REPUBLIQUE ALGERIENNE DEMOCRAYIQUE ET POPULAIRE

MINISTRE DE L4ENSEIGNEMENT SUPERIEUR ET DE LA

RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERIDE TIZI-OUZOU

FACULTE DU GENIE ELECTRIQUE ET INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

Mémoire

De fin d'étude

En vue de l'obtention du diplôme master en informatique

Option réseaux mobilités et systèmes embarqués

Thème

Application Android qui utilise la technologie GPS

Dirigé par :

Mm BELKADI

Realise par :

Mr : SLIMANI Farid

Membres du jury

……...

……

……

Promotion 2015

Dédicaces

Je dédie cet humble travail:

A mes parents adorés

A mon frères et mes Sœurs

A toute ma famille

A tous mes amis

Farid

~ Remerciements ~

Je remercie tout d'abord Dieu qui m'a donné la foi et le courage pour accomplir ce projet.

Je tenu aussi à exprimer ma reconnaissance et profonde gratitude à ma promotrice Mm BELKADI pour m'avoir encadrés durant cette année, pour sa forte présence et sa disponibilité, pour son exigence scientifique et ses précieuses orientations méthodologiques, pour son encouragement et sa patience.

Que les membres du jury trouvent ici mes plus vifs remerciements pour avoir accepter d'honorer par leur jugement notre travail. Aussi, j'adresse mes remerciements à tous nos enseignants de l'UMMTO pour m'avoir appris le goût de l'effort et du travail.

Un grand merci aussi à toute personne qui de prés ou de loin a contribué à ce que cet humble travail voit le jour.

Sommaire

Chapitre III : Analyse et Conception

Chapitre IV : Réalisation

Introduction générale

De plus en plus des personnes accèdent désormais aux services de géolocalisation via des moyens "non traditionnels" comme les terminaux mobiles, et ce nombre ne cesse d'augmenter.

L'Android est une nouvelle plateforme en code source ouverte. De plus, selon Google qui est le majeur distributeur, Android est une plateforme puissante, moderne, sûre et ouverte. Grâce à l'ouverture du code source et des APIs, les développeurs obtiennent la permission d'intégrer, d'agrandir et de replacer les composants existants. Les utilisateurs peuvent adapter les applications à leur besoin. C'est pour cela nous avons choisi la plateforme Android pour faire une étude approfondie et pour développer notre application.

Le but de notre travail est de développer une application Android qui utilise la technologie gps, cette applications est destinée pour les appareils qui marche sous le système d'exploitation Android v2.1 ou une version ultérieure.

L'objectif principal de l'application est de permettre à l'utilisateur d'enregistrer ses déplacements, de calculer la distance parcouru, et de le consulter la liste des parcours effectué, de contrôler sa vitesse de déplacement (voiture), et de tracé un itinéraire d'un point verre un autre.

Afin de mener à bien notre projet, nous avons réparti le contenu de notre travail en quatre chapitres, ainsi :

- Le premier chapitre s'intitule « Etat de l'art d'Android », Ce chapitre est une introduction à la plate-forme, aux outils et à la configuration de notre environnement de travail.

- Le deuxième chapitre s'intitule « Géolocalisation », dans ce chapitre nous avons opté pour une description de cette technologie qui fait parler d'elle de plus en plus. Cette dernière sert à déterminer la position géographique précise d'un individu dans un environnement bien déterminé, et de naviguer vers des emplacements précis.

- Le troisième chapitre sous le nom « Analyse et Conception », est consacré à l'analyse et la conception de notre système à travers son architecture globale puis détaillée en définissant les différentes fonctionnalités attendues du système. Et pour permettre de détailler toutes les étapes de travail, une modélisation est faite avec le langage UML.

- Le quatrième chapitre sous le nom « Réalisation », consiste à présenter la réalisation du système. On commence par décrire l'environnement du travail et les outils de développement utilisés puis les différentes interfaces et une brève explication pour chacune d'elle pour donner une vue générale et complète sur le système.

Enfin, nous terminons par une conclusion générale qui résume l'apport essentiel de notre travail et qui tente de s'ouvrir sur un nouvel élément de réflexion qui est le développement de l'aspect sécurité et confidentialité du système.

Chapitre I
Etat de l’art d’Android

I.1. Introduction :

Par son ouverture et ses possibilités de déploiement, la plate-forme Google Android basée sur Linux offre un socle et un environnement de développement puissants pour créer des applications mobiles robustes et ergonomiques. Elle met à la portée des professionnels et des particuliers la réalisation d'applications à la fois riches en fonctionnalités et adaptées aux contraintes de l'utilisation mobile.
Rappelons les points clés d'Android en tant que plate-forme :

- elle est innovante car toutes les dernières technologies de téléphonie y sont intégrées : écran tactile, accéléromètre, GPS, appareil photo numérique etc. ;
- elle est accessible car en tant que développeur vous n'avez pas à acheter de matériel spécifique (si vous voulez aller plus loin que l'utilisation d'un émulateur, un téléphone Android pour effectuer vos tests vous sera toutefois nécessaire), ni à connaître un langage peu utilisé ou spécifique : le développement sur la plate-forme Android est en effet réalisé en langage Java, un des langages de programmation les plus répandus ;
- elle est ouverte parce que la plate-forme Android est fournie sous licence open source, permettant à tous les développeurs et constructeurs de consulter les sources et d'effectuer les modifications qu'ils souhaitent.

Ce chapitre est une introduction à la plate-forme, aux outils et à la configuration de notre environnement de travail.

I.2. Description : [1]

Android est un système d'exploitation pour Smartphones et tablettes tactiles conçu par Android. D'autres types d'appareils possédant ce système d'exploitation existent, par exemple des téléviseurs, des radioréveils ou des autoradios et même des voitures. Il a été développé par une petite startup qui fut acheté en 2007 par Google qui poursuit activement son développement avec l'Open Handset Alliance, Android est distribué sous licence open source depuis 2008. Ce système est assez nouveau auprès des programmeurs.

Selon Google qui est le majeur distributeur, Android est une plateforme puissante, moderne, sécurisée et ouverte. Elle est basé sur le kernel Linux 2.6 (noyau Linux 2.6) et utilisant la plateforme java pour ses applications. Elle est entièrement gratuite et sa plateforme très flexible ce qui permet aux développeurs d'intégrer, d'agrandir et de replacer les composants existants et d'adapter les applications aux besoins du client ou les remplacer entièrement, l'utilisateur peut donc personnaliser facilement son appareil.

De plus, il n'y a pas de distinction entre les applications natives et les applications qui sont développées par les développeurs, toutes sont disponibles sur l'**Android Market** (maintenant appelé **Google Play Store**).

En termes d'applications, Android a intégré plusieurs services de Google pour accéder rapidement aux services d'internet comme Gmail, You Tube, Google Talk, Google Calendar et Google Maps.

I.3. Open Handset Alliance :

L'Open Handset Alliance (abrégé OHA) est un consortium Regroupant de grands constructeurs et développeurs de logiciels dont le but est de développer des normes ouvertes pour les appareils de téléphonie mobile.

Le consortium a été créé le 5 novembre 2007 à l'initiative de Google qui a su fédérer autour de lui 34 compagnies. On note ainsi la présence d'opérateurs comme NTT DoCoMo, Sprint Nextel, Telecom Italia ou Bouygues Telecom. Des équipementiers tels que Samsung Electronics, Motorola Mobility ou LG Electronics. Des semi-conducteurs comme Intel ou Nvidia. Et enfin des incontournables de l'Internet dont eBay. On remarquera quelques grands absents dont Nokia.

I.4. Bugdroïde :

Le personnage nommé Bugdroid est le petit robot vert utilisé par Google pour présenter Android. Ce personnage est sous licence « creative commons by (3.0) » et peut donc être utilisé librement.

Le site Engadget annonce que Bugdroid, le logo d'Android, serait en fait un personnage d'un jeu des années 1990 sur Atari :Gauntlet: The Third Encounter.

Figure I.1 : Le personnage Bugdroid.

I.5. Historique des versions d'Android :

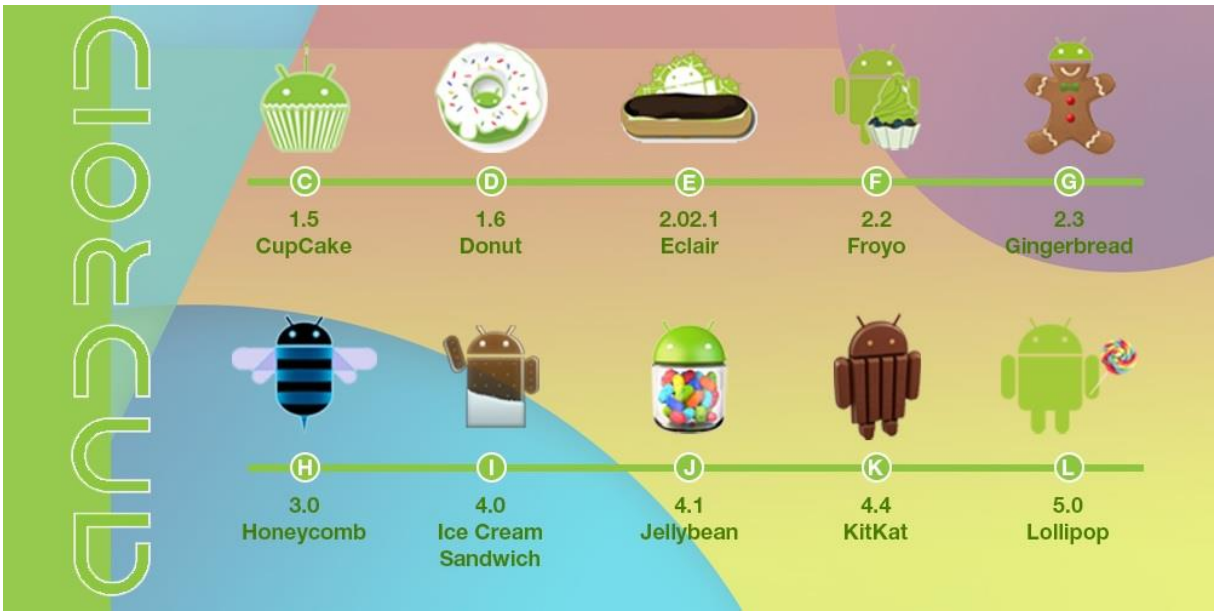

Figure I.2 : Evolution des versions d'Android.

Android a débuté avec la sortie de la version 1.0 « Apple pie » en septembre 2008. Android a connu plusieurs mises à jour depuis sa première version. Ces mises à jour servent généralement à corriger des bugs et à ajouter de nouvelles fonctionnalités.

Dans l'ensemble, chaque version est développée sous un nom de code basé sur des desserts. Ces noms de codes suivent une logique alphabétique, en voici quelques-unes :

Android 1.5 « Cupcake » a apporté de nombreuses améliorations. La première grande étape fut une mise à jour du noyau vers la version 2.6.27 qui a rendu le système plus stable et corrigea le manque d'API et rend le système plus utilisable.

Android 1.6 « Donut », **Android 2.0** et **2.1 « Eclair »** ont apporté d'importantes améliorations respectivement sur les fonctionnalités et sur l'interface graphique du système.

Android 2.2 « Froyo » a fortement mis l'accent sur la synergie avec Internet. L'envoi d'applications et de liens instantanés depuis un ordinateur est désormais possible. Aussi, Google annonce-t-elle que le navigateur chrome intégré à Android 2.2 est le navigateur mobile le plus rapide au monde grâce à l'intégration du moteur JavaScript V8.

Android 2.3 « Ginerbread » Dernière version dédiée uniquement aux smartphones. Cette version est parfois utilisée sur de petites tablettes.
Gingerbread était un système très réussi, qui devint à l'époque le système mobile le plus populaire.

Android 3.0 « Honeycomb » est spécialement étudié pour les tablettes tactiles. Les premiers modèles devraient être annoncés en 2011.

On y apprend quelques nouveautés comme la prise en charge de la vidéo-conférence via Gtalk, la nouvelle interface Gmail ou encore le lecteur de livre électronique Google.
La refonte graphique de l'interface utilisateur est assez réussie, plus d'informations devraient suivre dont surement des éclaircissements sur l'intégration ou non de l'interface de cette version d'Android sur les futurs smartphones.

Android 4.0 « Ice Cream Sandwich » devrait arriver très vite (10/ 2011) pour rajouter encore plus de fonctionnalités aux terminaux. Pour le développement, ces nouvelles versions d'Android devraient proposer de nouveaux composants permettant de réaliser des applications avec une ergonomie plus adaptée aux tablettes tactiles.

Android 3.0 et **Android 4.0** devraient apporter plus d'outils aux constructeurs leur permettant de proposer des tablettes tactiles, qui seront capables de rivaliser (surtout au niveau de l'ergonomie) avec Ipad.

Android 4.1 « Jelly Bean » Il ajoute un système de notification améliorée, la reconnaissance vocale sans connexion internet, et le « Project Butter » qui augmente la fluidité d'Android.

Android 4.4 « KitKat » Consommation en ressource moins élevée nécessitant moins de RAM, nouvelles icônes plus soignées, la barre du bas et celle de statut deviennent transparentes sur certains menus et changent de couleur en fonction du contenu affiché.

Android 5.0 «Lollipop »Les changements les plus importants d'Android 5.0 sont sa disponibilité sur les nouvelles plateformes Android TV et Android Auto, le renouveau de l'interface utilisateur nommée Material Design, l'amélioration du système de notifications, l'environnement d'exécution ART **(en)** qui remplace officiellement Dalvik pour améliorer les performances, ainsi que l'amélioration de l'autonomie de la batterie via le projet Volta[2].

I.6. Fonctionnalités d'Android : [2]

Android a été conçu pour intégrer au mieux les applications existantes de Google, comme le service de courrier Gmail, l'agenda Google Calendar ou encore la cartographie Google Maps.

Voici quelques fonctionnalités proposées par Android classées par version :

Version	Nom	API Level	Fonctionnalités
Android 1.0	Apple Pie (tarte aux pommes)	1	Fonctionnalités de base (téléphonie, SMS, réseaux de données, accès au répertoire, …etc)
Android 1.1	Banana Split	2	• corrections de bogues (alarme, Gmail...) • Amélioration des fonctionnalités MMS • Introduction des applications payantes sur l'Android Market

Android 1.5	Cupcake (petit gâteau)	3	• Nouveau clavier avec auto complétion • Support Bluetooth • Enregistrement de vidéos • Widgets • Animations améliorées
Android 1.6	Donut (Beignet)	4	• Recherche vocale améliorée • Indicateur d'utilisation de la batterie • Apparition de fonctionnalités pour les réseaux privés virtuels (VPN)
Android 2.0 ; 2.0.1 ; 2.1	Eclair (sorte de gâteau à la crème)	5 ; 6 ; 7	• Interface graphique améliorée • Gestion des contacts changée • Support d'HTML 5 • Support Bluetooth 2.1 • Clavier virtuel amélioré • Refonte de l'installation et des mises à jour du kit de développement • Réduction de la consommation de la batterie • Quelques améliorations graphiques comme l'écran de verrouillage • Nouveau bureau à 5 écrans virtuels • Fonds d'écran animés • Nouveaux effets 3D (notamment sur la gestion des photos) • Amélioration du client Gmail
Android 2.2	Froyo (yaourt glacé)	8	• Nouveaux Widgets • Gestion des points d'accès au réseau • Support multilingue
Android 2.3	Gingerbread (pain d'épice)	9	• Amélioration d'interface graphique • Téléphonie IP • Nouveau clavier virtuel
Android 3.0	Honeycomb (rayon de miel)	11	• Optimisé pour les tablettes et équipement à écran large • Multitâche et système de notification amélioré
Android 4.0	Ice cream sandwich (Sandwich à la crème glacée)	12	• Nouvelle interface graphique • Amélioration de la sécurité • Beaucoup de raccourcis (Appareil photo, accès sdcard, …etc)

Android 4.1	Jelly Bean (dragée)	16	• Interface utilisateur plus fluide • Amélioration de l'accessibilité • Notifications extensibles • Recherche vocale améliorée • Application appareil photo améliorée

Tableau I.1 : Fonctionnalités d'Android.

I.7. Architecture Android :

Le diagramme suivant illustre les composants principaux du système d'exploitation Android. Chaque section sera décrite dans ce qui suit :

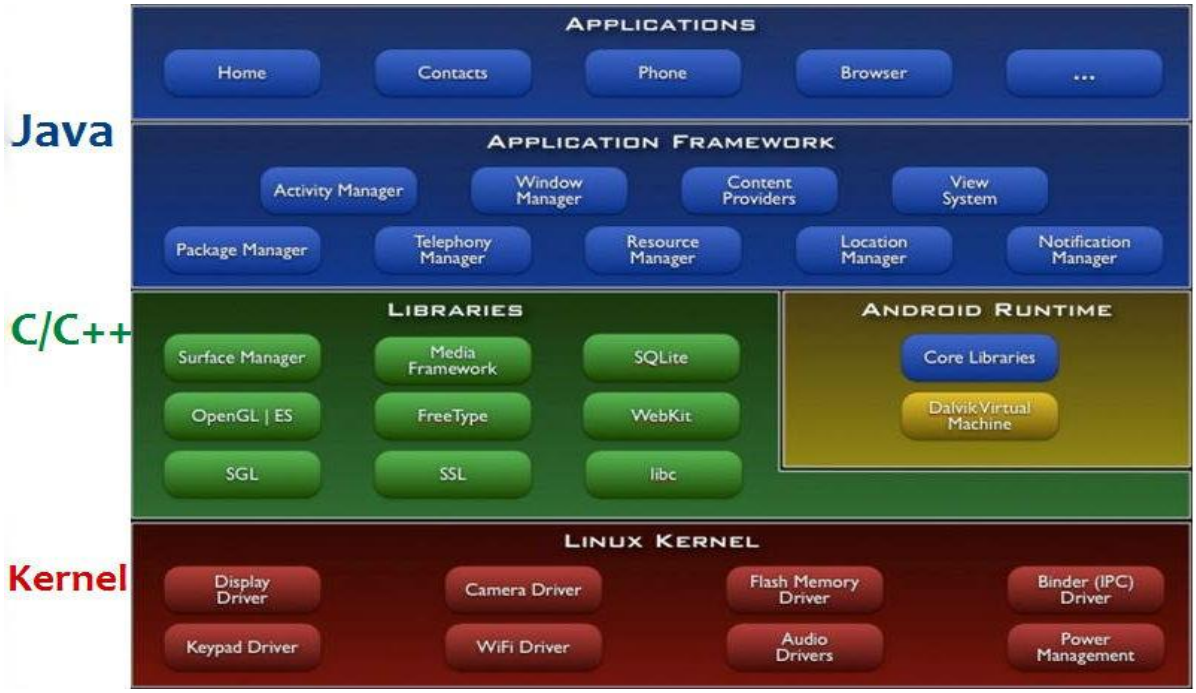

Figure I.3 : Architecture de l'Android

Android est basé sur un kernel linux 2.6.xx. Au-dessus de cette couche, on retrouve les librairies C/C++ utilisées par un certain nombre de composants du système Android.

Au-dessus des librairies, on retrouve l'Android Runtime. Cette couche contient les librairies cœurs du Framework ainsi que la machine virtuelle exécutant les applications.

Au-dessus de la couche "Android Runtime" et des librairies cœurs, on retrouve le Framework permettant au développeur de créer des applications. Enfin au-dessus du Framework, il y a les applications.

I.7.1. Applications :

Android est fourni avec un ensemble d'applications dont un client email, une application SMS, un calendrier, un service de cartographie, un navigateur… toutes écrites en JAVA.

I.7.2. Framework de développement :

En fournissant une plateforme de développement ouverte, Android offre aux développeurs la possibilité de créer des applications extrêmement riches et innovantes. Les développeurs sont libres de profiter du matériel périphérique et informations sur la localisation d'accès, exécuter des services d'arrière-plan, définir des alarmes, ajouter des notifications à la barre d'état, etc.

Les développeurs ont un accès complet au même framework API utilisé par les applications de base. L'architecture d'application est conçue pour simplifier la réutilisation des composants, n'importe quelle application peut publier ses capacités et n'importe quelle autre application peut alors faire usage de ces capacités (soumis à des contraintes de sécurité appliquées par le framework). Ce même mécanisme permet aux composants d'être remplacés par l'utilisateur.

Toutes les applications sous-jacentes forment un ensemble de services et de systèmes, y compris:

- Un jeu extensible de vues qui peuvent être utilisées pour construire une application.
- Des fournisseurs de contenu qui permettent aux applications d'accéder aux données d'autres applications (telles que les Contacts), ou de partager leurs propres données.
- Un gestionnaire de ressources.
- Un gestionnaire de notification qui permet à toutes les demandes d'afficher des alertes personnalisées dans la barre d'état.
- Un gestionnaire d'activité qui gère le cycle de vie des applications et propose une navigation commune.

I.7.3. Bibliothèques :

Android dispose d'un ensemble de librairies C / C++ utilisées par les différents composants du système Android. Elles sont offertes aux développeurs à travers le framework Android. En voici quelques-unes:

Système de bibliothèque C : une mise en œuvre dérivée de BSD de la bibliothèque C standard du système (libc), destinés aux systèmes embarqués basés sur Linux.

Comme cela a été dit précédemment, Android ne supporte pas la glibc, donc les ingénieurs d'Android ont développé une librairie C (libc) nommé Bionic libc. Elle est optimisée pour les appareils mobiles et a été développée spécialement pour Android.

