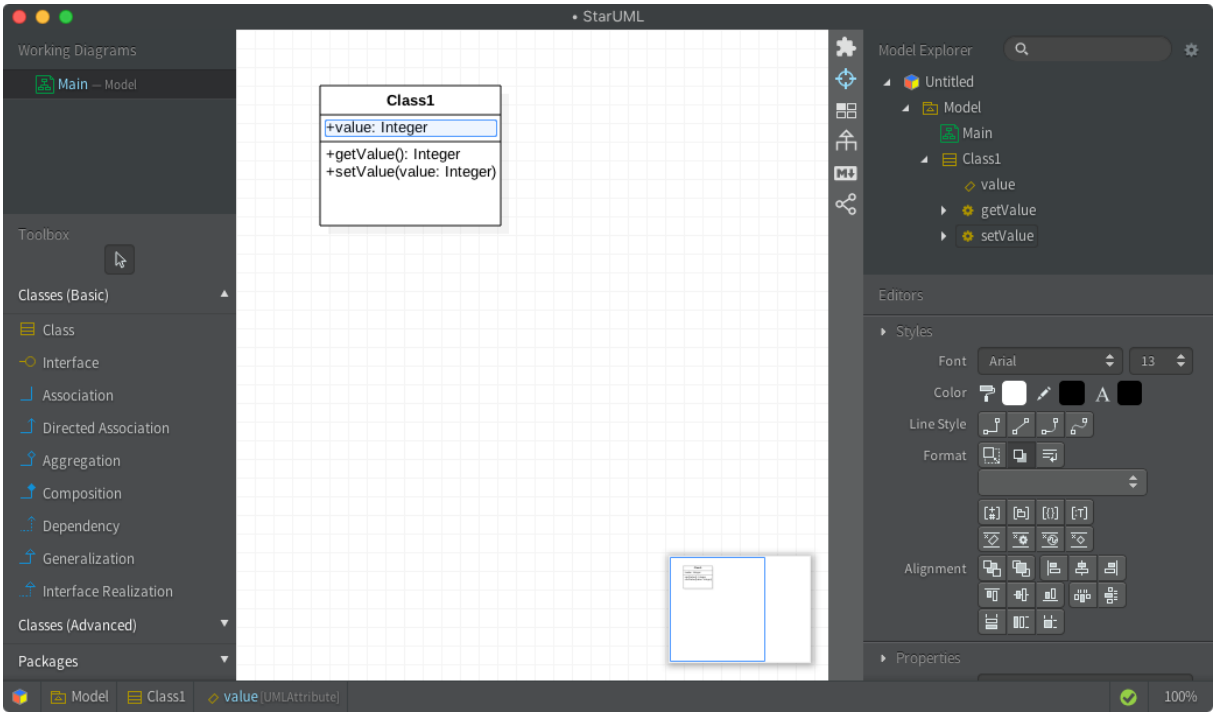


Figure III.1 - StartUML

III.3.2 Outils de développement

II.3.2.1 Android Studio

Android Studio est l'environnement de développement intégré (IDE) officiel de Google pour le développement d'applications Android, il est basé sur IntelliJ IDEA de JetBrains, un éditeur de code et outil de développement puissant et complet.

Il offre un outil de conception d'interfaces graphiques UI très moderne, permet de gérer le développement d'applications multilingues ainsi que des outils de débogages, d'inspection et de monitoring puissants.

Le développement se fait principalement sur Java, cependant depuis 2017 Android Studio supporte officiellement le développement d'applications Android sous Kotlin le nouveau langage de programmation développé par JetBrains. **[22]**

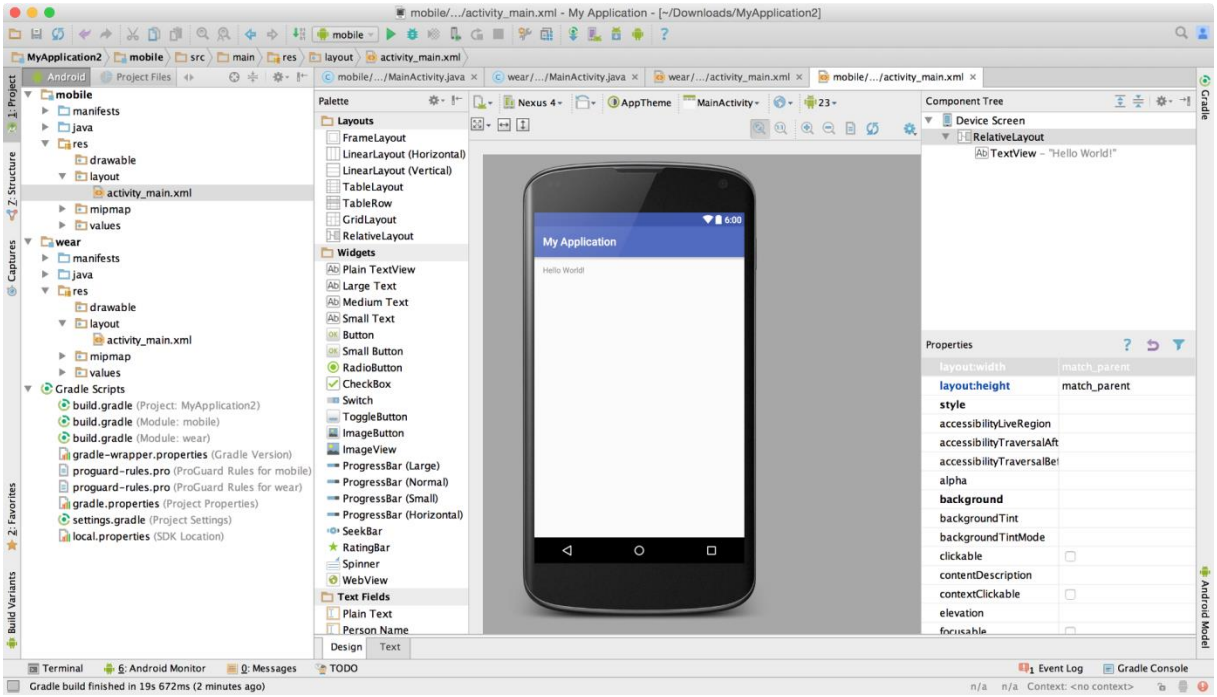


Figure III.2 - Android Studio

II. 3.2.2 Le SDK Android (Version 19)

Le kit de développement (SDK) d'Android est un ensemble complet d'outils de développement. Il inclut un débogueur, des bibliothèques logicielles, un émulateur, de la documentation, des exemples de code et des tutoriaux. **[23]**

SQLite

SQLite est une bibliothèque écrite en C qui propose un moteur de base de données relationnelle accessible par le langage SQL. SQLite implémente en grande partie le standard SQL92 et des propriétés ACID.

Contrairement aux serveurs de bases de données traditionnels, comme MySQL ou PostgreSQL, sa particularité est de ne pas reproduire le schéma habituel client-serveur mais d'être directement intégrée aux programmes. L'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier indépendant de la plateforme.

SQLite est le moteur de base de données le plus distribué au monde, grâce à son utilisation dans de nombreux logiciels grand public comme Firefox, Skype, Google Gears, dans certains produits d'Apple, d'Adobe et de McAfee et dans les bibliothèques standards de nombreux langages comme PHP ou Python.

De par son extrême légèreté, il est également très populaire sur les systèmes embarqués, notamment sur la plupart des Smartphones modernes : l'iPhone ainsi que les systèmes d'exploitation mobiles Symbian et Android l'utilisent comme base de données embarquée. **[24]**

DB Browser for SQLite

C'est un système de gestion de base de données pour des bases de données SQLite, il combine une interface facile à utiliser, une vitesse fulgurante et des fonctionnalités avancées.

DB Browser for SQLite permet de travailler avec un large éventail de base de données SQLite.

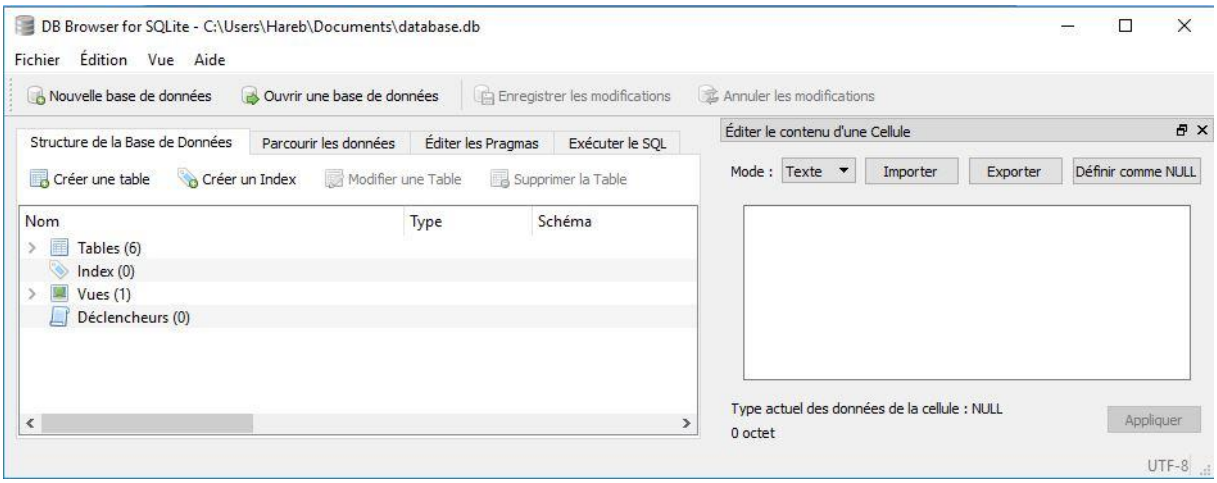


Figure III.3 - DB Browser for SQLite

Retrofit

Retrofit est une librairie permettant de réaliser des appels à des webservices REST sur Android de façon très simple et s'occupe de convertir les données JSON dans des objets Java et propose une architecture adaptée pour réagir en conséquence aux codes HTTP. **[25]**

XAMPP

XAMPP est une distribution Apache entièrement gratuite et facile à installer contenant MySQL, PHP et Perl permettant de faire fonctionner localement (sans se connecter à un serveur externe) des scripts PHP. **[26]**

PHPStorm

PHPStorm est un puissant environnement de développement intégré pour PHP, HTML, JavaScript, basé sur IntelliJ comme Android Studio offrant donc une interface semblable.

III.3.3 Langages de programmation

- **Java**

Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy. Il fut présenté officiellement le 23 mai 1995 au SunWorld. La société Sun a été ensuite rachetée en 2009 par la société Oracle qui possède désormais Java.

La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou Linux, avec peu ou pas du tout de modifications.

Pour cela, diverses plateformes associées visent à garantir la portabilité des applications développées en Java. **[27]**

- **PHP**

  PHP (PHP Hypertext Preprocessor) est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. Il peut être intégré facilement au HTML.

- **JSON**

  JSON (JavaScript Object Notation) est un format d'encodage de données permettant de sérialiser une structure de données au format texte pour les stocker ou pour les échanger notamment via Internet mais aussi entre les différentes couches d'une application. La syntaxe très simple de JSON explique principalement son succès.