Les ingénieurs d'Android ont décidé de développer une libc propre à la plateforme Android car ils avaient besoin d'une libc légère (la libc sera chargée dans chaque processus) et rapide (les appareils mobiles ne disposent pas de CPU puissant).

La Bionic libc a été écrit pour supporter les CPU ARM, bien que le support x86 soit présent. Il n'y pas de support pour les autres architectures CPU telles que PowerPC ou MIPS. Néanmoins, pour le marché des appareils mobiles, seulement l'architecture ARM est importante.

Cette libc est sous licence BSD (Berkeley Software Distribution License). Elle reprend une grande partie du code des glibc issue d'OpenBSD, FreeBSD et NetBSD.

- Ces caractéristiques importantes :

Elle pèse environ 200Ko, soit la moitié de la glibc L'implémentation des pthreads (POSIX thread) a été complètement réécrite pour supporter les threads de la machine virtuelle Dalvik. De ce fait, la Bionic libc ne supporte pas les threads POSIX .

Les exceptions C++ et les "wide char" ne sont pas supportés.

Médiathèques : basée sur PacketVideo de OpenCore; les librairies permettant la lecture et l'enregistrement audio et vidéo, ainsi que la gestion des fichiers image, y compris MPEG4, H.264, MP3, AAC, AMR, JPG et PNG.

Le schéma ci-dessous décrit tous les éléments de l'architecture de ces médiathèques:

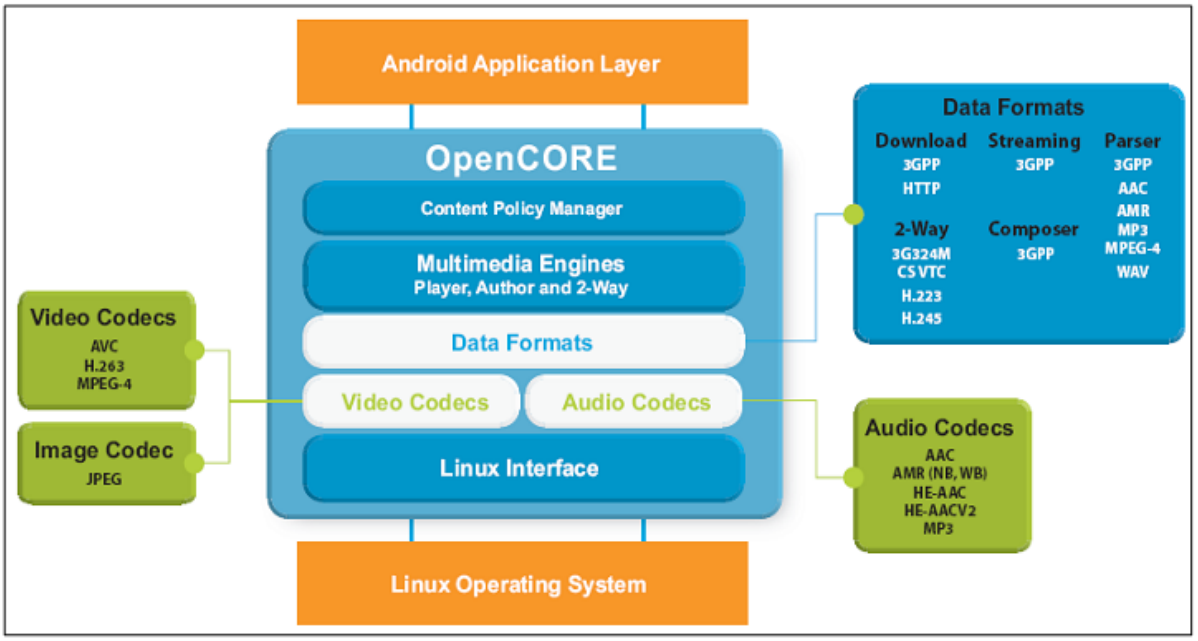

Figure I.4 : Architecture de ces médiathèques.

- Surface Manager : gère l'accès au sous-système d'affichage et de façon transparente.

- LibWebCore : le navigateur web présent dans Android est basé sur le moteur de rendu sous licence BSD WebKit.

- WebKit : est moteur de rendu, qui fournit une "fondation" sur laquelle on peut développer un navigateur web. Il a été originellement dérivé par Apple du moteur de rendu KHTML pour être utilisé par le navigateur web Safari et maintenant il est développé par KDE project, Apple, Nokia, Google et d'autres. WebKit est composé de deux librairies : WebCore et JavascriptCore qui sont disponibles sous licence GPL.

WebKit supporte le CSS, Javascript, DOM, AJAX. La dernière version a obtenu 100% au test Acid 3. La version de WebKit présent dans Android a été légèrement modifiée pour s'adapter aux appareils mobiles. Ainsi, le moteur de rendu basé sur WebKit présent dans Android supporte l'affichage sur une colonne.

- SGL : le moteur graphique 2D.
- Bibliothèques 3D : une implémentation basée sur OpenGL ES 1.0 API; les bibliothèques utilisent l'accélération 3D matérielle (si disponible).
- FreeType : bitmap et vectoriel de rendu de police.
- SQLite : un moteur de base de données relationnelle puissante et légère, disponible pour toutes les applications.

I.7.4. Android Runtime :

Android inclut un ensemble de librairies de base offrant la plupart des fonctionnalités disponibles dans les bibliothèques de base du langage de programmation Java.

Chaque application Android s'exécute dans son propre processus, avec sa propre instance de la machine virtuelle Dalvik. Dalvik a été écrit pour que le dispositif puisse faire tourner plusieurs machines virtuelles de manière efficace. La machine virtuelle Dalvik exécute des fichiers dans l'exécutable Dalvik (. DEX), un format optimisé pour ne pas encombrer la mémoire. La machine virtuelle est la base de registres et fonctionne grâce aux classes compilées par un compilateur Java et transformées dans le format DEX.

La machine virtuelle Dalvik s'appuie sur le noyau Linux pour les fonctionnalités de base telles que le filetage et la gestion de la mémoire de bas niveau.

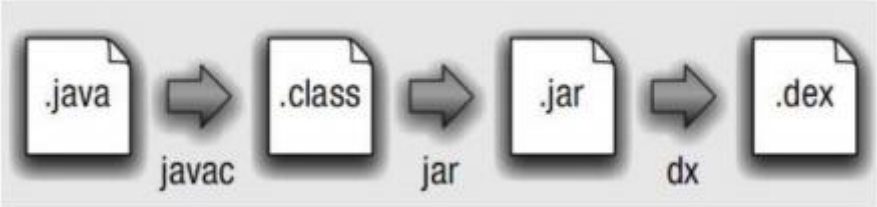

Figure I.5 : Transformation d'un fichier .java à un fichier .dex.

I.7.5. Linux Kernel :

Android est basé sur un kernel linux 2.6 mais ce n'est pas linux. Il ne possède pas de système de fenêtrage natif. La glibc n'étant pas supportée, Android utilise une libc customisée appelée Bionic libc.

Enfin, Android utilise un kernel avec différents patches pour la gestion de l'alimentation, le partage mémoire, etc. permettant une meilleure gestion de ces caractéristiques pour les appareils mobiles.

Android n'est pas linux mais il est basé sur un kernel linux. Pourquoi sur un kernel linux ?

Le kernel linux a un système de gestion mémoire et de processus reconnu pour sa stabilité et ses performances.

Le model de sécurité utilisé par linux, basé sur un système de permission, est connu pour être robuste et performant. Il n'a pas changé depuis les années 70.

- Le kernel linux fournit un système de driver permettant une abstraction avec le matériel. Il permet également le partage de librairies entre différents processus, le chargement et le déchargement de modules à chaud.

- Le kernel linux est entièrement open source et il y a une communauté de développeurs qui l'améliorèrent et rajoutent des drivers.

C'est pour les points cités ci-dessus que l'équipe en charge du noyau a décidé d'utiliser un kernel linux.

I.8. Les composants principaux d'une application :

Les applications Android sont composées de 4 types de composants :

I.8.1. Activités :

Une Activité représente un écran de l'application. Une application peut avoir une ou plusieurs activités (par exemple pour une application de messagerie on pourrait avoir une Activité pour la liste des contacts et une autre pour l'éditeur de texte). Chaque Activité est implémentée sous la forme d'une classe qui hérite de la classe Activity.

I.8.2. Services :

Les services n'ont pas d'interface graphique et tournent en tâche de fond. Il est possible de s'inscrire à un service et de communiquer avec celui-ci en utilisant l'API Android.

I.8.3. Broadcast receivers :

Il se contente d'écouter et de réagir aux annonces broadcast (par exemple changement de fuseau horaire, appel entrant...).

I.8.4. Content providers :

Il permet de partager une partie des données d'une application avec d'autres applications.

I.9. Cycle de vie d'une activité :

Une activité n'a pas de contrôle direct sur son propre état, il s'agit plutôt d'un cycle rythmé par les interactions avec le système et d'autres applications. Voici un schéma qui présente ce que l'on appelle le cycle de vie d'une activité, c'est-à-dire qu'il indique les étapes que va traverser notre activité pendant sa vie, de sa naissance à sa mort.

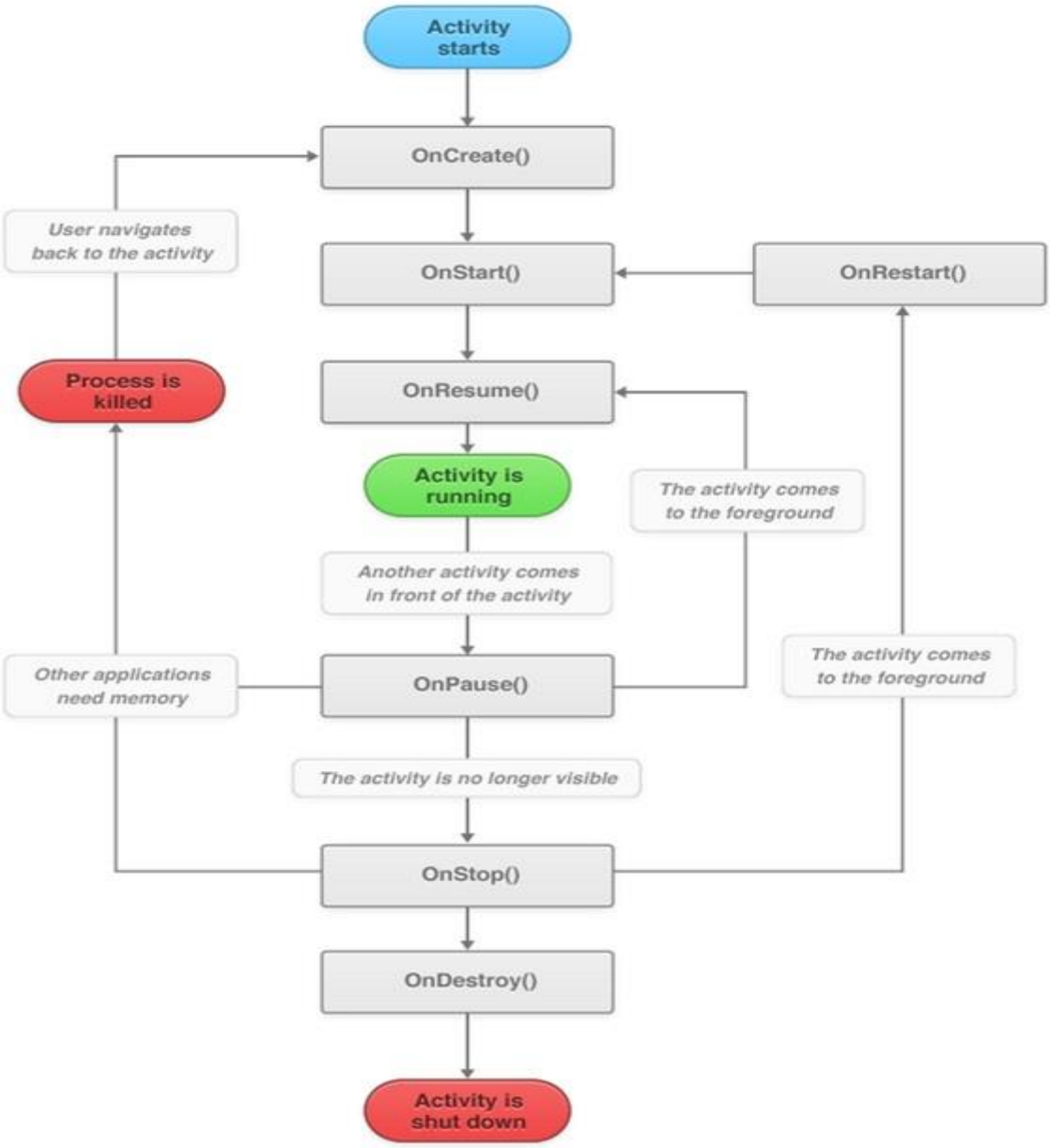

Figure I.6 : Cycle de vie d'une activité.

Une activité peut se trouver dans quatre états qui se différencient surtout par leur visibilité :

Active (Running) :
L'activité est visible en totalité.
Elle est sur le dessus de la pile, c'est elle qui a le *focus*. C'est ce que l'utilisateur consulte en ce moment même et il peut l'utiliser dans son intégralité et agir directement dessus.
Paused (en pause) :
L'activité est partiellement visible à l'écran.
C'est le cas lors de la réception d'un SMS et qu'une fenêtre semi-transparente se pose devant l'activité pour afficher le contenu du message et permettre d'y répondre par exemple.

Ce n'est pas sur cette activité qu'agit l'utilisateur. L'application n'a plus le focus, c'est l'application sous-jacente qui l'a. Pour que notre application récupère le focus, l'utilisateur devra se débarrasser de l'application qui l'obstrue, puis il pourra à nouveau interagir avec.

Stopped (Arrétée) :

L'activité n'est pas visible à l'écran.

Si une activité est complètement masquée par une autre activité, il est arrêté. Il conserve tous les états membres et de l'information, cependant il n'est plus visible pour l'utilisateur que sa fenêtre est cachée et il sera souvent tués par le système lorsque la mémoire est nécessaire ailleurs.

Dead (Mort) :

Cette activité a terminé ou il n'a jamais été démarré.

Si une activité est en pause ou arrêtée, le système peut chuter l'activité de la mémoire, soit par lui demandant de se terminer, ou tout simplement tuer le processus. Quand il est affiché de nouveau à l'utilisateur, il doit être redémarré et restauré à son état antérieur.

Les transitions d'états sont captées par les méthodes suivantes :

onCreate() : l'activité est en création
onStart() : l'activité va devenir visible
onResume() : l'activité est maintenant visible
onPause() : l'activité va être mise en pause
onStop() : l'activité ne sera plus visible
onDestroy() : l'activité va être détruite

I.10. Développement d'une application Android :

Voici quelques étapes principales dans le processus de développement d'une application sur Android :

- Faire la conception de la base de données.
- Créer des classes pour représenter les données physiques (couche Mapping) et pour définir des actions comme : supprimer, ajouter, modifier des données.
- Dessiner des interfaces en les fichiers XML ou en codage :
  - ✓ Les vues (View) : Text, Edit, List, Image, Web, Map, etc.
  - ✓ Les arrangements (layout) : Frame, Linear, Relative, Table, Absolute.
- Choisir des arrangements (layout) : Les layouts sont les ressources qui indiquent les interfaces des activités. On utilise les fichiers XML pour exprimer les interfaces.
  Mais il existe d'autre technique pour dessiner l'interface. Dans cette technique, on programme directement les composants graphiques en utilisant le codage.
- Organiser des ressources: les constantes globales (string.xml), les icônes, les images, etc.
- Créer et mettre à jour le fichier de configuration : AndroidManifest.xml. AndroidManifest.xml (configuration de l'application) est utilisé pour stocker les dispositions (settings) globales comme les permissions de l'application, les activités, les filtres de l'intention.

Figure I.7 : Une application sur Android.

- Créer des activités (Créer les classes pour exécuter les fonctions avec la base de données (ajouter, supprimer, modifier, mettre à jour, etc.) :
  - ✓ Chaque activité peut correspondre avec un écran ou une fonction de cette application.
  - ✓ Il existe quelques activités qui s'occupent des méthodes pour communiquer avec la base de données (Couche Mapping).
  - ✓ Les Intents sont utilisées pour orienter des activités (CALL, ACTION_MAIN, ACTION_VIEW, etc).

I.11. Prise en main de l'environnement Android :

La première étape de notre travail avec l'environnement Android a été d'appréhender le SDK, l'architecture et le développement d'une application ainsi que son déploiement sur un terminal embarquant Android.

I.11.1. Présentation du SDK :

Google a mis en place un grand nombre d'outils pour aider les développeurs Android.

- ❖ **Le portail des développeurs**

La première chose à visiter est le portail des développeurs Android, mis en place par Google. [6].

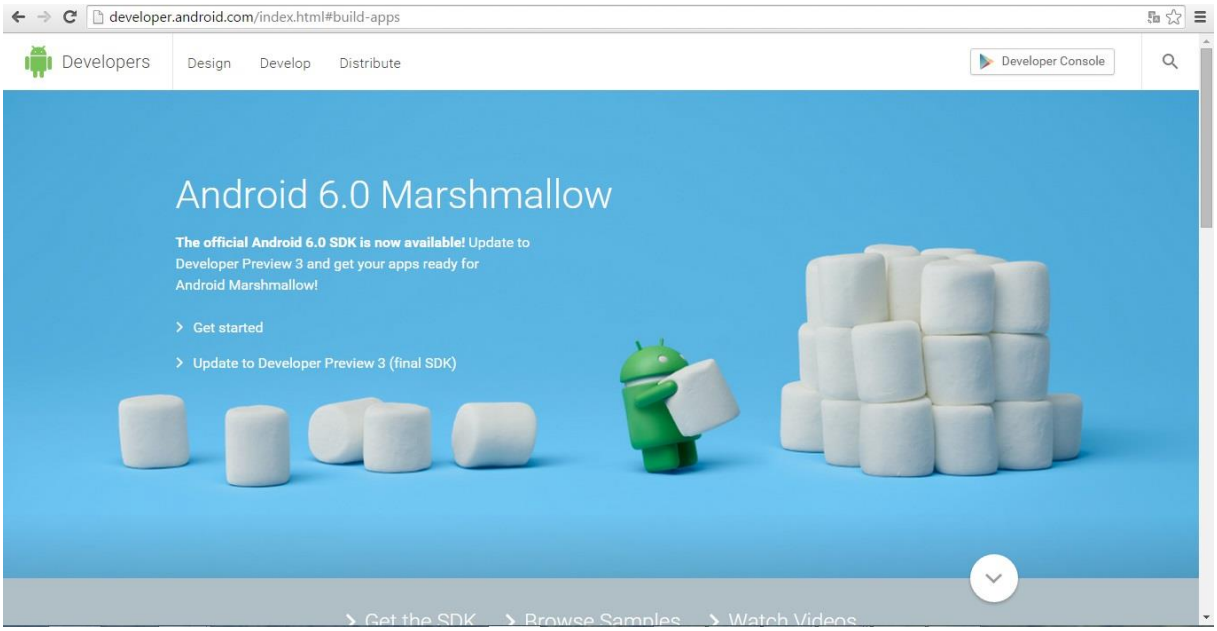

Figure I.8 : Portail des développeurs Android [6]

Très complet, ce site présente Android, explique comment installer et utiliser les différents outils (SDK, NDK etc.), propose un ensemble de tutoriels et articles concernant le développement d'applications Android, expose la référence de l'API Android ainsi que les actualités liées à Android.

Le tout est très bien fait et permet de rapidement être confortable vis-à-vis du développement sur Android.

I.11.2. Le SDK Android : [7]

L'outil le plus important est le SDK Android. Facile à installer, il permet de télécharger tous les outils indispensables au développement d'applications. Un petit logiciel permet d'abord de télécharger les différentes versions du SDK (une version du SDK par version d'Android : 1.4, 1.5, 1.6, 2.0 etc.). Il permet également de télécharger les différentes versions des Google APIs (APIs pour intégrer des fonctionnalités liées aux services Google tels que Maps etc.) ou de la documentation JavaDoc. Son fonctionnement est similaire aux gestionnaires de paquets de Linux.

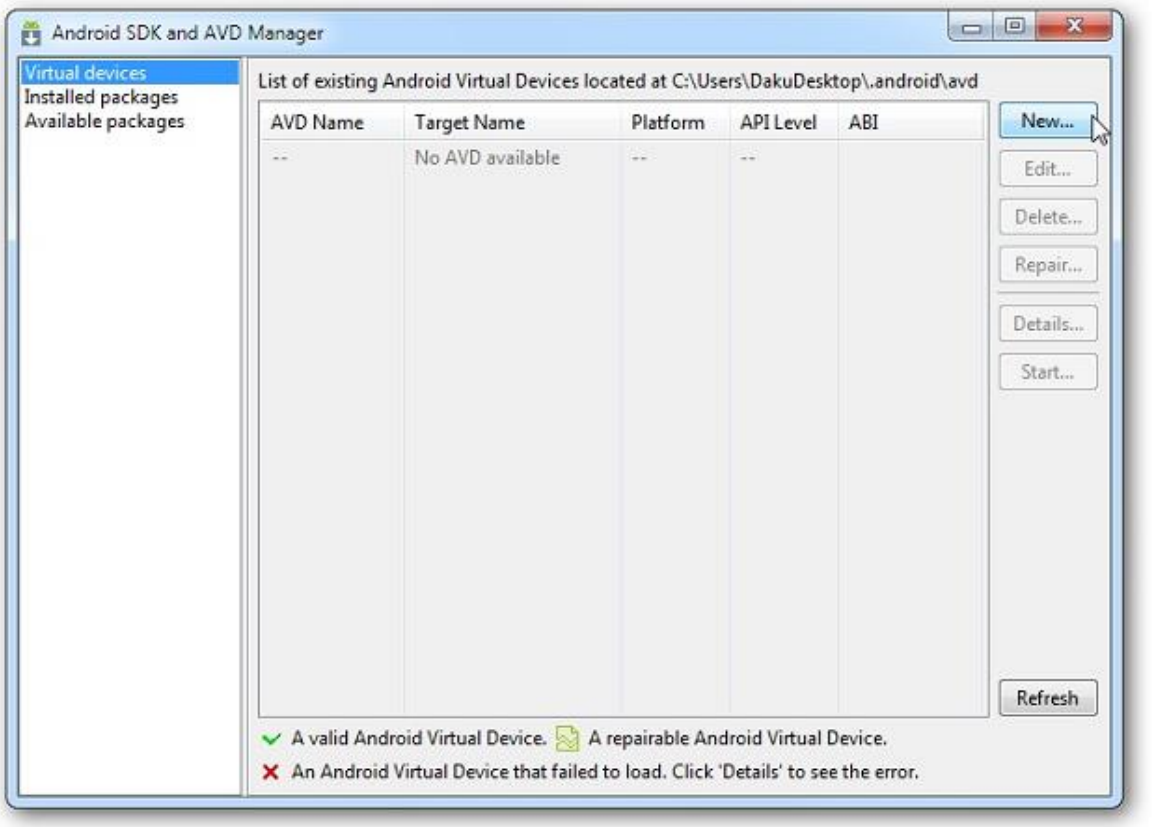

Figure I.9 : Interface d'installation du SDK Android.

I.11.3. ADT pour Eclipse :

Eclipse est l'Environnement de Développement Intégré (ou IDE) le plus largement utilisé pour la programmation Java, très performant, il est de plus gratuit et open source.

Figure I.10 : Logo Eclipse et site web.

Le langage privilégié pour le développement d'applications Android est justement Java. Google a donc tout naturellement conçu un plugin pour Eclipse (un plugin est un module qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités).

Android Development Tools, ou ADT, est très complet et surtout très pratique : conception graphique d'interfaces utilisateur, debug distant sur un téléphone, gestion de l'architecture de fichiers d'une application etc.

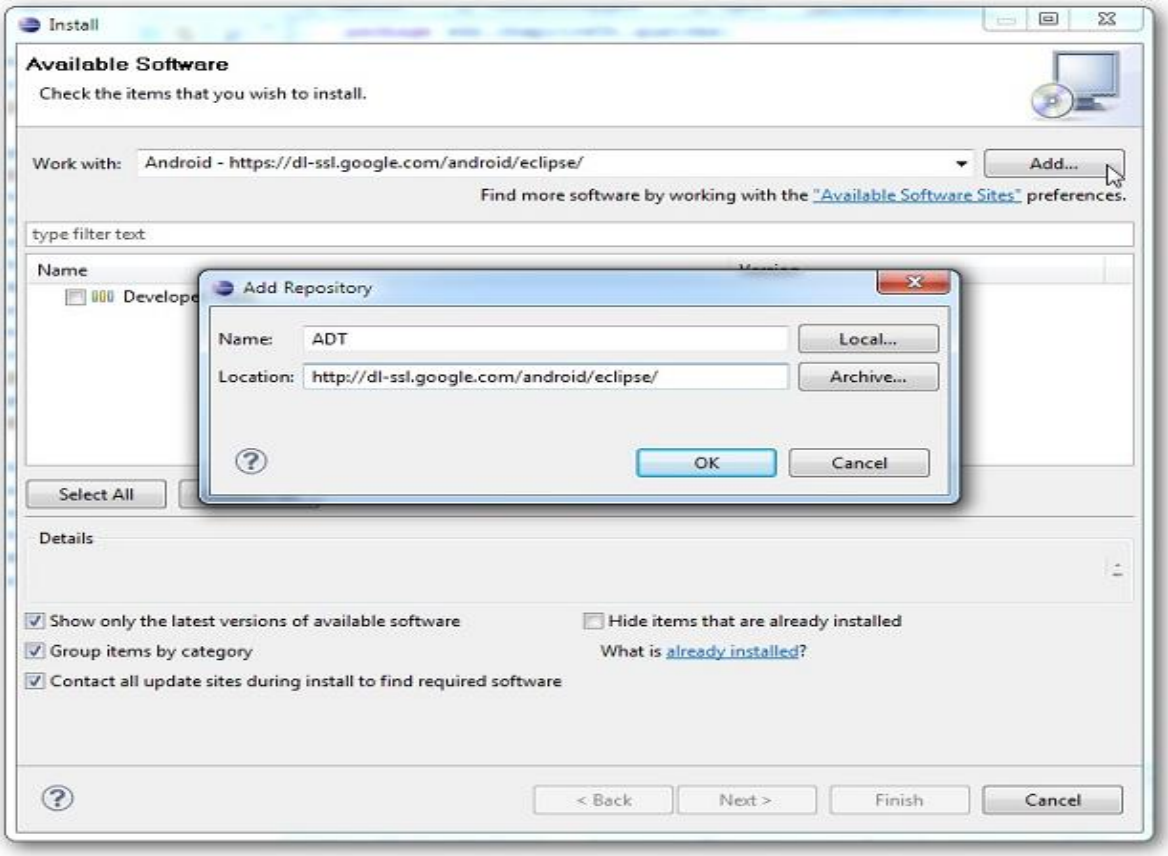

Figure I.11 : Interface d'installation ADT.

I.11.4. Emulateur :

Nous l'avons évoqué plus haut, le SDK propose un émulateur Android. Il permet de lancer sur la machine du développeur un terminal virtuel représentant à l'écran un téléphone embarquant Android. C'est bien évidemment un outil indispensable pour le développement mobile. A chaque version d'Android est associée une version de l'émulateur, permettant au développeur de voir exactement à quoi ressemblera son application sur un matériel réel.

Rappelons cependant que l'émulateur ne propose pas toutes les fonctionnalités d'un vrai téléphone. Il ne permet par exemple pas d'émuler la gestion du Bluetooth.

Figure I.12 : Interface du simulateur Android.

I.12. Conclusion

Dans ce chapitre, nous avons fait une étude de l'état de l'art d'Android tout en présentant un bref historique, les fonctionnalités que nous pouvons trouvées sur ce système d'exploitation et l'architecture d'Android, à savoir les principaux composants du système.

En fin nous avons présenté l'environnement software et hardware utilisé pour la programmation Android.

Chapitre II
Géolocalisation

II.1. Introduction :

La géolocalisation, un terme encore flou pour beaucoup d'entre nous, peu d'articles se consacrent à son aspect technique, nous avons donc choisi de donner une description de cette nouvelle technologie qui fait parler d'elle de plus en plus. Cette dernière sert à déterminer la position géographique précise d'un individu dans un environnement bien déterminé.

Les premiers services de géolocalisation commençaient à faire leur apparition sur le marché depuis quelques années dans les secteurs militaire et civil.

Étant un marché porteur, ces technologies sont un enjeu stratégique et commercial aussi bien pour les opérateurs de communications électroniques que pour les grands pays de ce monde.

II.2. Techniques de géolocalisation : [5]

Pour géolocaliser l'utilisateur grâce à son Smartphone, il existe plusieurs techniques que l'on recoupe afin d'obtenir un résultat à la fois rapide et précis.

II.2.1. Géolocalisation par satellite :

La géolocalisation par satellite consiste à calculer, grâce aux signaux émis par une constellation de satellites prévue à cet effet, la position actuelle sur la face terrestre d'un terminal équipé d'une puce compatible.

Cette position est alors traduite en termes de latitude, longitude et parfois altitude et peut alors être représentée physiquement sur une carte. Le réseau satellite de positionnement le plus connu est le GPS (Global Positionning System). La géolocalisation par satellite est la méthode utilisée pour la navigation GPS en voiture.

Dans le cas du GPS, pour que le repérage spatial fonctionne, un réseau constitué d'une vingtaine de satellites tourne autour de la Terre à une altitude de 20 200 km, répartis sur 6 orbites (4 par orbite). Ces satellites constituent un maillage du ciel et servent de repères aux navigateurs GPS dans leur processus de calcul de position. Ce système de satellites est conçu de façon à ce qu'il y en ait toujours au moins quatre « visibles » par les navigateurs GPS, sans quoi la position ne peut pas être déterminée.

Pour qu'un terminal soit capable de se géolocaliser grâce au réseau GPS, celui-ci doit être équipé d'une puce électronique GPS, puce que l'on retrouve dans tous les Smartphones actuels.

Le GPS offre une précision allant de 15 à 100 mètres pour les applications civiles mais possède comme inconvénients une impossibilité d'utilisation à l'intérieur et un allumage assez long.

II.2.2. Géolocalisation par GSM :

Les téléphones mobiles sont connectés en permanence à des antennes GSM puisque c'est grâce à ces antennes que les appels et les envois de SMS peuvent être effectués. Grâce à la triangulation, c'est-à-dire avec trois antennes différentes, le Smartphone peut être repéré par sa position.

Ces antennes captent le signal émis par le téléphone et le calcul de la position peut être effectué par plusieurs méthodes :

- ✓ en fonction du temps que le signal met pour atteindre l'antenne.
- ✓ en fonction de l'angle d'arrivée et de la force du signal.
- ✓ en fonction des identifiants des antennes GSM auxquels un terminal se connecte, cette méthode, appelée Cell ID, est couramment utilisée. Une base de données fait le lien entre les identifiants des antennes et leur position.

La précision de la géolocalisation par GSM se situe entre 200 mètres et plusieurs kilomètres, c'est pour cela que cette technique ne peut pas être utilisée seul pour géolocaliser un utilisateur et son Smartphone. De plus, la couverture géographique n'est pas complète en Algérie et dans le monde.

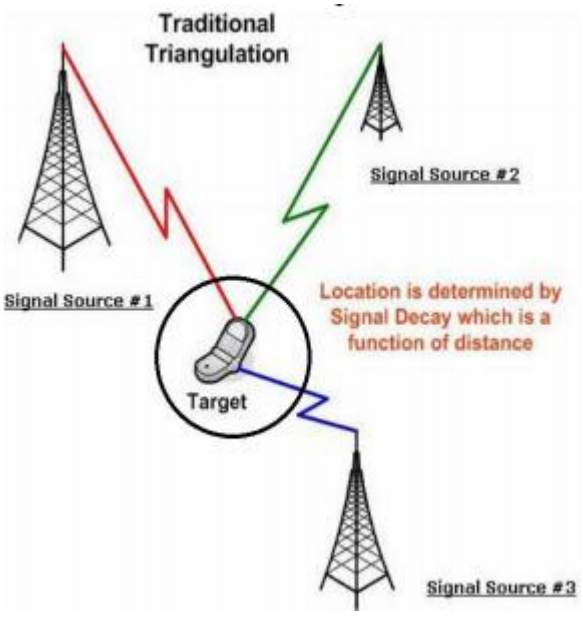

Figure II.1 : géolocalisation par GSM.

II.2.3. Géolocalisation par Wifi : [3]

Cette technique reprend le même principe que la localisation GSM qui utilise la méthode Cell ID. La géolocalisation se fait en utilisant l'identifiant des bornes d'accès Wifi

que le Smartphone détecte et en recoupant les données. Cependant, cette technique a un défaut important car la présence de borne Wifi en zone rurale est faible.

II.2.4. Utilisation des méthodes mixtes de géolocalisation :

Pour faire face aux inconvénients de chaque technique de géolocalisation, il y a actuellement une technique qui combinent les trois précédentes et qui est capable de géolocaliser le Smartphone dans n'importe quelle situation possible. La précision de ce positionnement va varier en fonction des technologies disponibles, mais le temps de réponse à l'allumage et l'adaptabilité s'en verront améliorées.

Grâce à cette utilisation multiple qui permet de géolocaliser une personne à l'extérieur en utilisant le GPS, et de le suivre à l'intérieur des bâtiments ou des tunnels en utilisant la technologie GSM couplée au Wifi pour plus de précision.

Figure II.2 : géolocalisation mixte.

II.2.5. Géolocalisation indoor :

Comme vu précédemment, plusieurs technologies peuvent être employées pour géolocaliser un terminal. Et cette couverture fonctionne très bien en "outdoor" (à l'extérieur). En revanche, de par la nature des signaux émis, ces systèmes sont inefficaces en "indoor" (à l'intérieur). Il existe cependant un moyen de se géolocaliser à l'intérieur même d'un bâtiment. Il s'agit de placer des bornes Wi-Fi dans le bâtiment en question pour émettre des signaux Wi-Fi permettant ainsi la géolocalisation des Smartphones. C'est

ce qui a été mis en place au sein du centre commercial de 4 étages "Les quatre temps" à Paris. Une application Smartphone permet de se localiser précisément dans le centre et nous permet de trouver facilement le chemin à emprunter pour se rendre à la boutique désirée.

La géolocalisation indoor va très certainement être amenée à se développer dans l'avenir car les possibilités d'exploitations sont très importantes.

Figure II.3 : géolocalisation Indoor.

II.3. Technologies de positionnement : [4]

Il y a plusieurs méthodes pour localiser le MS[1] (Mobile station). Ces méthodes sont basées sur la transmission de certains signaux et leur réception à l'autre bout.

La technologie de positionnement utilisée est choisie selon les besoins (par exemple le temps de réponse) des applications. Parmi les différentes technologies de positionnement existantes, nous avons :

[1] MS est un terminal mobile : le téléphone portable, un PDA...

II.3.1. Global Positioning System (GPS) :

Le GPS (système de positionnement global) permet de déterminer des positions en 3 dimensions en tout lieu du globe et à toute heure (sauf exceptions). Il est actuellement le principal système mondial de positionnement par satellite, et le seul à être entièrement opérationnel (contrairement à GLONASS [russe], Beidou [chinois] et Galileo [européen]).

II.3.1.1. Histoire (brève) du GPS :

Le GPS a été conçu au début des années 70 par le département de la Défense des Etats-Unis, mais la constellation de 24 satellites opérationnels a été complétée en 1993. À l'origine, le GPS était destiné à répondre aux besoins des militaires, mais la Présidentiel Décision Directive du 28 mars 1996 s'achevait par la déclaration suivante :

« Nous continuerons de fournir en permanence le service de positionnement standard GPS à des fins pacifiques d'utilisation civile, commerciale et scientifique à l'échelle mondiale et sans imposer de redevances directes aux utilisateurs ». [Traduction]

Le GPS offrait toutefois deux niveaux de service : un service de positionnement standard (SPS) accessible à tout utilisateur, et un système de positionnement précis (PPS) principalement réservé aux militaires américains. Le 1er mai 2000, le président Bill Clinton annonça qu'il mettait fin à cette dégradation volontaire du service. Aujourd´hui, le GPS est utilisable sans frais, par tous, et n'importe où dans le monde. Environ 80% des utilisateurs sont désormais des civils et profitent du système avec une précision de l'ordre de 20 mètres dans le plan horizontal, 95% du temps.

II.3.1.2. Composition du GPS :

Dans le langage courant, on utilise le mot « GPS » pour désigner le récepteur. Or « GPS » désigne le système de positionnement en entier, et ce système se compose de trois éléments : le **segment spatial**, le **segment contrôle** et le **segment utilisateur**.

II.3.1.2.1. Segment spatial :

Le segment spatial est composé d'un réseau de 28 satellites en orbite quasi-circulaire autour de la terre, à une hauteur à peu près de 20000 km, dont la période de révolution est de 12 heures sidérales. Ces satellites sont répartis sur 6 plans orbitaux inclinés 55° par rapport à l'équateur. Cette répartition spatiale garantit la visibilité en permanence d'au moins six satellites en tout point du globe. La durée de vie minimale du système est de 7 ans.

Il existe trois catégories de satellites GPS:

Bloc I: Satellites lancés entre 1978 et 1985.Tous les satellites du Bloc I sont maintenant hors-service excepté un seul, qui est activé de manière périodique. Leur durée de vie est de 4,5 ans. La principale différence entre ces satellites et les générations suivantes est l'impossibilité de dégrader volontairement le signal transmis.

Bloc II : Seconde génération de satellites GPS mis sur orbite à partir de 1985. Ils ont la capacité de dégrader le signal émis et leur durée de vie est de 7,5 ans.

Bloc III: ils ont été construits pour avoir une durée de vie de 10 ans. Ils sont capables de communiquer entre eux et ont été mis sur orbite depuis 1996 afin de maintenir une constellation complète.

Figure II.4 : Segment spatial.

II.3.1.2.2. <u>Le segment de contrôle</u> :

Le segment de contrôle comprend cinq stations de poursuite situées à Colorado Springs, Hawaii, Ascension Island, Diego Garcia et Kwajalein. Ces stations sont les yeux et les oreilles du système GPS, elles vérifient l'état des satellites lors de leur passage au-dessus d'elles. Ces stations transmettent ensuite leurs données à la station principale de Colorado Springs. C'est là que les paramètres décrivant l'orbite des satellites et la qualité des horloges embarquées sont estimés, la vérification de la santé des satellites et la détermination d'un repositionnement éventuel sont également contrôlés. Cette information est alors renvoyée à trois stations de chargement qui transmettent l'information aux satellites. Grâce à la répartition uniforme des stations de contrôle, tous les satellites GPS sont captés à 92% du temps.

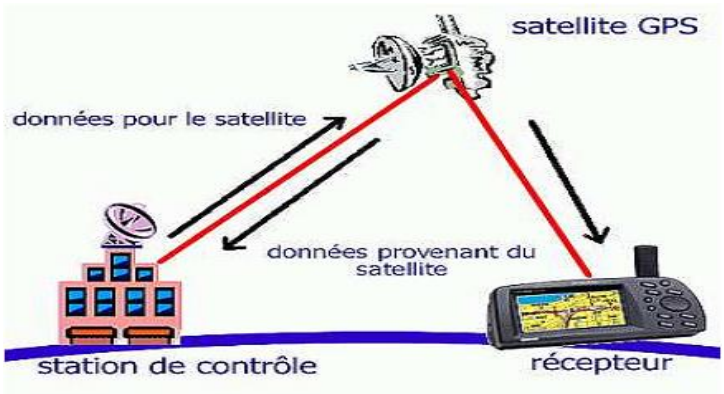

Figure II.5 : Le Segment de contrôle.

II.3.1.2.3. Segment utilisateur :

Le segment utilisateur est constitué de récepteurs qui ont été conçus afin de décoder le signal transmis par les satellites pour déterminer la position, la vitesse et le temps de l'utilisateur.

Figure II.6 : Segment utilisateur.

II.3.1.3. Principe de Fonctionnement du GPS :

Notre observateur en possession d'un récepteur GPS reçoit les informations de tous les satellites de la constellation qui sont en visibilité. Les informations reçues permettent au récepteur GPS de donner une position par rapport à un référentiel appelé ECEF (Earth Centered, Earth Fixed).

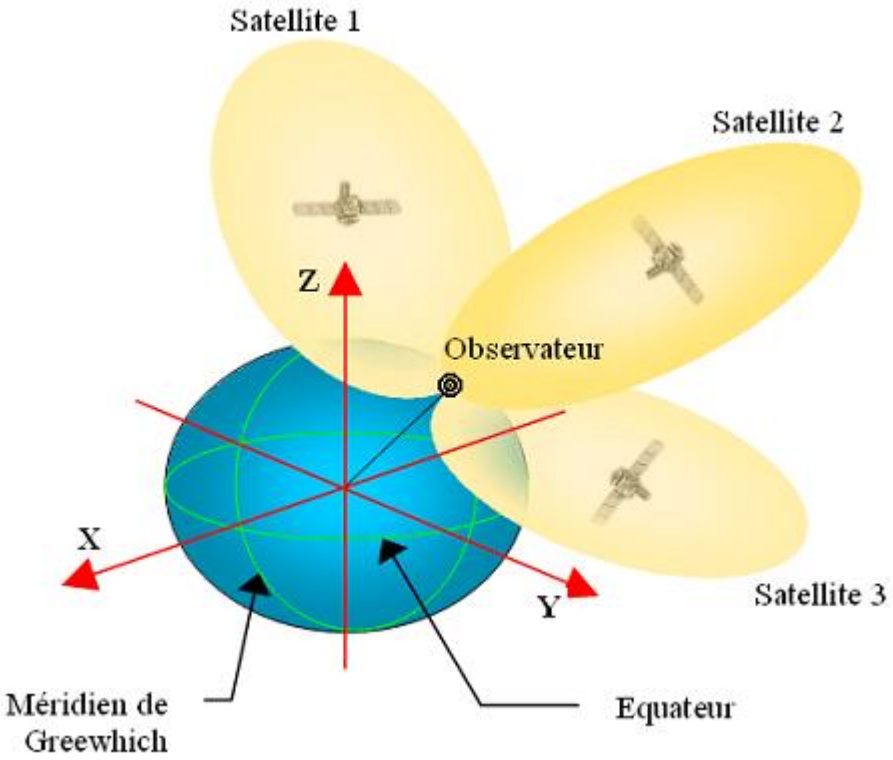

Figure II.7 : Fonctionnement du GPS.