III.3. 4 Outils matériels

Au cours des différentes étapes de développement nous avons utilisé la configuration matérielle suivante :

- Laptop Acer : Intel Core i7 @3.2GHZ, RAM 6GO, Windows 10
- Smartphone Huawei Y6II

III.4 Présentation de quelques interfaces de l'application

Dans cette section nous allons présenter quelques interfaces de notre application.

III.4.1 L'application Android

La figure suivante est l'interface du menu principal de l'application

Figure III.3 - Menu principal de l'application

Les taches effectuées par chaque bouton sont décrites ci-dessous :

-Le bouton « **services médicaux** » permet de consulter la map et rechercher un service médical à proximité.

- « Alerter mes proches » : Permet d'alerter ses proches en envoyant un SMS prédéfini.

- « Options » : Permet d'accéder aux options de l'application.

- « Mise à jour » : Permet de mettre à jours la base de données de l'application.

En choisissant la fonctionnalité « Services médicaux », l'utilisateur peut consulter la map et rechercher un service médical à proximité.

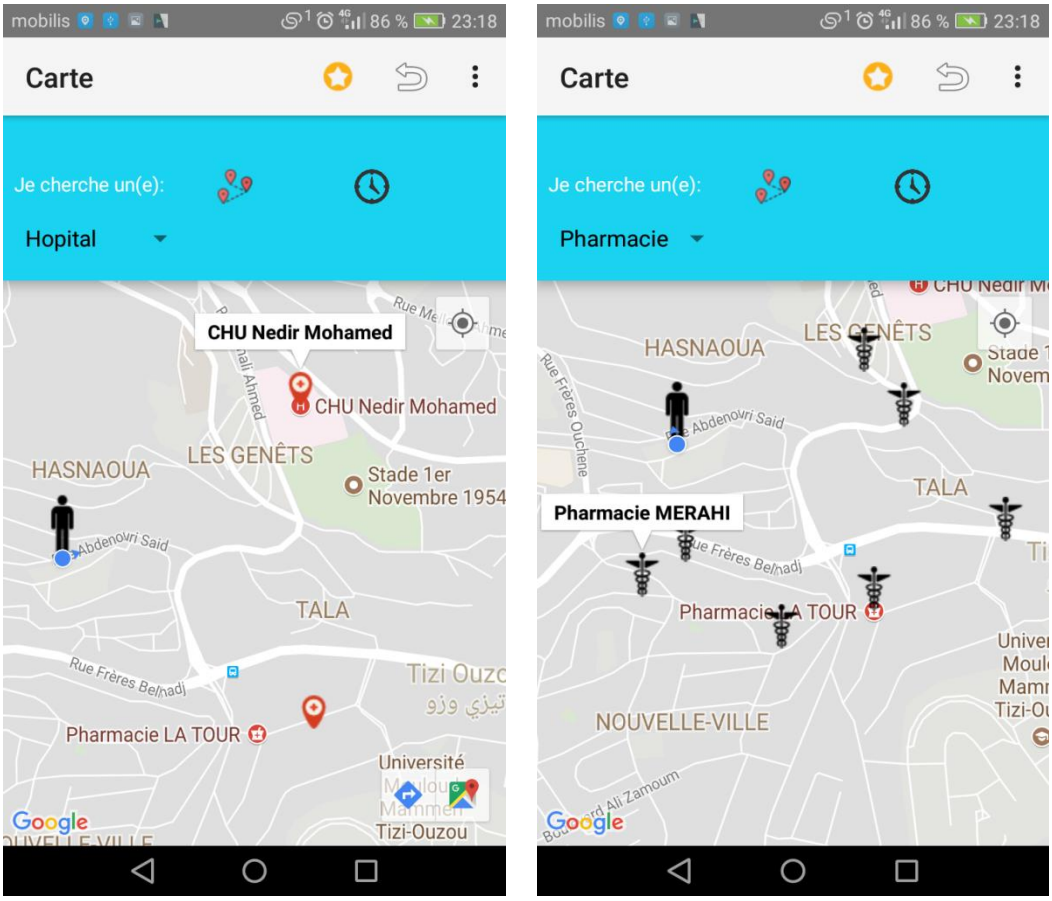
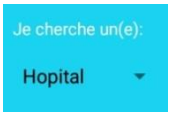


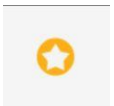
Figure III.4 - Affichage des services médicaux les plus proches

Cette interface nous permet voir sa position et de localiser les services médicaux à proximité selon le type choisi dans la liste déroulante plus haut, si le GPS n'est pas activé, l'utilisateur sera automatiquement redirigé vers les paramètres de son téléphone pour l'activer.

Les fonctionnalités disponibles sont :



Permet de choisir le type de service médical recherché.


Permet d'accéder à la liste de ses lieux favoris.


Permet de retourner vers le menu principal.

En cliquant sur un des services localisés sur la carte, nous accédons à l'interface suivante permettant d'afficher les informations concernant le lieu.

Figure III.5 -Fiche détaillée d'un lieu

Cette interface permet de tracer l'itinéraire vers le service, l'ajouter à ses favoris pour le retrouver plus tard, contacter le service par téléphone ou par email.