Les informations envoyées par chaque satellite sont entre autre :

- La position exacte du satellite dans le système ECEF
- Le moment exact où le signal a été envoyé

Le récepteur lui analyse l'écart temporel entre l'émission du signal et la réception puis, grâce à un savant calcul il détermine sa distance par rapport aux différents satellites. Sa position sera par conséquent à l'intersection de toutes les sphères dont leur rayon sera égal à la distance satellite récepteur.

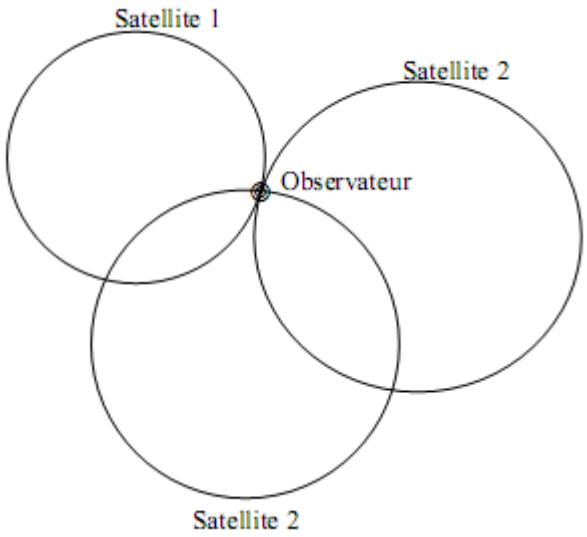

Figure II.8 : positionnement de l'observateur à l'intersection des 3 satellites.

Mais les choses n'étant pas parfaites la réalité sera tout autre. Notre observateur sera en fait quelque part dans la zone commune aux trois satellites.

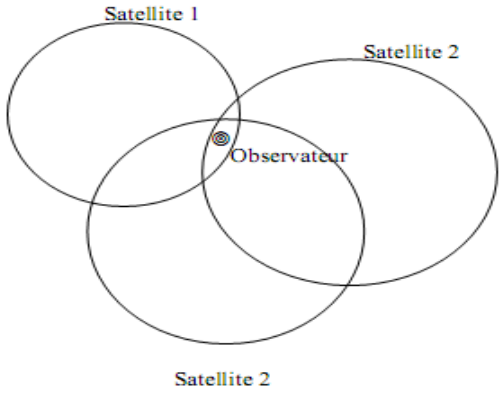

Figure II.9 : positionnement de l'observateur dans la zone commune des 3 satellites.

Quelques chiffres clef :

- La constellation GPS est constituée de 24 satellites
- Orbite elliptique presque circulaire de 26000 Km de rayon
- Le tour complet de l'orbite est effectué en 12 heures environ.

II.3.1.4. Les erreurs de positionnement :

Bien que la position soit donnée avec une grande précision, il faut tenir compte des erreurs du système GPS. Sont principalement à l'origine de ces erreurs :

- Freinage des ondes électromagnétiques dans l'ionosphère (5 à 100 Km d'altitude)
- Freinage des ondes électromagnétiques dans la troposphère (0 à 50 Km d'altitude)
- Erreur de synchronisation des horloges des satellites et du récepteur GPS
- Plus diverses raisons comme l'effet relativiste, réflexion des ondes etc...
- Le nombre de satellites en visibilité et leurs répartitions dans le ciel ont une influence sur la précision de la position.

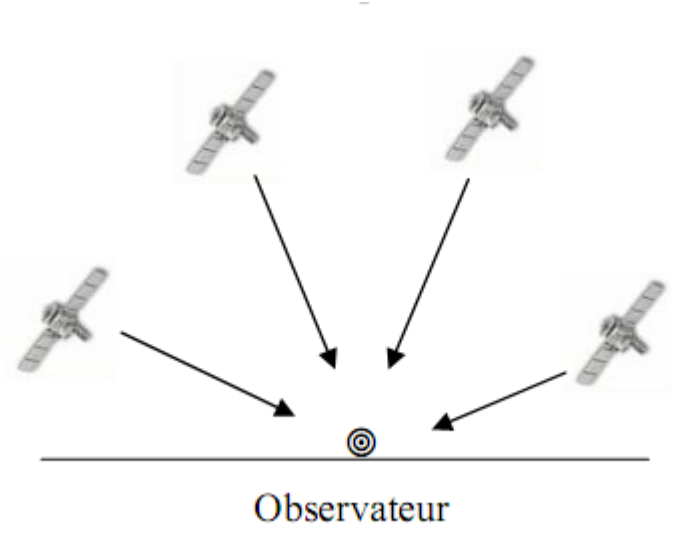

Figure II.10 : Bonne répartition.

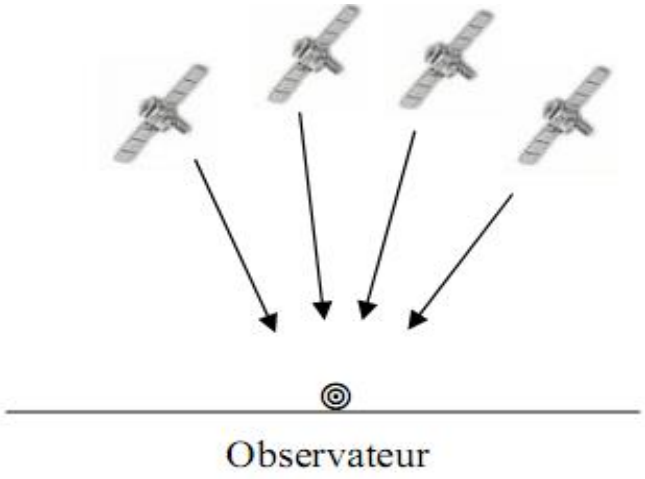

Figure II.11 : Mauvaise répartition.

- Le relief ainsi que la végétation perturbent la réception du signal.

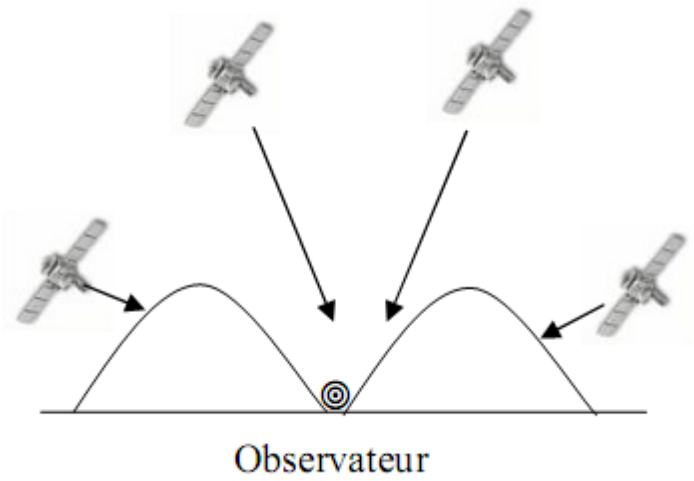

Figure II.12 : Dans une vallée, Mauvaise réception.

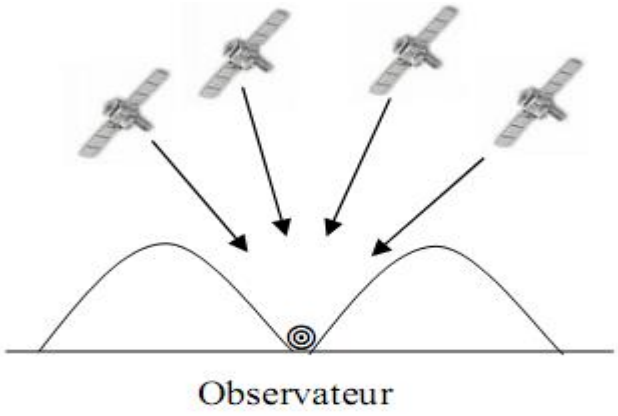

Figure II.13 : Dans une vallée, Meilleure réception.

Le récepteur GPS fourni en général la synthèse de ces informations en affichant la précision de la position. Une page particulière permet de visualiser la distribution des satellites ainsi que la qualité du signal.

Il est possible sur certains récepteurs d'utiliser le système WAAS (Wide Aera Augmentation System). Si le satellite WAAS est en vue, la précision du récepteur sera améliorée.

Ce système DGPS permet de corriger les erreurs des satellites par comparaison de ces erreurs par rapport à une base référentielle terrestre.

Ordre de grandeur des erreurs :

- ✓ Données des éphémérides 2m
- ✓ Traversée de l'ionosphère 4m
- ✓ Traversée de la troposphère 1m
- ✓ Réflexion du signal sur les obstacles 2m
- ✓ Horloges satellites 2m

II.3.1.5. <u>Le système de coordonnées locales</u> : [5]

Le système de coordonnées utilisé par le système GPS (ECEF) est sans réel intérêt pour l'utilisateur. Une conversion dans un système plus pratique est nécessaire.

Le système le plus approprié est le système E, N, U (East, North, Up) qui s'exprime en général par :

- ✓ La latitude Φ
- ✓ La longitude λ
- ✓ L'altitude h
- ✓ N est la distance mesurée le long de la droite localement perpendiculaire à l'ellipsoïde entre l'observateur et l'intersection avec l'axe polaire

Il faut également tenir compte que la terre n'est pas une sphère mais un ellipsoïde dont les demi grands axes s'expriment par les valeurs a et b.

Le système ENU le plus couramment utilisé est le système WGS84 (World Geodic Survey 1984).

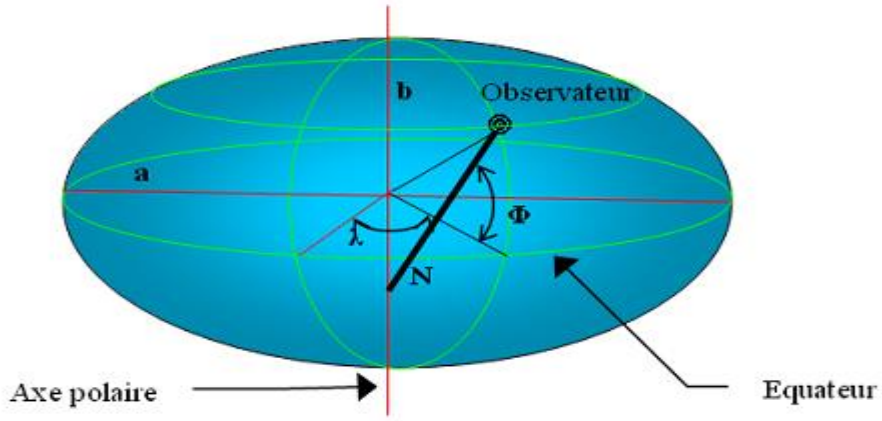

Figure II.14 : Système de coordonnées locales.

II.3.1.6. Les projections [5]

La terre est un ellipsoïde qui est par définition non développable. Il est nécessaire de trouver des artifices afin de pouvoir représenter la surface terrestre sur une feuille de papier qui elle est plane. Le principe de base est de projeter la surface de la terre sur une surface qui elle est développable (cône ou cylindre). Il y a deux types de représentations, elles sont caractérisées par la façon de représenter un cercle.

- ✓ La représentation conforme : l'image d'un cercle reste un cercle, les angles sont conservés.
- ✓ La représentation équivalente : la surface d'un cercle est représentée par une ellipse de même aire, les surfaces sont conservées.

Vous trouverez ci-après deux exemples de projections, les coniques et les cylindriques. Ce ne sont pas les seuls types de projections, il en existe une grande quantité mais ce sont les plus utilisés.

- ❖ Les représentations coniques.

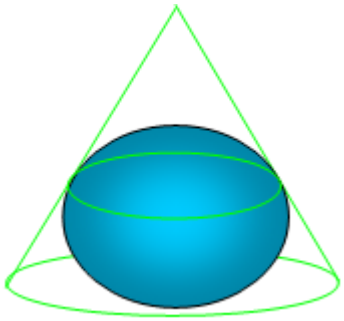

Figure II.15 : Directe tangente.

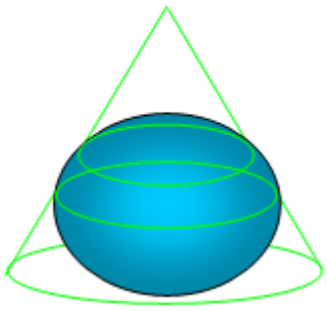

Figure II.16 : Directe sécant

Exemple de représentation conique :

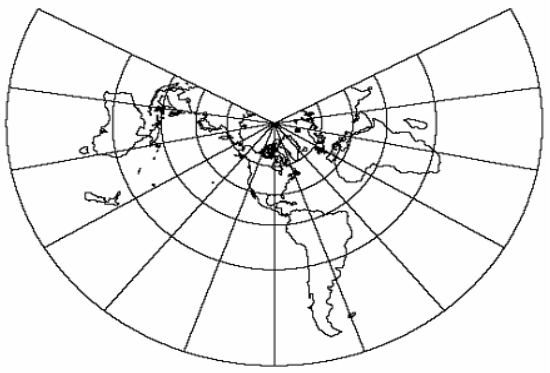

Figure II.17 : représentation conique.

- Les représentations cylindriques.

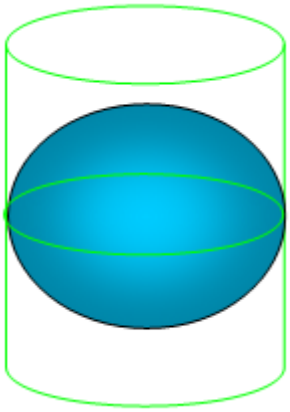

Figure II.18 : Directe.

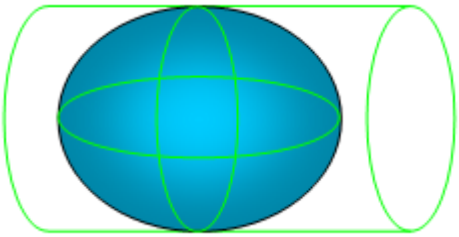

Figure II.19 : Transverse.

Exemple de représentation cylindrique :

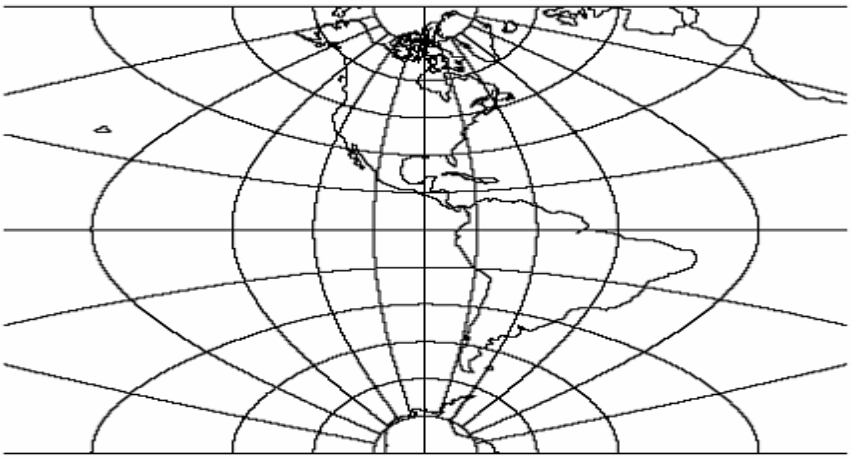

Figure II.20 : représentation cylindrique.

❖ La projection Universal Transverse Mercator (UTM).

La projection cylindrique UTM couvre le monde entier et est constituée de 60 fuseaux de 6 degrés d'amplitude en longitude.

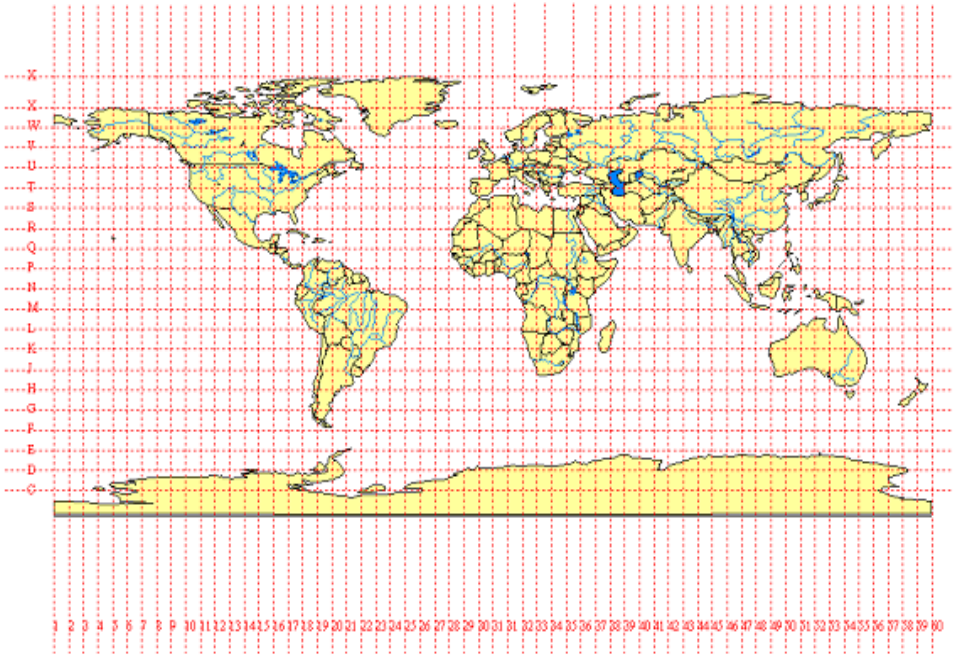

Figure II.21 : La projection UTM.

II.3.2. A-GPS (Assisted GPS) :

L'Assisted GPS (A-GPS) a été conçu par Giovanni Vannucci, chercheur au centre de recherche de Lucent Technologies, Bell Labs. Au milieu des années 90, il entreprend de localiser les téléphones portables à l'aide de liaisons satellites afin de pouvoir secourir les gens en cas d'urgence. De sa collaboration avec Bob Richton, du Wireless Technology Laboratory, naît la méthode A-GPS qui réduit le temps nécessaire pour établir la première connexion satellite (TTFF pour Time To First Fix). Le délai passe de plusieurs minutes avec un système GPS traditionnel à quelques secondes avec l'A-GPS.

Contrairement au GPS, qui nécessite un récepteur et une antenne, l'A-GPS travaille en relation avec un serveur A-GPS hébergé chez l'opérateur. Le terminal mobile, muni d'un récepteur GPS miniaturisé, envoie une requête au serveur par le réseau IP.

Celui-ci, qui connaît en temps réel le positionnement des satellites, sert d'aiguilleur et indique au terminal les signaux GPS à suivre. Avec cette méthode, le récepteur A-GPS du terminal mobile peut, à la différence des récepteurs GPS traditionnels, détecter des signaux de très faible amplitude.

Si le mobile parvient à rester connecté au réseau de l'opérateur dans des endroits souterrains (parking...), il peut encore théoriquement fournir des données sur sa position. En revanche,

dès que le mobile ne reçoit plus le réseau de l'opérateur et perd le signal GPS, les données ne sont plus rafraîchies. Le niveau de précision est élevé puisqu'il est annoncé entre 5 et 10 mètres contre 3 à 50 mètres avec le GPS traditionnel. De plus, le terminal mobile, à l'image du dernier iPAQ hw6515 de HP, n'a pas besoin d'antenne et de récepteur externe.

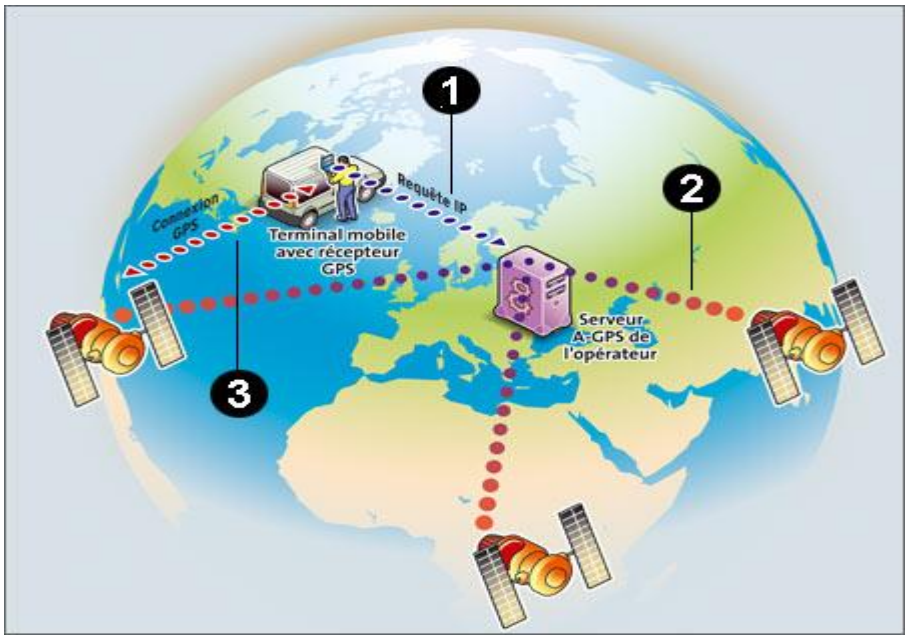

Figure II.22 : A-GPS.

1. Pour établir la liaison avec le satellite, le terminal mobile envoie une requête au serveur A-GPS hébergé chez l'opérateur.
2. Le serveur, qui suit en temps réel le positionnement des satellites, indique au terminal le signal GPS qu'il faut suivre.
3. Le terminal mobile fonctionne alors de manière autonome. En cas de perte de signal, il relance une requête auprès du serveur.

II.3.3. E-OTD (Enhanced Observed Time Difference) :

Une station doit recevoir un signal synchrone de la part du MS ; la différence de temps de transmission entre le MS et deux BTS[2] décrit une hyperbole. Avec trois stations on peut estimer la position du MS grâce à l'intersection des hyperboles.

L'exactitude de la position est de 125m, mais à la différence du GPS cette méthode ne dépend pas de la clarté du ciel.

[2] BTS équipement radio émetteur/récepteur qui communique avec le MS.

II.3.4. TOA (Time Of Arrival) :

TOA calcules le temps de transmission entre la station mobile et le BTS et vise versa. Considérant le fait que le temps de propagation d'une onde radio est connu, il est alors possible d'estimer la distance qui sépare la station mobile du BTS.

Cette méthode permet de localiser l'utilisateur dans un cercle qui à pour rayon la distance qui sépare le BTS de la MS et qui a pour centre le BTS.

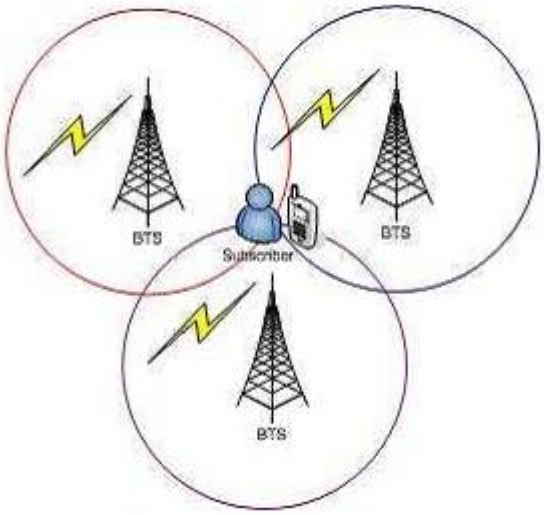

Figure II.23 : TOA.

II.3.5. Cell-ID :

Cette méthode est basée sur le réseau GSM et plus particulièrement sur l'indentification de la cellule. Elle utilise l'identifiant de la BTS dans lequel la station mobile est actuellement enregistrée. La topologie du réseau GSM est ensuite utilisé afin d'estimer la position du mobile. La précision de cette méthode dépend principalement de la taille de la cellule ainsi que sur l'environnement avoisinant (c'est-à-dire rural ou urbain).

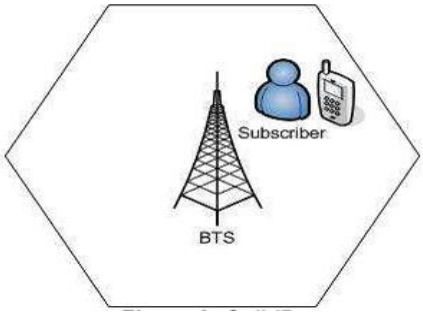

Figure II.24 : Cell-ID.

II.4. Autres GNSS :

(Système de navigation globale par satellite) Il existe bien d'autres systèmes de navigation tels que :

II.4.1. Galileo :

Est le système lancé par l'Europe dans le but de ne plus être dépendant du système américain. Il est opérationnel depuis 2013. Le système Galileo aura quasiment la même structure que le NAVSTAR GPS et les récepteurs pourront fonctionner avec NASTAR GPS et Galileo. Ce système sera compatible avec EGNOS qui est un réseau de stations au sol en Europe, ce qui permettra d'être très précis.

II.4.2. GLONASS :

Est le système russe, fonctionnel depuis des années. Il est en partie compatible avec le système américain. Il est compose d'une flotte de 24 satellites.

II.4.3. Beidou :

Est le système chinois. Actuellement, il n'est compose que de cinq satellites géostationnaires au-dessus de la Chine, couvrant ainsi l'intégralité de son territoire. Une deuxième phase de Beidou va être d'envoyer 30 satellites non géostationnaires afin de couvrir en entier la Terre.

L'unique raison d'être de ce système est de s'affranchir de la dépendance des Etats-Unis. Le Japon et l'Inde sont en train de développer des systèmes locaux, composes de satellites géostationnaires.

Tous les principes de fonctionnements, calculs et problèmes physiques propres au GPS le sont également pour tous les GNSS.

II.5. Conclusion :

Dans ce chapitre, nous avons parlés un peu de la géolocalisation précisément le GPS, nous avons présenté les techniques utilisées tels que la géolocalisation par satellite, par wifi, et par GSM, nous avons présenté les technologies de positionnements, et d'autres systèmes de navigation.

Cependant de nouvelles technologies concurrentes ont vu le jour avec la géolocalisation par satellite, par wifi, et par GSM. Le positionnement par les systèmes sans fil est une aubaine pour les opérateurs de communication mobile qui sont toujours à la recherche de nouveaux services à offrir à leur clientèle.

Nous allons essayer dans la prochaine étape de notre travail (**GPS Android**), à savoir l'analyse et conception, de mettre en place un système d'information s'adaptant au mieux à l'atteinte des objectifs.

Chapitre III
Analyse et Conception

III.1. Introduction :

Apres avoir élaboré les différents concepts nécessaires à l'accomplissement de notre application, nous passons à la phase d'analyse et conception

La conception de toute solution informatique est d'une grande importance, elle doit être traitée avec rigueur et précision, car elle constitue la base du système à développer. Avant de s'engager dans la conception, il est impératif de passer par la phase d'analyse qui permet d'identifier les différents acteurs qui interagissent avec le système ainsi que leurs besoins. Puis on passe à la conception, qui en s'appuyant sur les résultats de la phase d'analyse, donnera la description détaillée du système ciblé et des objectifs à atteindre.

Pour ce faire nous avons adopté la conception avec l'UML (Unified Modelling language), qui permet de bien représenter la dynamique de notre GPS Android par la série des diagrammes qu'il offre.

III.2. Les Méthodes de conception :

Les méthodes utilisées dans les années 80 pour organiser la programmation fonctionnelle (notamment MERISE) étaient fondées sur une modélisation séparée des données et des traitements.

Lorsque la programmation orientée objet prend de l'importance au début des années 90, la nécessité d'une méthode qui lui soit adaptée devient évidente. Plus de cinquante méthodes apparaissent entre 1990 et 1995, mais uniquement trois d'entre elles se sont détachées nettement après quelque années : OMT, BOOCH et OOSE.

Chaque méthode avait ses avantages et ses partisans. Le nombre de méthode en compétition s'était réduit mais le risque d'un éclatement subsistait : la profession pouvait se diviser entre ses trois méthodes, créant autant de continents intellectuel qui auraient eu du mal à communiquer. C'est pour cela que l'unification de ses trois méthodes c'est avérée indispensable.

Pour la modélisation de notre application, nous allons utiliser le langage de modélisation unifié UML.

III.3. Présentation de l'UML : [8] & [9]

III.3.1. Définition :

UML (Unified Modeling Language traduit en « langage de modélisation objet unifié ») se définit comme un langage de modélisation graphique et textuel destiné à comprendre , et décrire des besoins, spécifier et documenter des systèmes, esquisser des architecture logicielles , concevoir des solutions et communiquer des point de vue , UML unifier a la fois les notation et les concepts orientés objet.il ne s'agit pas d'une simple notation graphique , car les concepts transmis par un diagramme ont une sémantique précise et son porteur de sens au même titre que les mots d'un langage . UML unifier également les notations nécessaire aux déférentes activités d'un processus de développement et offre, par ce biais, le moyen d'établir le suivi des diésions prises, depuis l'expression de besoin jusqu'au codage.

III.3.2. Modélisation avec L'UML :

UML propose 9 diagrammes de modélisation, réparties sur trois axes du niveau conceptuel :

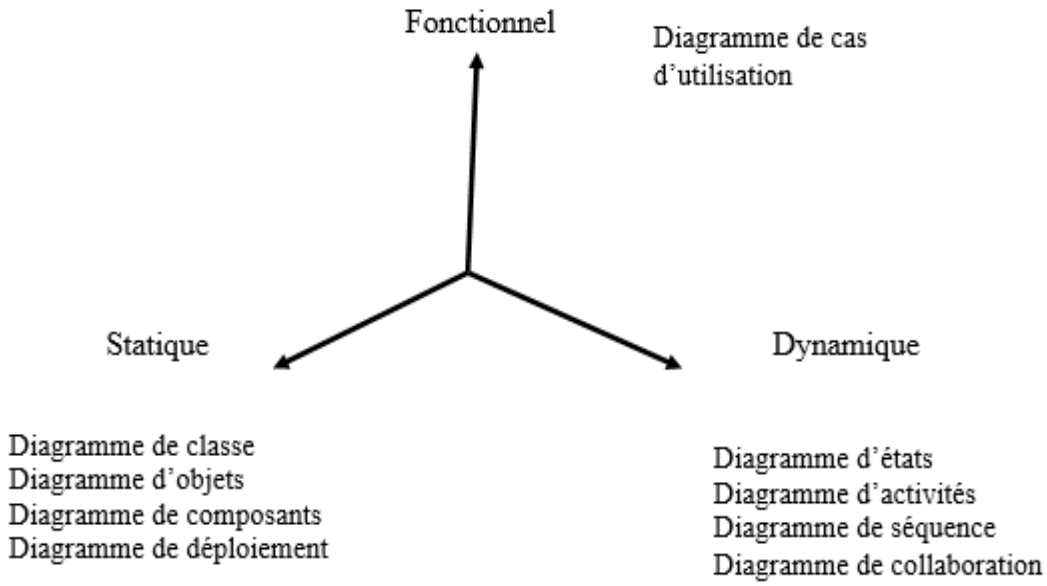

Figure 3.1. Les diagrammes de l'UML

La figure suivante donne la représentation graphique de la démarche à suivre pour la modélisation de notre application :

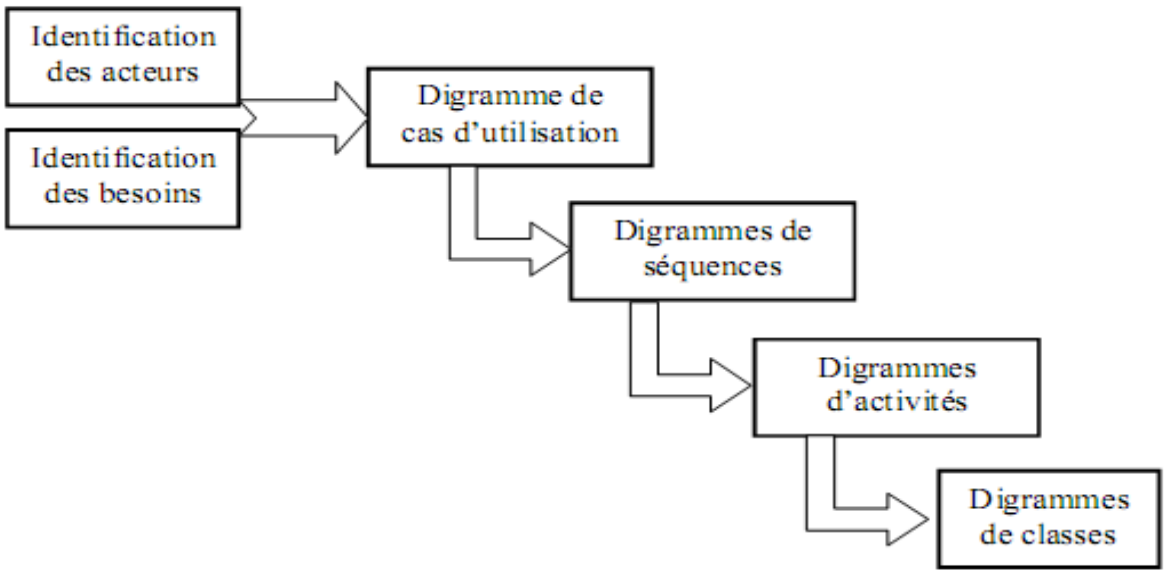

Figure 3.2. Représentation graphique de la démarche de modélisation

III.4. Analyse

III.4.1. Identification des acteurs

Notre système communique avec des acteurs afin de réaliser un ensemble d'opérations. Un acteur représente un rôle que peut jouer l'utilisateur dans le système. L'acteur est associé à un cas d'utilisation, c'est-à-dire qu'il peut interagir avec lui et participer à son scénario.

Les acteurs de notre système sont toute personne qui utilise une machine (Téléphone, Tablette, etc..) qui marche sous le système Android 2.1 ou une version ultérieure. On les regroupe sous le nom utilisateur « **User** ».

III.4.2. Identification des besoins

L'objectif principal de notre application est de permettre à l'utilisateur de se localiser dans n'importe quel endroit dans le monde, il se fait juste que son appareil est muni de la fonctionnalité GPS et qu'il a accès à internet.

Les fonctionnalités de notre application sont :

- Démarrer l'enregistrement d'un parcours.
- Choisir un type d'activité.
- Calculer la distance parcourue.
- Exécuter l'application en arrière plant.
- Récupérer l'application à travers une notification.
- Arrêter l'enregistrement.
- Enregistrer un parcourt.
- Afficher la liste détaillée des parcours effectués.
- Modifier la source de localisation.
- Activer le contrôleur de vitesse.
- Déclencher une alarme si on a dépassé la vitesse indiqué par l'utilisateur.
- Désactiver le contrôleur de vitesse.
- Dessiner l'itinéraire entre un point de départ et un point d'arrivé.

III.4.3. <u>Spécification des tâches</u> :

Définition : Une tâche est l'ensemble des différentes fonctions qui peut être accédés pas un acteur bien spécifie.

L'acteur définis précédemment (**user**) effectue un certain nombre de taches, ces taches sont résumées dans le tableau ci-dessous :

Acteurs	Taches
User	T01: Parcourue les cartes de Google maps. T02 : Ajouter l'enregistrement d'un parcours. T03 : Enregistrer un parcours. T04 : Consulter la liste des parcours. T05 : Supprimer un parcours. T06 : Contrôler sa vitesse. T07 : Demander un itinéraire. T08 : Paramétrer l'application.

Tableau 3.1 : Spécification des tâches.

III.4.4. Spécification des scénarios :

Définition : Un scénario est une instance (occurrence) d'un cas d'utilisation, à chaque fois qu'une instance d'un acteur déclenche un cas d'utilisation, un scenario est créé. Ce scenario suivra un chemin particulier dans le cas d'utilisation.

Remarque : Durant notre étude les scenarios seront symboliser par **Si (i** : représente le numéro du scenario).
Appareil : définit un Téléphone, Tablet ou autre appareil Android.

Le tableau ci-dessous récapitule les déférents scenarios crées par les cas d'utilisation cités.

Acteurs	**Taches**	**Scénarios**
« « User	T01 : Parcourue les cartes de Google maps.	S01 : Cliquer sur l'icône de l'application. S02 : Cliquer sur l'option de l'application. « GPS traqueur » S03 : Visualiser les cartes (zoomer, se localiser…)
	T02 : Ajouter l'enregistrement d'un parcours	S01, S02 S04:Cliquer sur le bouton« Démarrer l'enregistrement » S05: Choisir un type d'activité dans l'interface « Choisissez un type d'activité ». S06:Cliquer sur le bouton « Arrêter l'enregistrement » de l'application
	T03 : Enregistrer un parcours	S01, S02, S04, S05 S07 : Saisir un nom dans le champ « Nom du parcours» de l'interface « Arrêt de l'enregistrement ? » Cliquer sur le bouton « Ok »
	T04 : Consulter la liste des parcours	S01, S02 S08 : Cliquer sur le bouton «Liste des parcours » de l'application
	T05 : supprimer un parcours	S01, S02 S08 : Cliquer sur le bouton «Liste des parcours » de l'application S09 : Cliquer sur un parcours S10 : Cliquer sur « oui » de l'interface « supprimer parcour »
	T05 : Contrôler sa vitesse	S01 S11 : Cliquer sur l'option «Vitesse contrôleur » S12 : Saisir une vitesse à ne pas dépassé S13 : Clique sur le bouton switch pour l'activer

	T06 : Demander un itinéraire	S01 S14 : Cliquer sur l'option de l'application. « Itinéraire traceur » S15 : saisir un point de départ et d'arriver S16 : Cliquer sur le bouton « Rechercher »
	T07 : Paramétrer l'application.	S01 S17 : Cliquer sur le bouton « menu» du l'appareil. S18 : Modifier la source de localisation S19 : Choisir le Système d'unités S20 : Consulter la page « A propos» de l'application

Tableau 3.2. Spécification des scénarios

III.4.5. Les cas d'utilisation :

Définition : Un cas d'utilisation (**user case**) représente un ensemble de séquence d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier. Un cas d'utilisation modélise un service rendu par le système.

III.4.5.1. Description des cas d'utilisations :

Nous procéderons aux descriptions de quelques cas d'utilisations de notre système :

Cas d'utilisation : Ajouter l'enregistrement d'un parcours
Scenarios: S01, S02, S04, S05, S06
Acteur: User
Description:

1. L'utilisateur lance l'application après avoir cliqué sur son icône.
2. Le système affiche l'écran d'accueil.
3. L'utilisateur clique sur l'option « GPS traqueur »
4. L'utilisateur clique sur le bouton « Démarrer l'enregistrement ».
5. Le système affiche l'interface « Choisissez un type d'activité ».
6. L'utilisateur choisit un type d'activité parmi la liste d'activités proposées.
7. Le système enregistre les choix fait par l'utilisateur et génère une notification.
8. L'utilisateur sur le bouton « Arrêter l'enregistrement » de l'application

Cas d'utilisation : « Ajouter l'enregistrement d'un parcours ».

Cas d'utilisation : Enregistrer un parcours
Scenarios: S01, S02, S04, S05, S07.
Acteur: User
Description:

1. L'utilisateur lance l'application après avoir cliqué sur son icône.
2. Le système affiche l'écran d'accueil.
3. L'utilisateur clique sur l'option « GPS traqueur ».
4. L'utilisateur clique sur le bouton « Démarrer l'enregistrement ».
5. Le système affiche l'interface « Choisissez un type d'activité ».
6. L'utilisateur choisit un type d'activité parmi la liste d'activités proposées.
7. Le système enregistre les choix fait par l'utilisateur.
8. L'utilisateur clique sur le bouton « Arrêter l'enregistrement ».
9. Le système construit et affiche l'interface « Arrêt de l'enregistrement ? ».
10. L'utilisateur saisie un non dans le champ « Nom du parcours ».
11. L'utilisateur clique sur le bouton « Ok » de l'interface « Arrêt de l'enregistrement ? ».

Cas d'utilisation : Enregistrer un parcours.

Cas d'utilisation : Consulter les informations d'un parcours

Scenarios: S01, S02, S08
Acteur: User
Description:

1. L'utilisateur lance l'application après avoir cliqué sur son icône.
2. Le système affiche l'écran d'accueil.
3. L'utilisateur clique sur l'option « GPS traqueur ».
4. L'utilisateur clique sur le bouton « Liste des parcours ».
5. Le système construit et affiche la liste des parcours et les informations du parcours.

Cas d'utilisation : « Consulter la liste des parcours ».

Cas d'utilisation : Contrôler sa vitesse

Scenarios: S01, S11, S12, S13.
Acteur: User
Description:

1. L'utilisateur lance l'application après avoir cliqué sur son icône.
2. Le système affiche l'écran d'accueil.
3. L'utilisateur cliquer sur l'option «Vitesse contrôleur »
4. L'utilisateur saisir une vitesse à ne pas dépassé
5. L'utilisateur clique sur le bouton switch pour l'activer
6. Le système calcule la vitesse de déplacement de l'utilisateur et la compare à la vitesse saisir par l'utilisateur, et si la vitesse de déplacement dépasse la vitesse saisir le système déclenche une alarme.

Cas d'utilisation : Contrôler sa vitesse.

Cas d'utilisation : Demander un itinéraire.

Scenarios: S01, S14, S15, S16.
Acteur: User
Description:

1. L'utilisateur lance l'application après avoir cliqué sur son icône.
2. Le système affiche l'écran d'accueil.
3. L'utilisateur clique sur l'option de l'application. « Itinéraire traceur »
4. L'utilisateur saisir un point de départ et d'arriver
5. L'utilisateur clique sur le bouton « Rechercher »
6. Le système calcule l'itinéraire et dessine une ligne entre les deux points saisir par l'utilisateur sur une carte Google map.

Cas d'utilisation : Demander un itinéraire.

Cas d'utilisation : paramétrer l'application (Modifier la source de localisations)
Scénarios: S01, S17, S18
Acteur: User

1. L'utilisateur lance l'application après avoir cliqué sur l'icône.
2. Le système affiche l'écran d'accueil.
3. L'utilisateur clique sur le bouton « menu » de l'appareil.
4. Le système construit et affiche l'interface des fonctions cachées de l'application
5. L'utilisateur clique sur le bouton « Sources de localisations ».
6. Le système affiche l'interface « Paramètres de sécurité et de localisations ».
7. L'utilisateur choisie la source de localisation.

Cas d'utilisation « paramétrer l'application (Modifier la source de localisations) ».

III.5 Conception

Le processus de conception de notre projet se caractérise par deux niveaux : le niveau applicatif et le niveau « données ».