En choisissant de tracer l'itinéraire nous accédons à l'interface suivante :

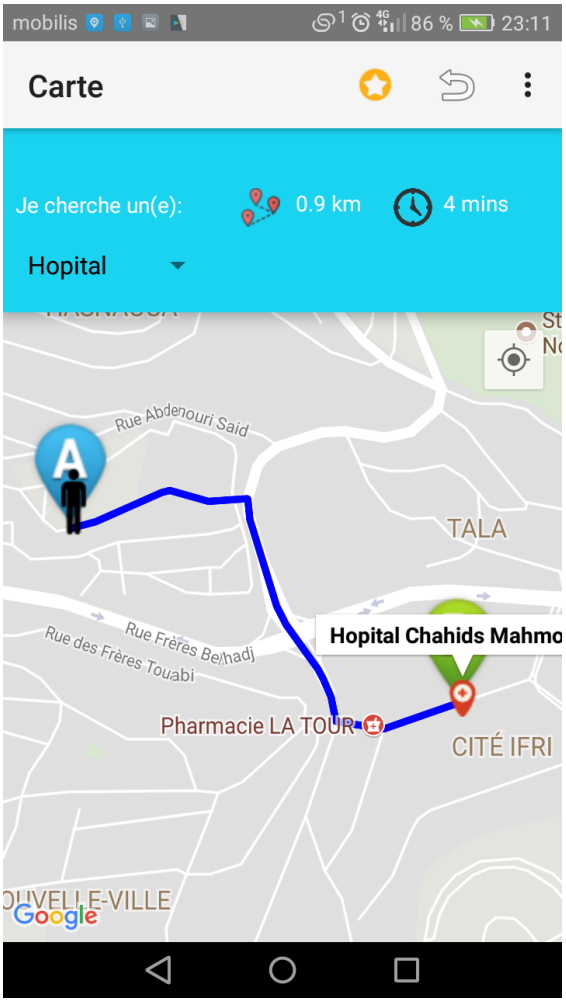


Figure III.6 - Visualisation de l'itinéraire

Dans cette interface l'utilisateur peut visualiser l'itinéraire le plus proche vers sa destination, la position actuelle de l'utilisateur est marquée par un « A » la destination par un « B ».

En haut à droite est affichée la distance entre l'utilisateur et sa cible ainsi que la durée estimée pour l'atteindre.

En cliquant sur le bouton « Favoris »  en haut l'utilisateur peut accéder à la liste de ses lieux favoris

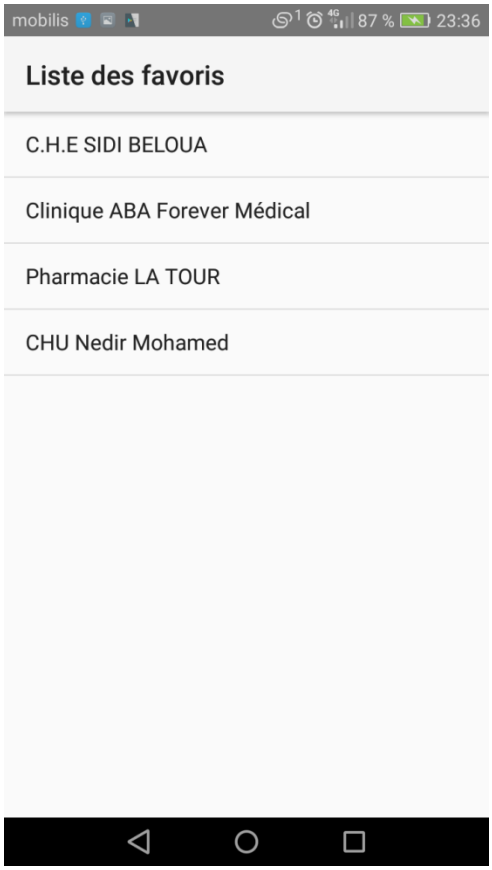


Figure III.7 - Liste des favoris

Cette interface représente la liste des lieux ajoutés aux favoris par l'utilisateur, elle permet de retrouver des services médicaux facilement à tout moment sans devoir repasser par la carte.

En cliquant sur un des lieux nous accédons à sa fiche détaillée vue précédemment.

Dans le menu principal en cliquant sur « **Alerter mes proches** » nous pouvons envoyer un message d'alerte à ses proches prédéfini en le sélectionnant.

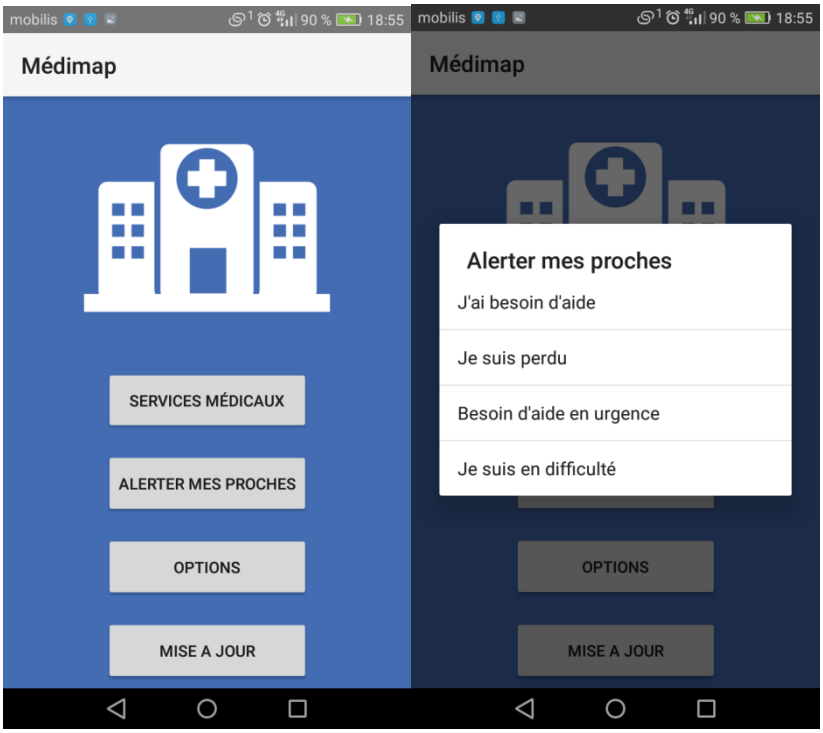


Figure III.8 - Alerter mes proches

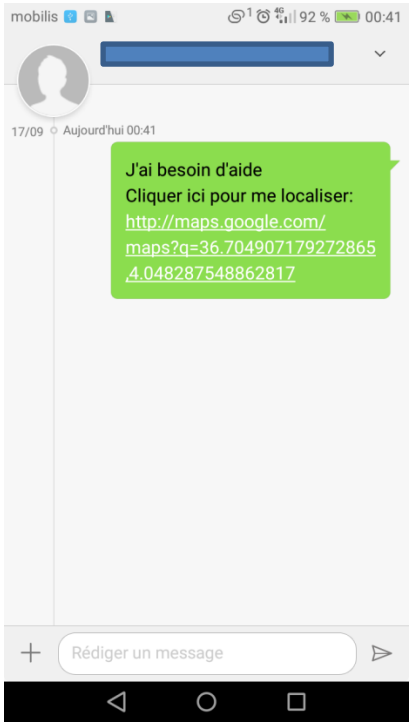


Figure III.9 - Envoi message SOS

Le message envoyé contient le message prédéfini ainsi qu'un lien google maps avec ses coordonnées pour être localisé.

Le bouton « Options » nous permet d'accéder à interface suivante :

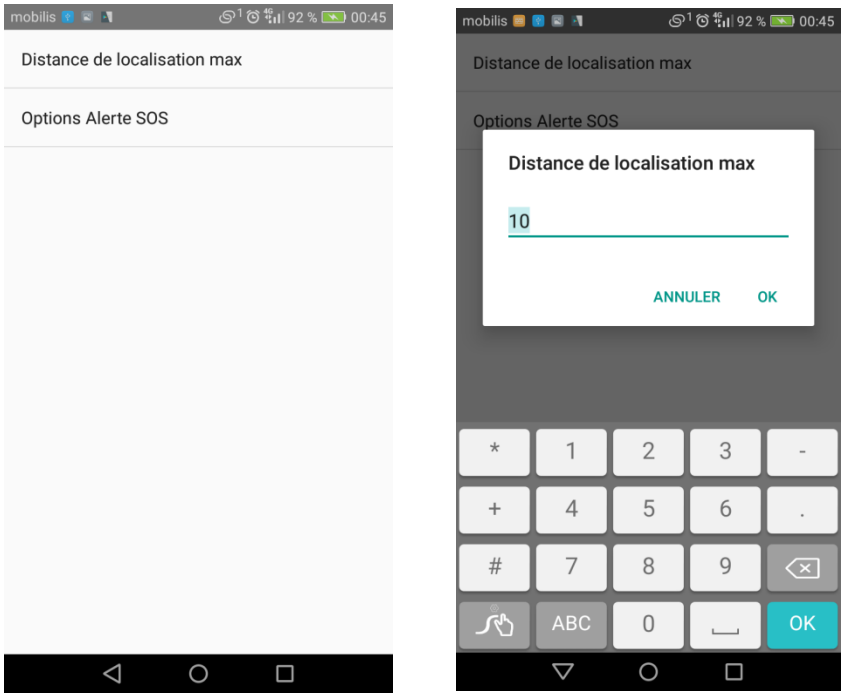


Figure III.10 - Menu options

Nous avons la possiblité de modifier le rayon de localisation en le spécifiant en Km, nous pouvons aussi accéder aux options relatives à la fonction SOS

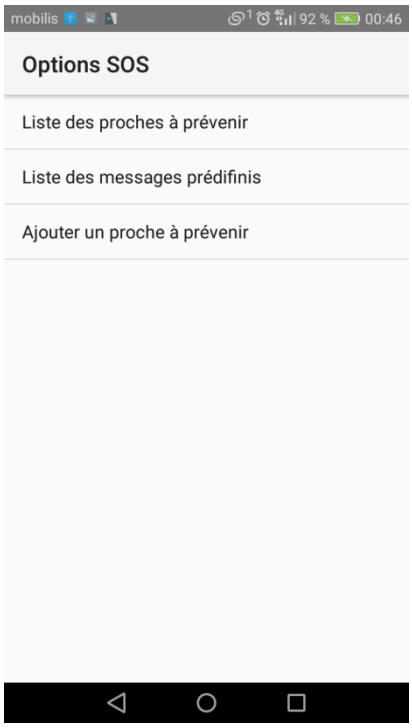


Figure III.11 – Liste options SOS

.

Nous pouvons gérer la liste des contacts à alerter, (Les consulter, les supprimer ou ajouter).