Le niveau applicatif s'appuie essentiellement sur quelques diagrammes du langage de modélisation UML, à cet effet nous avons adopté la démarche suivante:

- Après l'identification des différents acteurs ainsi que les cas d'utilisation qui sont mis en œuvre par ces acteurs, le diagramme de cas d'utilisation est élaboré.
- Chaque cas d'utilisation se traduit par un ou plusieurs scénarios. Chaque scénario fait l'objet d'une description sous forme graphique à l'aide d'un diagramme de séquence et un digramme d'activité.
- Une identification des classes est fournie par la synthèse des diagrammes de séquence, ainsi le diagramme de classe sera élaboré.

Le niveau données concerne l'organisation conceptuelle, logique et physique des données manipulées. Durant la partie analyse nous avons pu identifier les données nécessaires et indispensables au bon fonctionnement de l'application, et à travers la conception du niveau applicatif nous allons dégager les classes significatives, dès lors on peut élaborer la conception de la base de données.

III.5.1. Le niveau applicatif :

III.5.1.1. Le diagramme des cas d'utilisation:

Les diagrammes de cas d'utilisation permettent de représenter un ensemble de cas d'utilisation, d'acteurs et leurs relations. Ils présentent la vue statique des cas d'utilisation d'un système et sont particulièrement importants dans l'organisation et la modélisation des comportements d'un système.

La relation d'inclusion (include) : Elle indique que le cas d'utilisation source contient aussi le comportement décrit dans le cas d'utilisation destination. Cette relation permet de décomposer des comportements et de définir les comportements partageables entre plusieurs cas d'utilisations.

La relation d'extension (Extend) : Elle indique que le cas d'utilisation source ajoute son comportement au cas d'utilisation destination. L'extension peut être soumise à des conditions.

Lors de la phase d'analyse nous avons pu identifier les acteurs ainsi que les cas d'utilisation associés à ces derniers. Ce qui nous donne l'opportunité d'élaborer le diagramme des cas d'utilisation suivant :

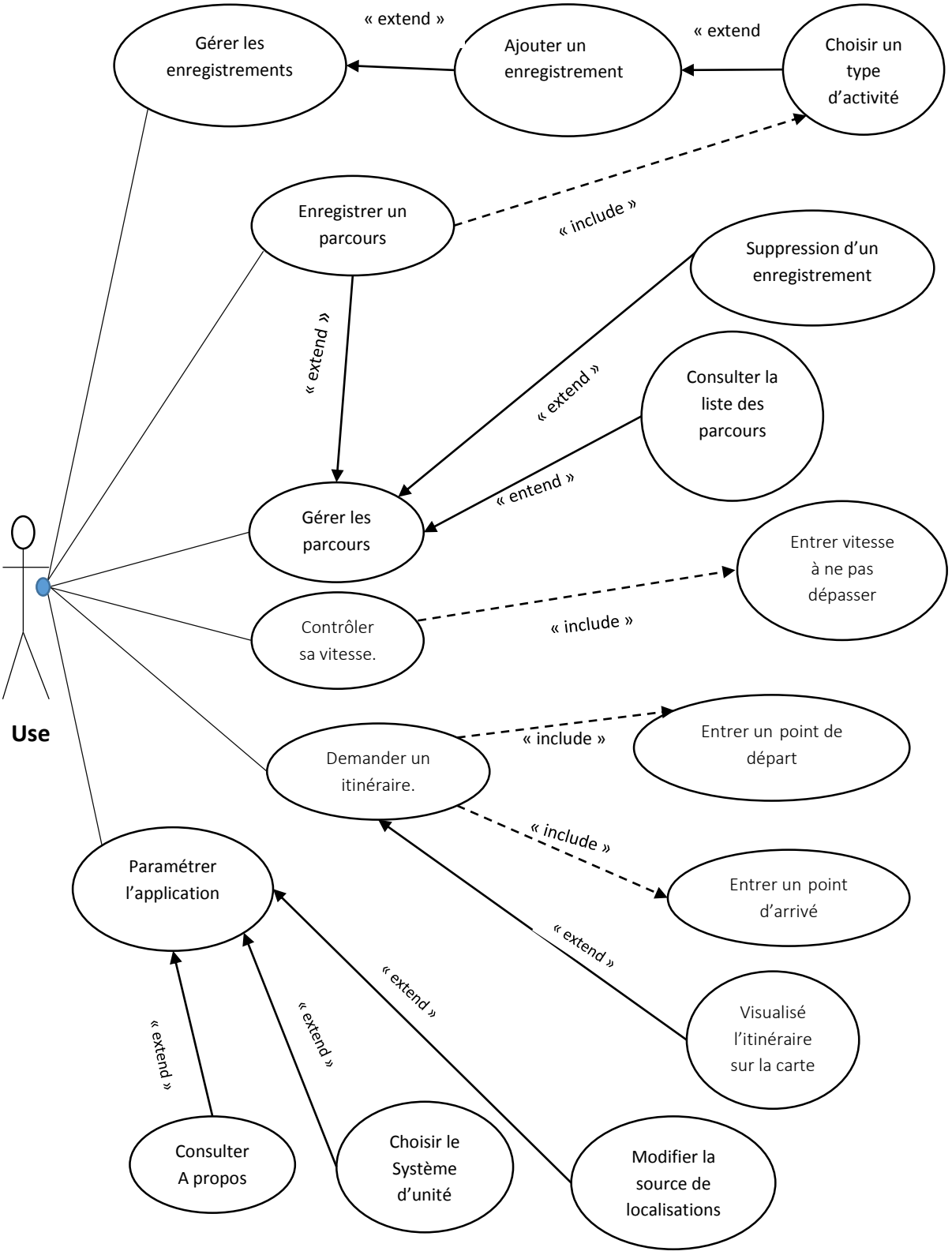

Figure 3.2. Diagramme des cas d'utilisation générale.

III.5.1.2 Les diagrammes de séquences :

Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur : saisir une donnée, consulter une donnée, lancer un traitement ; il met en évidence les objets manipulés ainsi que les opérations qui font passer d'un objet à l'autre. Dans notre cas on s'intéresse seulement à effectuer la représentation du diagramme de séquence pour les cas d'utilisation déjà présentés auparavant.

Les composants d'un diagramme de séquence sont les suivants :

Les objets : ils apparaissent dans la partie supérieure, ce qui facilite l'indentification des classes qui participent à l'interaction.

Les messages : ils sont représentés par des flèches directionnelles. Au-dessus des flèches directionnelles figurent un texte nous informant du message envoyé entre les objets.

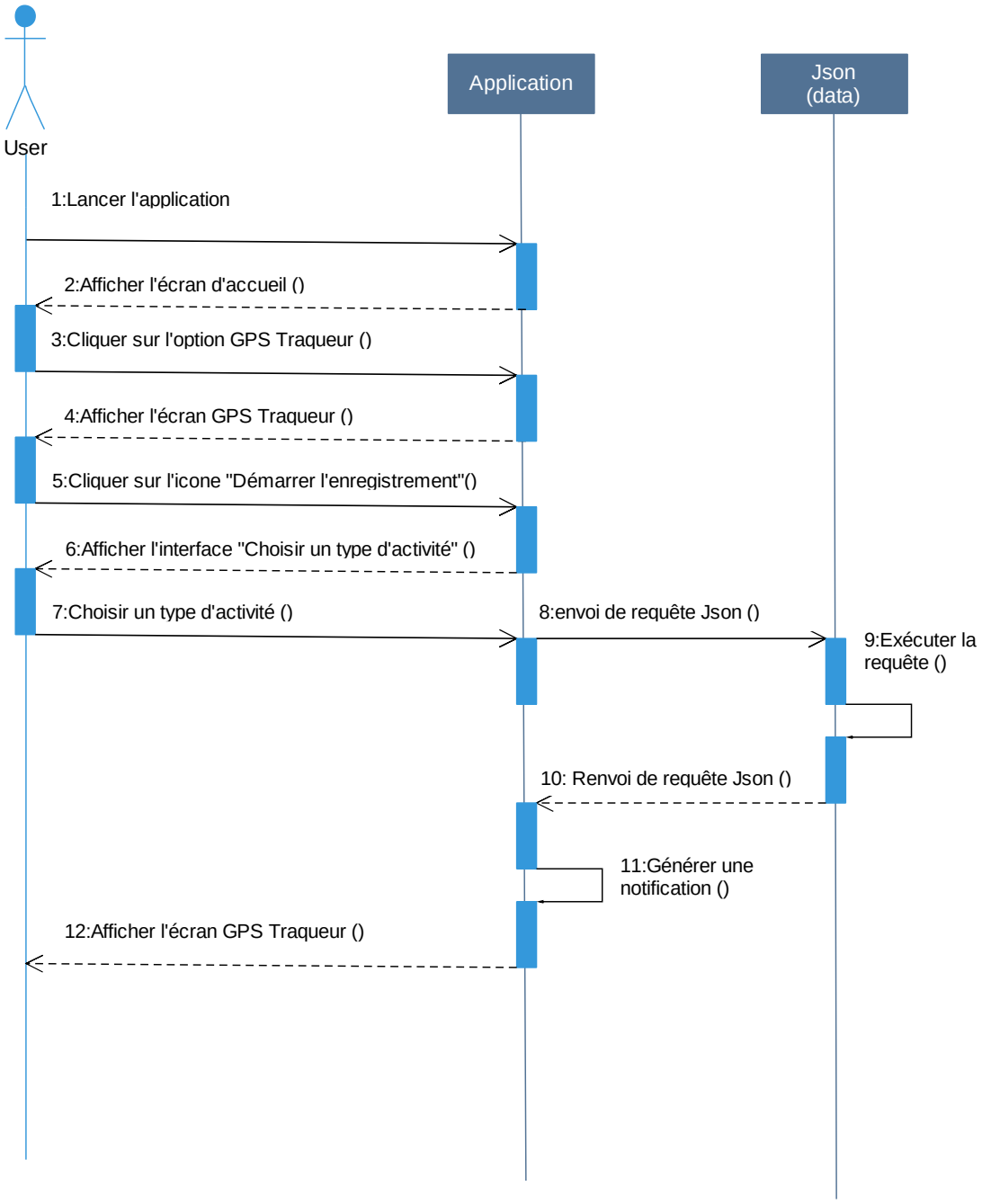

Figure 3.3 Diagramme de séquence de cas d'utilisation « Ajouter l'enregistrement d'un parcours ».

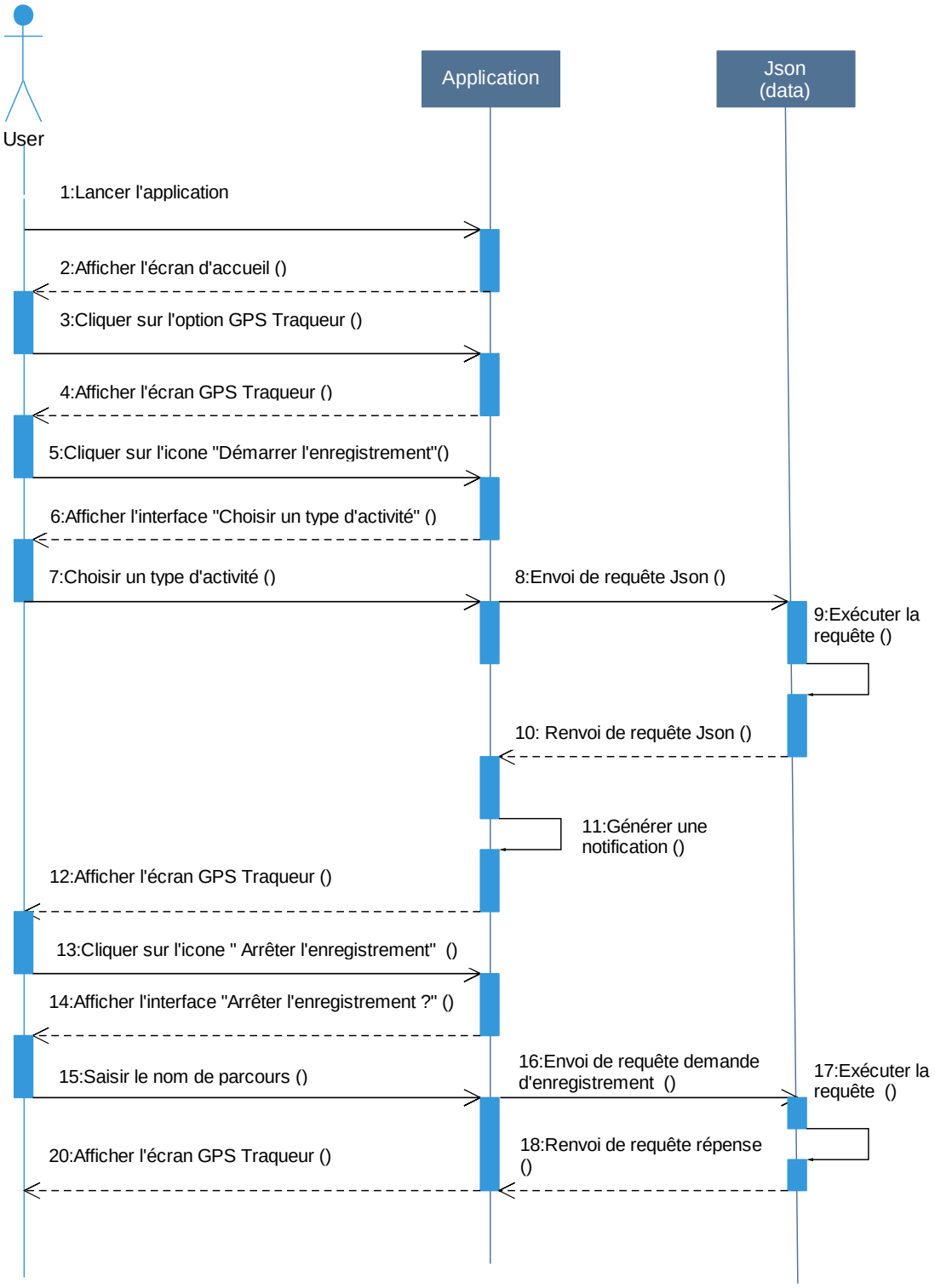

Figure 3.4 Diagramme de séquence de cas d'utilisation « Enregistrer un parcours ».

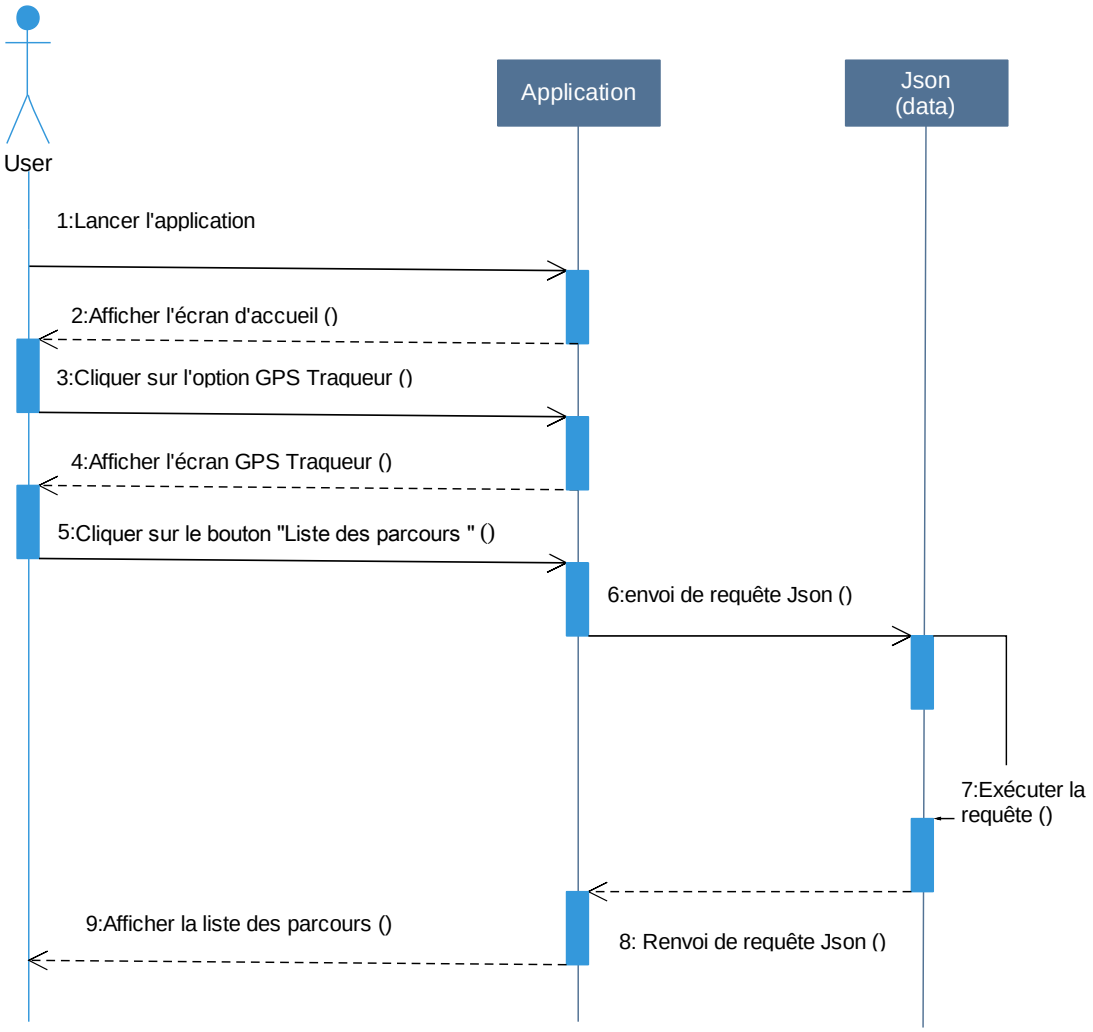

Figure 3.5 Diagramme de séquence de cas d'utilisation «Afficher la liste des parcours ».

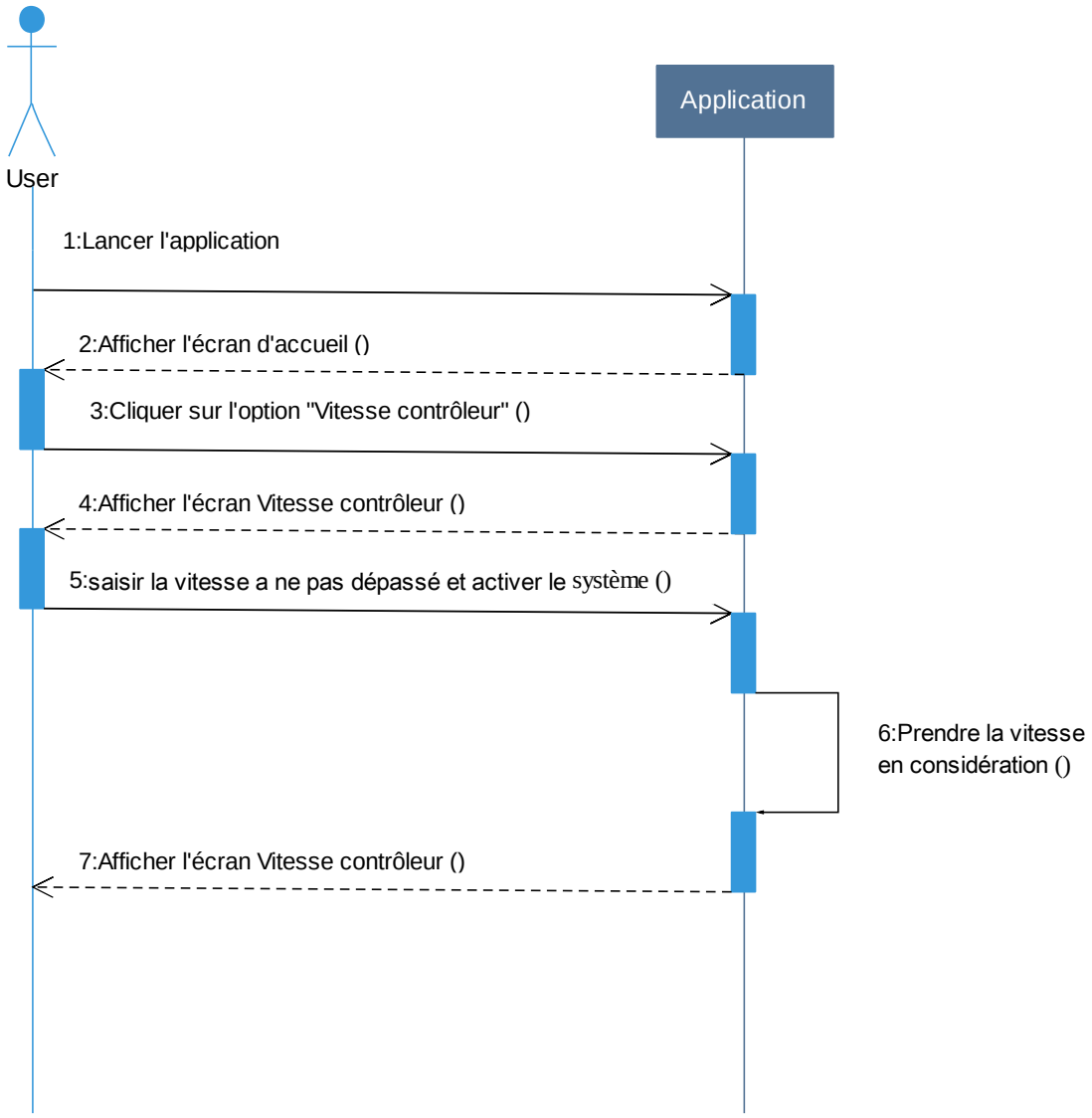

Figure3.6 Diagramme de séquence de cas d'utilisation «Contrôler sa vitesse ».