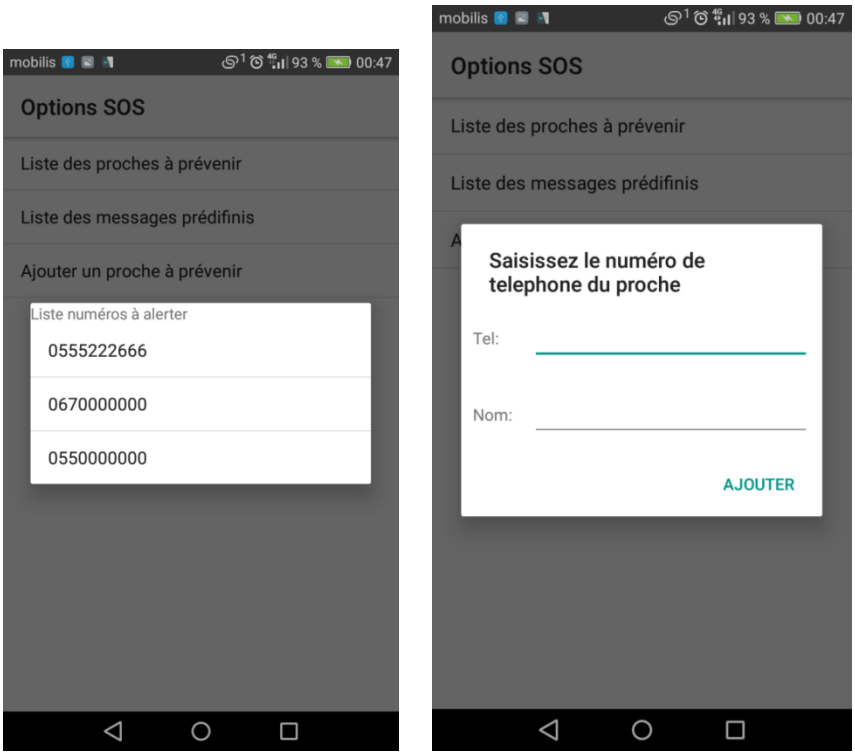


Figure III.12 – Gestion des proches à alerter

L'interface suivante permet de modifier les messages prédéfinis à envoyer par SMS.

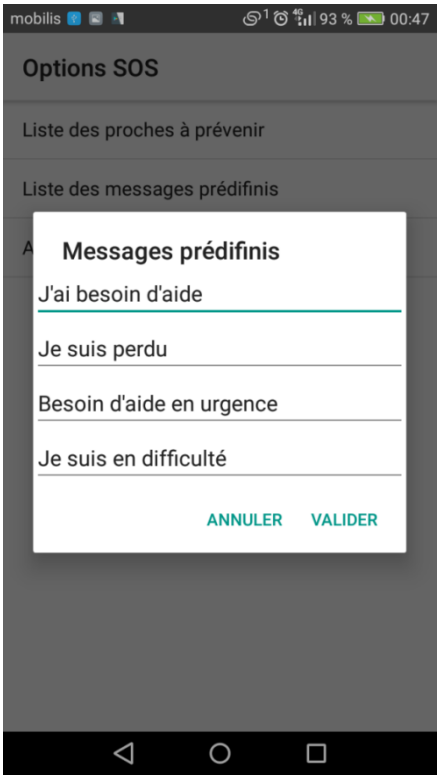


Figure III.13 - Gestion des messages prédifinis

III.4.2 L'Application web

L'application web permet principalement à l'administrateur de mettre à jour la base de données.

L'illustration suivante montre l'interface de connexion à l'espace administrateur.

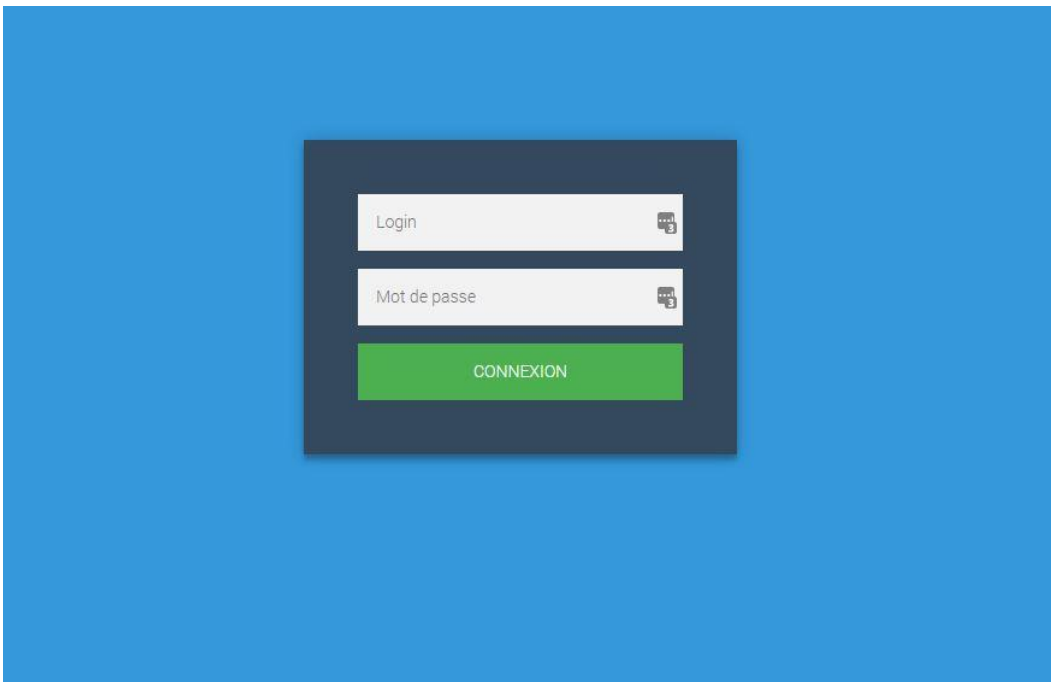


Figure III.13 – Interface de connexion à l'administration

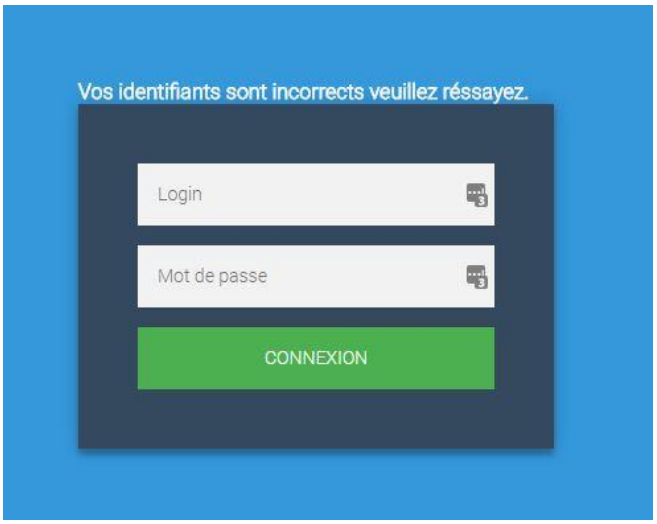


Figure III.14 – Erreur de connexion

Si le login et mot de passe sont corrects nous accédons à l'espace administrateur :