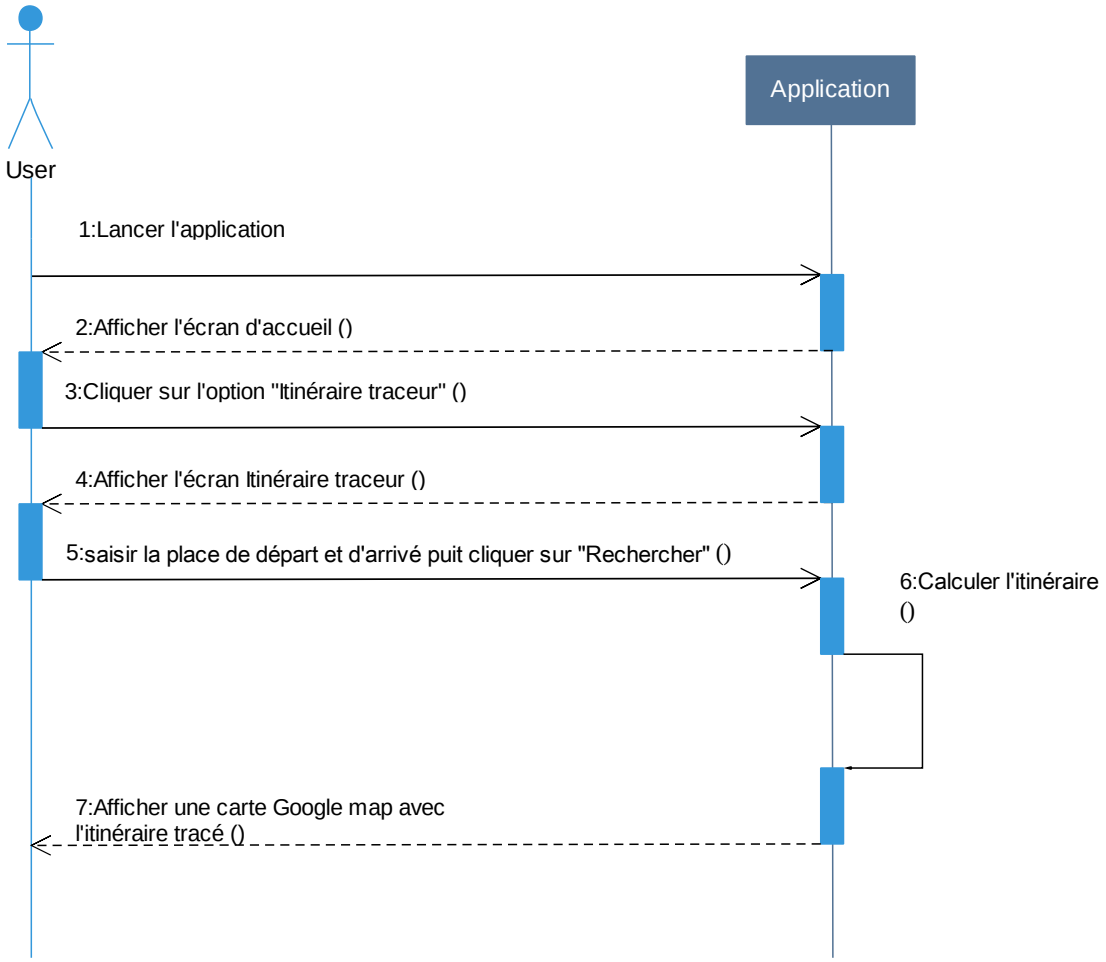

Figure3.7 Diagramme de séquence de cas d'utilisation «Demander un itinéraire».

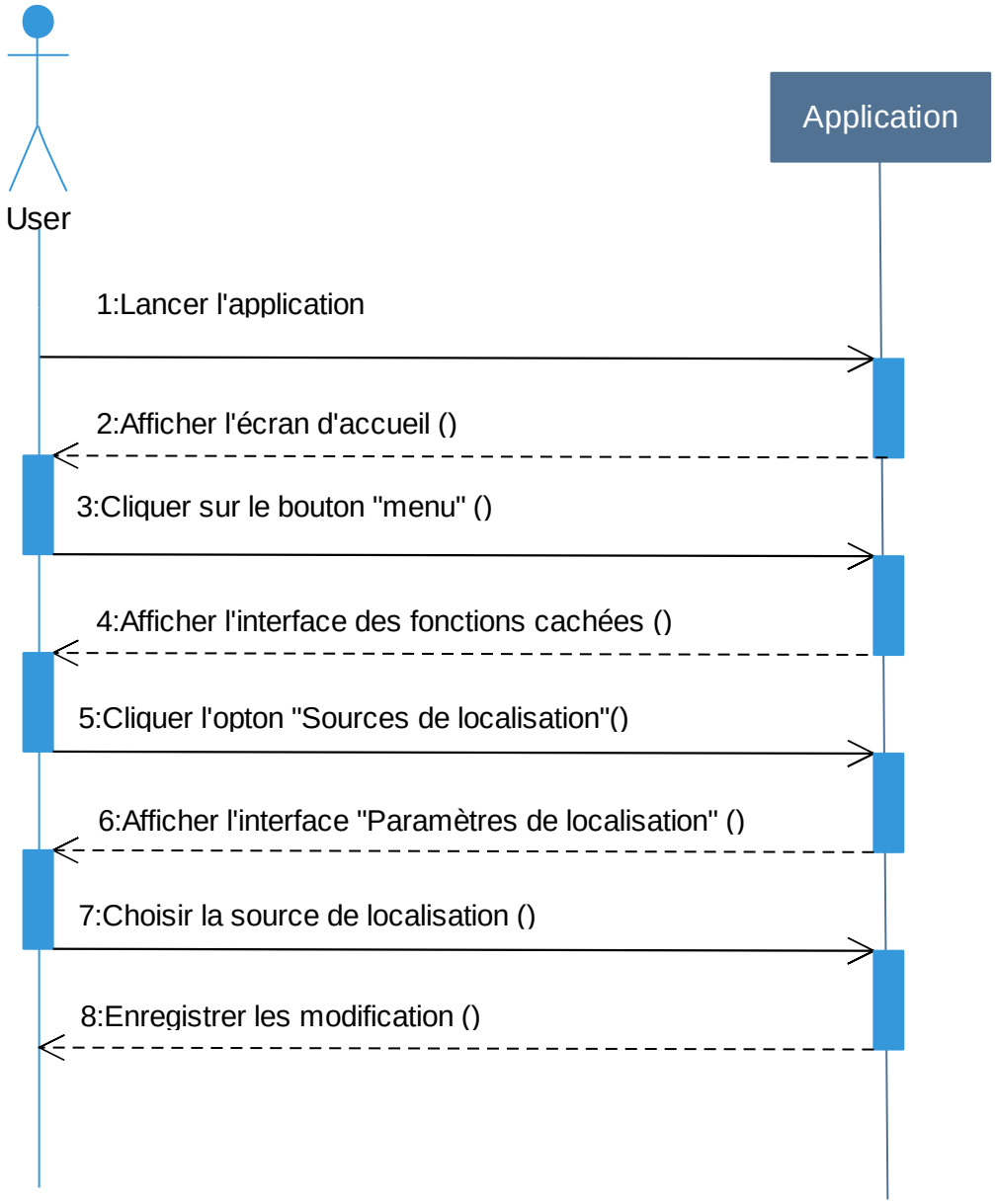

Figure3.7 Diagramme de séquence de cas d'utilisation «paramètre de l'application (modifier la source de localisation) ».

III.5.1.3 Les diagrammes d'activités:

Un diagramme d'activités : apporte un point de vue complémentaire à l'aspect dynamique de la modélisation. Il offre un pouvoir d'expression très proche des langages de programmation objets. Il est donc bien adapté à la spécification détaillée des traitements en phase de réalisation. Un diagramme d'activités se concentre plutôt sur les activités entre les objets, c'est-à-dire, il met en évidence l'activité qui a lieu dans le temps, donc les opérations transmises entre les objets.

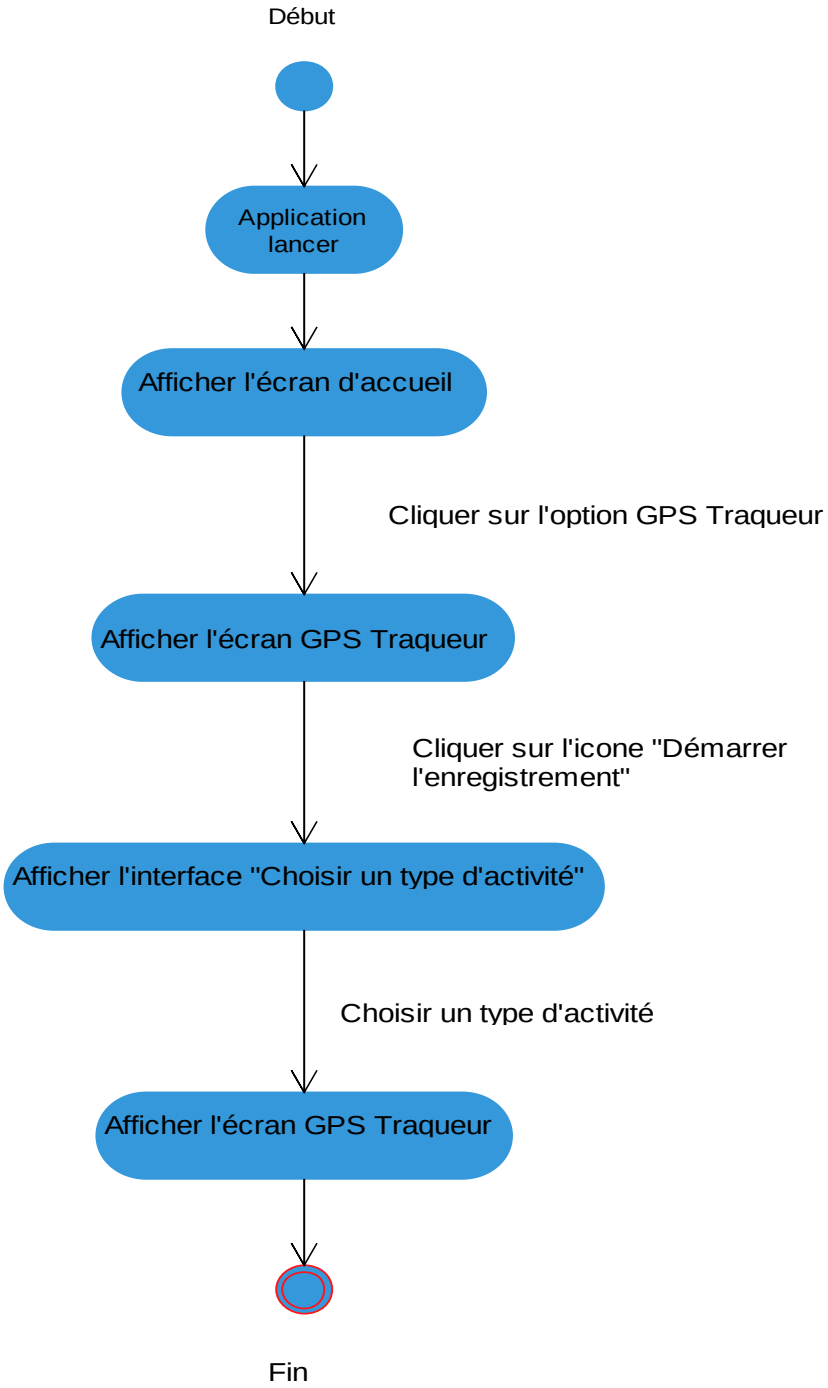

Figure 3.8 Diagramme d'activité « Ajouter l'enregistrement d'un parcours ».

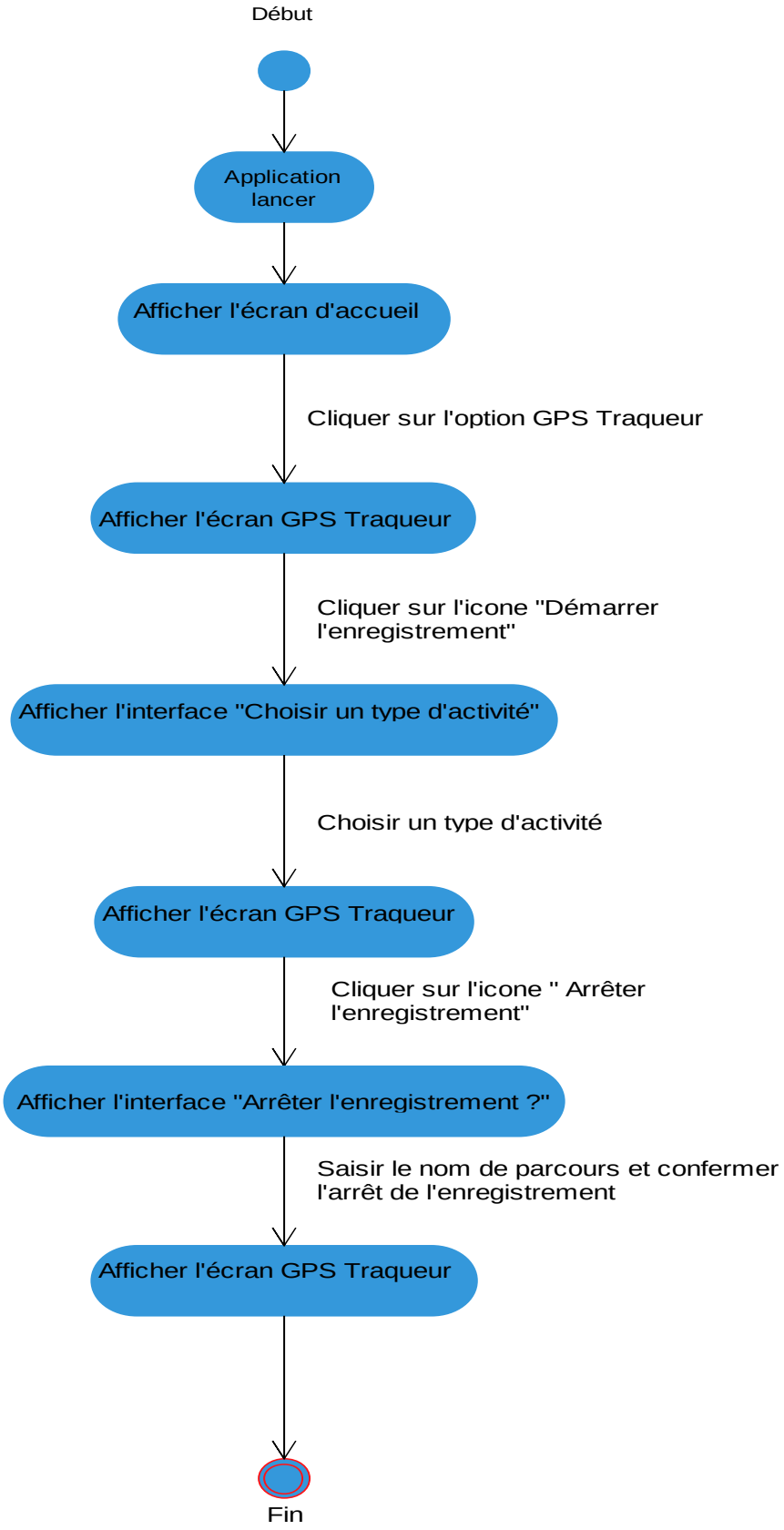

Figure 3.9 Diagramme d'activité « enregistrement d'un parcours ».

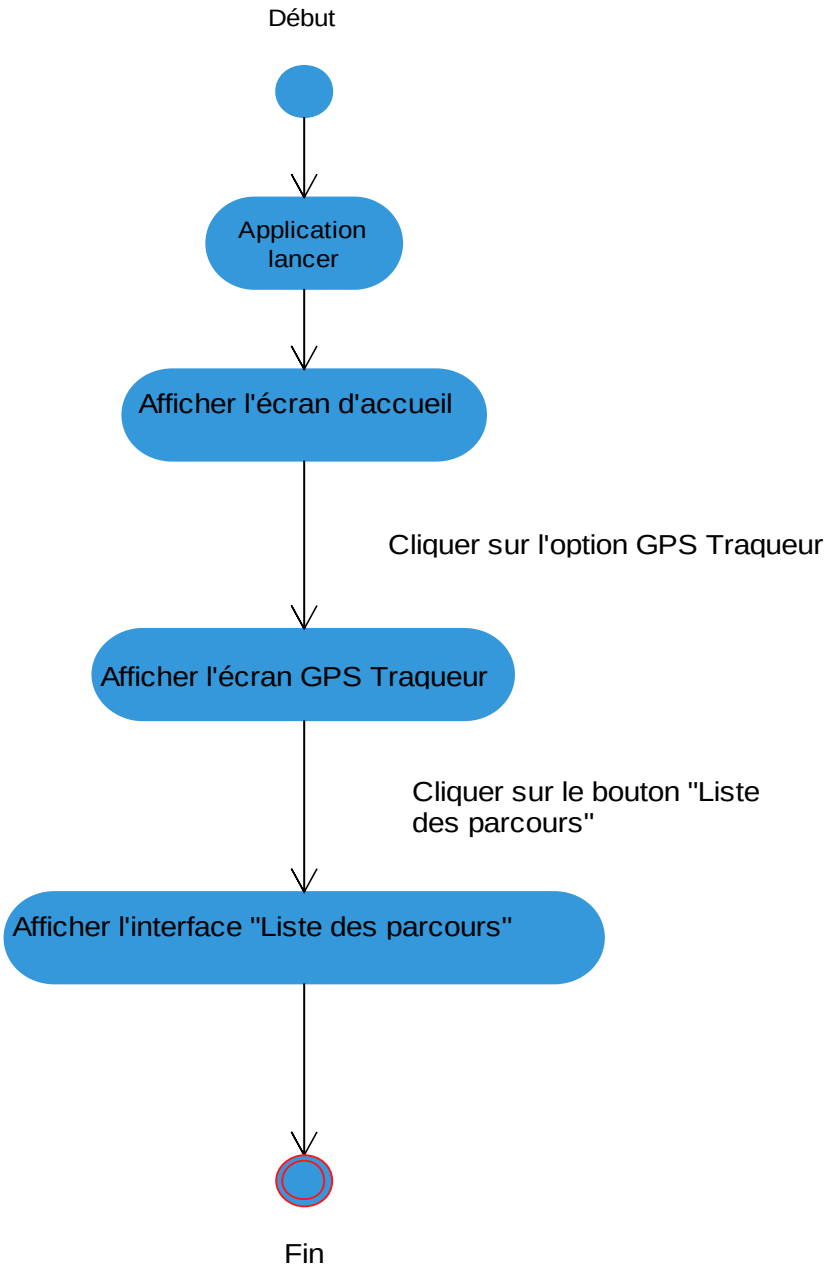

Figure 3.10 Diagramme d'activité « Afficher la liste des parcours ».

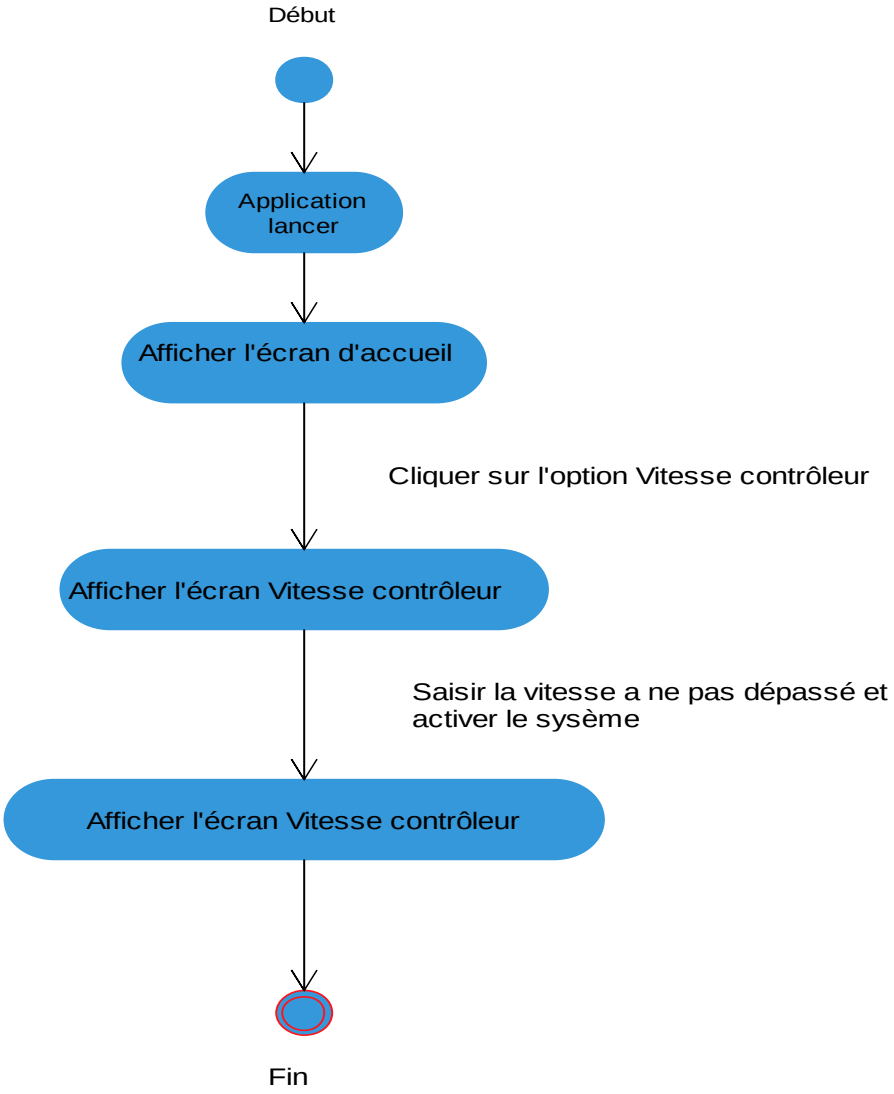

Figure 3.11 Diagramme d'activité « Contrôler sa vitesse ».