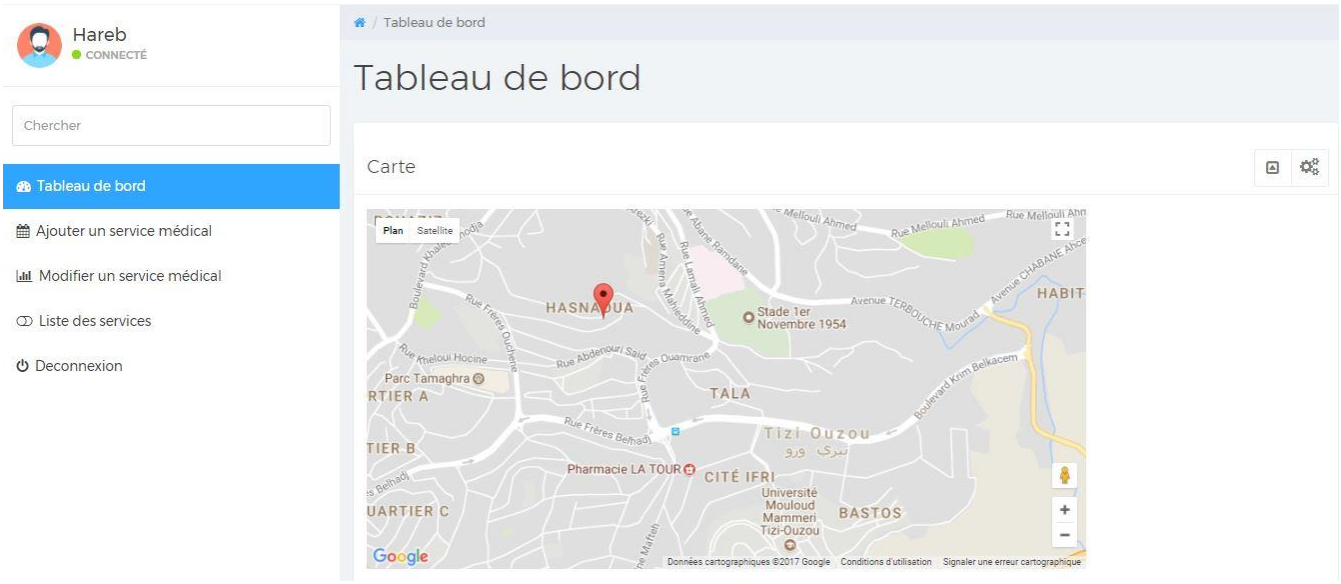


Figure III.15 - Espace administrateur

L'administrateur s'occupe principalement d'ajouter/modifier et supprimer les services médicaux de la base de données.

L'interface suivante montre la fonction « Ajouter un service médical ».

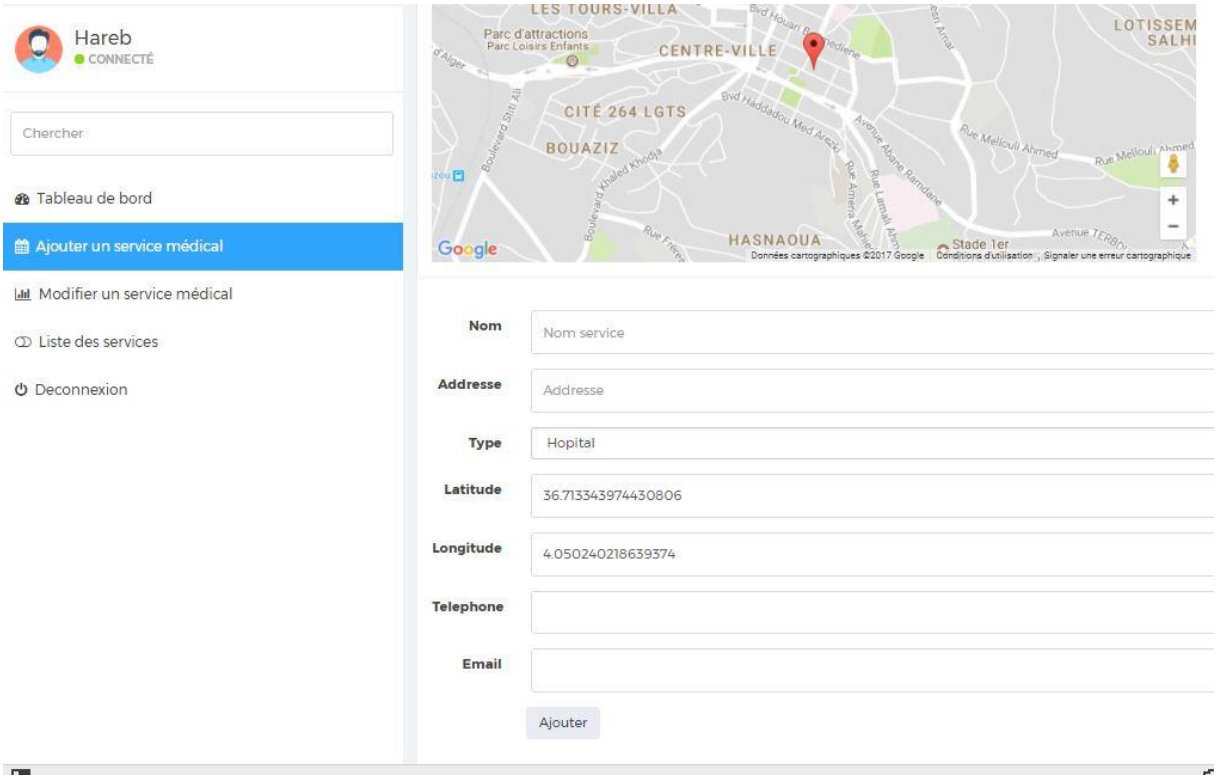


Figure III-16 - Ajout d'un service médical

L'administrateur peut ajouter un service médical en cliquant directement sur un point précis sur la carte afin de récupérer automatiquement les coordonnées, il peut aussi les saisir manuellement sur les champs et valider.

III.5 Conclusion

Dans ce dernier chapitre nous avons présenté les différents outils utilisés pour aboutir à la réalisation de l'application. Nous avons aussi montré une vue générale de l'application réalisée et les différents rendus visuels qu'elle offrait ainsi que ses différents composants en expliquant le rôle de chacun.

CONCLUSION GENERALE

L'objectif de notre travail était la conception et réalisation d'une application Android permettant de localiser les services médicaux les plus proches.

Cette application permet à l'utilisateur de connaitre sa position exacte et de localiser les services médicaux à proximité qui l'entourent, consulter la fiche détaillée d'un lieu, tracer l'itinéraire le plus proche vers la destination, contacter l'établissement, l'ajouter aux favoris, etc.

Elle permet en outre d'alerter ses proches avec un SMS SOS en cas d'urgence en envoyant un message prédéfini avec un lien vers la position exacte de l'utilisateur.

L'étude conceptuelle et technique qui ont abouti à la réalisation de notre solution constituent les chapitres de notre mémoire. Celui-ci nous a permis d'acquérir des connaissances inestimables dans le développement d'application mobiles, de faire face aux contraintes réelles tout en vivant les différentes phases du développement et la production d'une application mobile.

Pour ce faire nous avons recouru à différents outils et technologies jugées nécessaires à l'aboutissement de notre projet.

Dans un premier lieu nous avons fait une étude théorique sur les notions de base du système d'exploitation Android ainsi les technologies employées. Par la suite nous avons entamé l'analyse et la spécification des besoins de l'application et leur modélisation par le diagramme de contexte et les diagrammes de cas d'utilisation.

Partant de la spécification, nous avons réalisé la conception de l'application à travers les différents diagrammes UML à savoir le diagramme de classes et les diagrammes de séquence.

Enfin, dans le chapitre réalisation, nous avons décrit l'environnement de développement, et les différents outils, langages, technologies utilisées pour aboutir au résultat présenté dans les illustrations des interfaces qui le conclu.

L'application mobile est développée en langage JAVA sous Android Studio, les étapes de développement nous ont conduit aussi acquérir des connaissances dans le développement WEB notamment pour le développement des services web PHP permettant de gérer la connexion avec le serveur de base de données, assurer la mise à jour et la récupération de ces données distantes avec Android.

Ces étapes nous ont permis de réaliser une application complète et fonctionnelle. Comme perspectives nous pouvons envisager :

- L'intégration des pharmacies de garde.
- L'amélioration du design de l'application.

BIBLIOGRAPHIE

BIBILOGRAPHIE

[1] Netmarketshare. sur https://netmarketshare.com/mobile-phones.aspx?qprid=9&qpcustomb=1&qpcustom=iOS,Android

[2] https://fr.wikipedia.org/wiki/Apple_iOS

[3] http://www.futura-sciences.com/tech/definitions/smartphone-windows-phone-15584/

[4] http://www.numerama.com/tech/132165-les-versions-dandroid-les-plus-utilisees.html.

[5] https://iis-madagascar.com/repartition-versions-dandroid-succes-dandroid-7-nougat/

[6] http://javamind-fr.blogspot.com/2012/05/android-cycle-de-vie-des-activites-5x.html

[7] https://openclassrooms.com/courses/creez-des-applications-pour-android/votre-premiere-application-1

[8] http://www.e-naxos.com/Blog/post/Cross-plateforme-Android-Part-3-Activite-et-cycle-de-vie.aspx

[9] 5. (2009). Pro Android 5 | Dave MacLean.

[10] https://iutinfo.unice.fr/wppt/wp-content/uploads/2017/06/Etude_et_tests_solutions.pdf.

[11] https://developer.android.com/studio/index.html

[12] https://www.doc-etudiant.fr/Informatique/Programmation/Cours-Introduction-a-eclipse-130720.html.

[13] https://fr.wikipedia.org/wiki/Géolocalisation

[14] http://reperageterrestre.e-monsite.com/pages/le-gps-global-positioning-system.html

[15] http: //anrprodige.com/index.php?n=Technologies.Geolocalisation

[16] https://fr.organilog.com/454-fonctionnement-geolocalisation-mobile/

[18] Melle SEGHIER Nor El Houda ,Localisation d'un mobile dans un réseau UMTS, Mémoire de Magister, Universté d'Oran.

[19] Cours Univ Paris 5. sur http://www.math-info.univ-paris5.fr/~bouzy/Doc/UML-NotesCours.pdf

[20]. Chantal Morly, Jean Hugues, Berland le blanc , «UML2 pour l'analyse d'un système d'information» .

[21] https://en.wikipedia.org/wiki/StarUML.

[22] . https://www.developpez.com/actu/137409/La-premiere-preversion-d-Android-Studio-3-0-disponible-avec-le-support-de-Kotlin-plus-de-fonctionnalites-Java-8-et-bien-plus-encore/.

[23] Introduction to Android Application Development, Joseph Annuzzi, Jr. 2013

[24] https://fr.wikipedia.org/wiki/SQLite

[25] http://tutos-android-france.com/retrofit-webservices-rest/.

[26] https://www.apachefriends.org/fr/index.html.

[27] Java in a Nutshell, 6th Edition, Benjamin J Evans, David Flanagan 2014