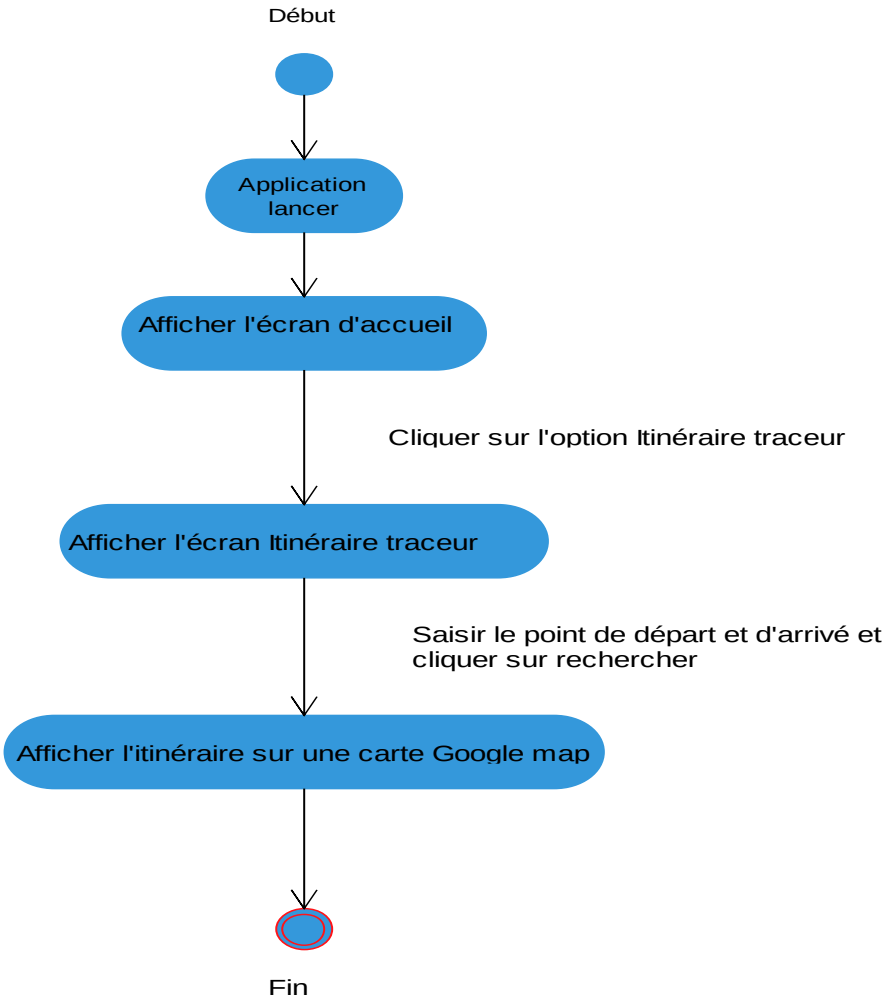

Figure 3.12 Diagramme d'activité « demander un itinéraire ».

III.5.1.4. Les diagrammes de classe :

Le diagramme des classes représente la structure statique d'un système. Il contient principalement des classes ainsi que leurs associations, mais on peut aussi y trouver des objets. L'intérêt majeur du diagramme de classe est de modéliser les entités du système. Vu le grand nombre des cas utilisation présentés précédemment, nous allons juste présenter quelques diagrammes des classes du notre système.

- Le caractère + est utilisé pour préciser la visibilité public.
- Le caractère - est utilisé pour préciser la visibilité protected.
- Le caractère # est utilisé pour préciser la visibilité private.

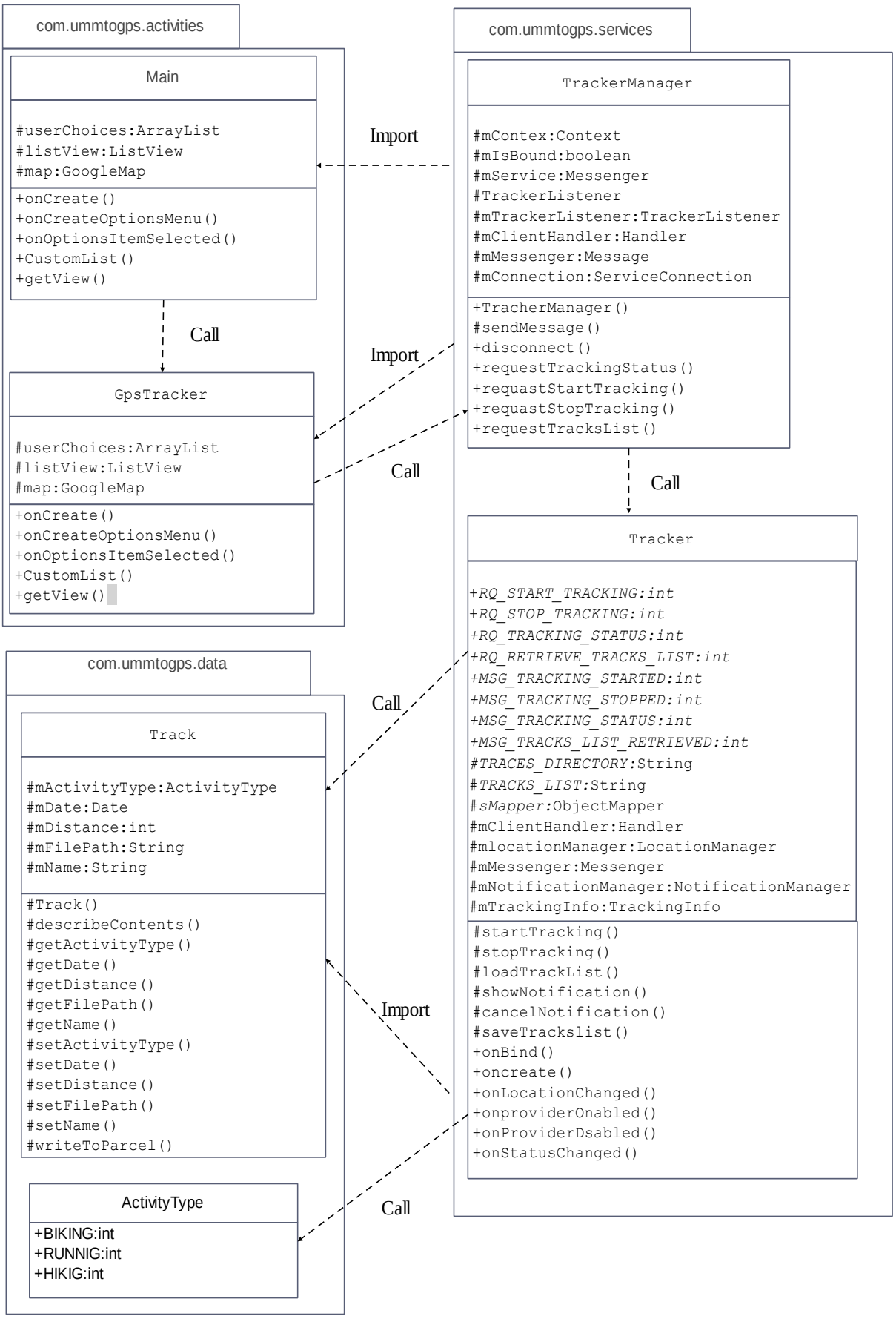

Figure 3.13 Diagramme de calasses « Ajouter l'enregistrement d'un parcours ».

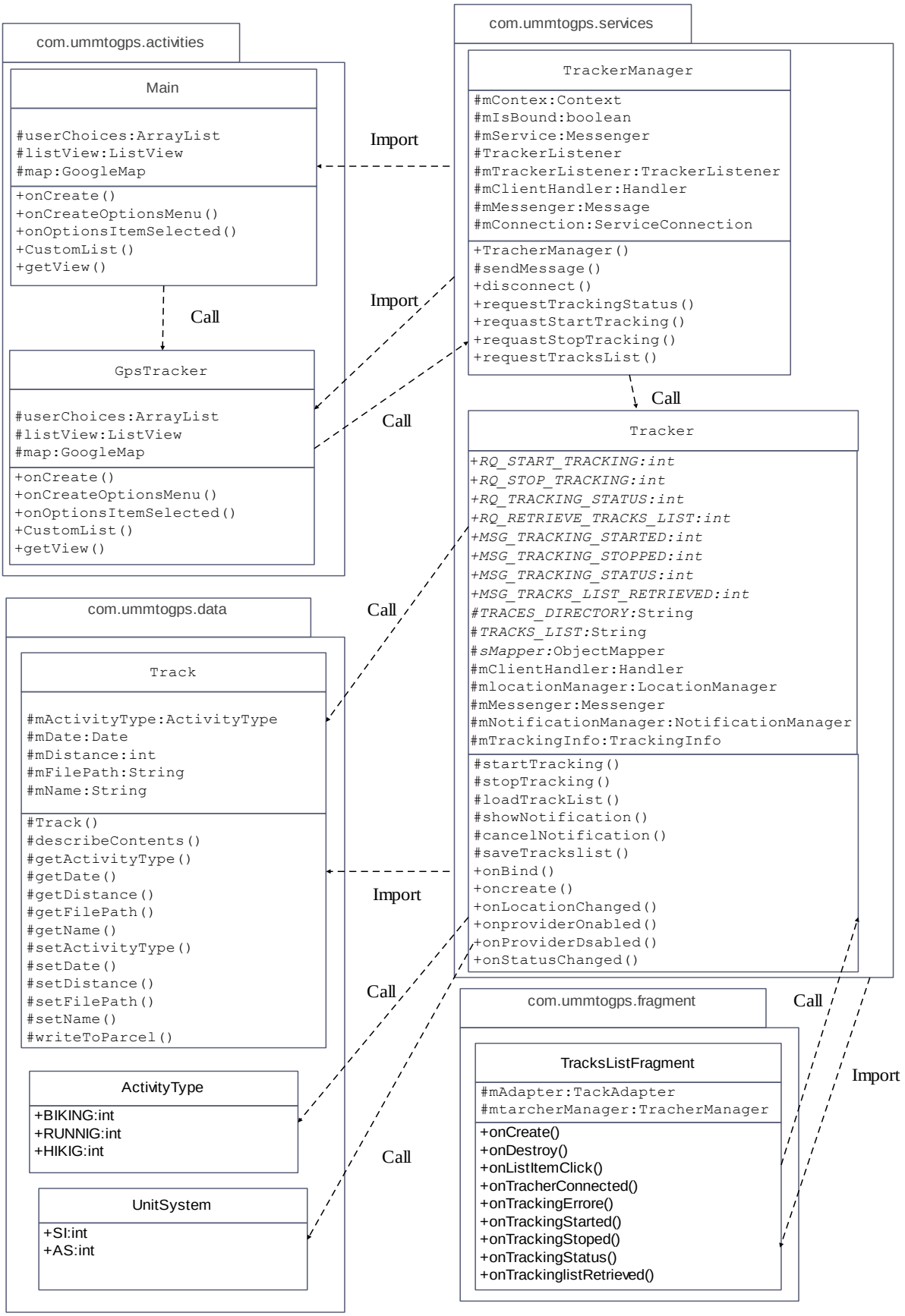

Figure 3.14 Diagramme de calasses « Afficher la liste des parcours ».

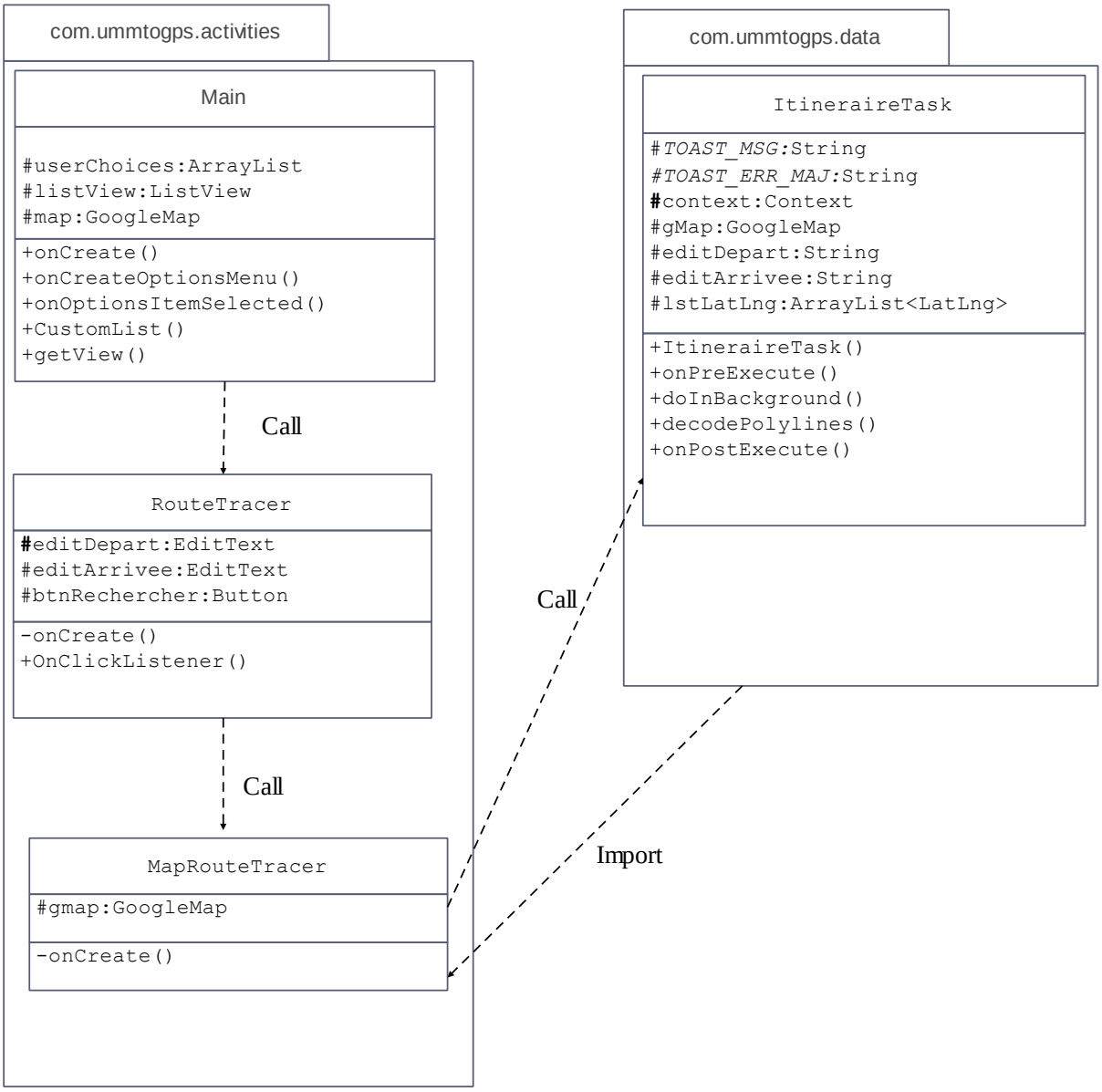

Figure 3.15 Diagramme de calasses « Demander un itinéraire ».

III.5.2 Le niveau données :

III.5.2.1 Schéma de la base de données :

Le schéma conceptuel de la base de données est représenté par le diagramme suivant :

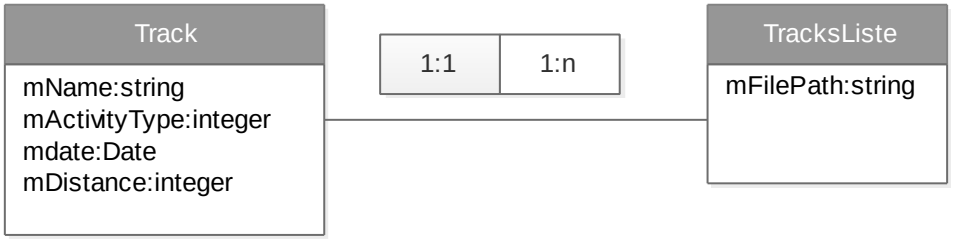

Figure 3.16 le schéma de a base de données

La base de données est séparée en deux parties car nous n'avons besoin que de deux objets à représenter à savoir les informations du parcours et le chemin d'accès à ce fichier cas chaque parcours est enregistré dans un fichier Json a part, pour la performance maximale et la simplicité de gestion de notre application.

III.5.2.2 les tables:

La table Track :

Champ	**Signification**	**Type**
mName	Nom du parcours enregistré	String
mDate	La date du l'enregistrement du parcours	Date
mDistance	La distance parcourue	integer
mFilePath	Le chemin d'accès au fichier json enregistré	String
mActivityType	Le type d'activité choisit lors de l'enregistrement du parcours	integer

La table TracksListe :

Champ	**Signification**	**Type**
mFilePath	Le chemin d'accès à chaque fichier json enregistré	String

III.6. Conclusion :

Dans ce chapitre, nous avons introduit les fonctionnalités de notre application. Pour les atteindre, nous avons proposé une solution fondée sur une analyse et une conception modélisées à l'aide du langage de modélisation unifié UML. Pour cela, nous avons défini les acteurs de notre application, les tâches qu'ils assurent et les scénarios associés à chaque tâche. Nous avons élaboré les diagrammes de cas d'utilisation de chaque acteur, élaboré les diagrammes de séquence, de classes de quelque cas d'utilisation. En fin, nous avons représenté les tables de la base de données de notre application.

Le chapitre suivant sera consacré à la réalisation de notre application, en présentant les outils de développement utilisés et les différentes fonctionnalités de notre application à travers ses différentes interfaces.

Chapitre IV
Réalisation

IV.1. Introduction :

Ce chapitre sera divisé en deux parties ; la première partie sera consacrée à la description de l'environnement du travail et les outils de développement de notre application. La deuxième partie contiendra les détails techniques d'implémentation des différentes fonctionnalités offertes par notre système.

Partie 1 : Description de l'environnement du travail et les outils de développement

IV.2. Environnement de travail :

IV.2.1. Environnement matériel :

Dans notre projet, l'environnement du travail disponible pour le développement de l'application est le suivant :

- PC Portable : Intel Core i7, 2.20 GHZ, 6 GO de mémoire vive, Windows 8.1.Pro 64bits

IV.2.2. Environnement cible :

Notre application est destinée pour tous les appareil équipé d'un système android 2.1 dite éclair, et toutefois, notre application, est utilisable sur la version 2.2 et sur les versions antérieures d'Android.

IV.3. Outils de développement :

Avant de commencer le développement d'une application, il faut nécessairement préparer l'environnement du travail. Dans notre cas, on a choisi d'utiliser la plateforme Eclipse. Pour la préparer, on a effectué les différentes tâches suivantes :

- Installation Eclipse
- Installation SDK Android
- Intégration SDK Android sous éclipse pour pouvoir créer des projets Android.

Après la préparation d'éclipse et pour pouvoir développer une application dans de bonnes conditions, il faut bien savoir choisir son environnement de développement selon les besoins, de ce fait, on a développé notre application en JAVA, qui va être compilée en un package DEX par le Dalvik VM ensuite déployée sous une extension APK pour être installée sur un appareil fonctionnant avec un système Android, pour cela on a choisi L'IDE Eclipse qui nous permettra de réaliser ce travail.

IV.3.1. Langage de programmation (Java) :

Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton de Sun Microsystems. Mais c'est également un environnement d'exécution. Java peut être séparé en deux parties. D'une part, le programme écrit en langage Java et d'autre autre part, une machine virtuelle (JVM) qui va se charger de l'exécution du programme Java. C'est cette plateforme qui garantit la portabilité de Java. Il suffit qu'un système ait une machine virtuelle Java pour que tout programme écrit en ce langage puisse fonctionner.

IV.3.2. IDE Eclipse :

Eclipse est un IDE qui permet de programmer dans différents langages grâce à ses nombreux plug-ins et notamment le plug-in d'Android. Une interface spécifique permet de gérer des fichiers java et de compiler ses programmes. Les fichiers sont organisés selon une arborescence qui correspond aux paquetages java définis. L'analyse syntaxique permet de mettre en valeur les mots clés dans les fichiers java. Eclipse dispose aussi d'un système d'auto complétion des fonctions, de détection des erreurs syntaxiques en temps réel sans oublier un système de débogage permettant d'exécuter ses programmes pas à pas.

IV.3.3. Le plugin ADT:

Pour développer sous Android, on a installé le plugin ADT qui rajoutera à Eclipse les fonctionnalités spécialisées dans le développement sous Android.

IV.3.4. le SDK Android:

Software Development Kit est un kit de développement basé sur le langage Java. Il s'agit des outils que Google fournit pour interagir avec Eclipse.

IV.3.5 Android :

Android : est un système d'exploitation open source pour smart phones, PDA et terminaux mobiles. Il s'agit également d'un langage de programmation basé sur le JAVA et sur le XML. Le JAVA permet d'interagir avec l'utilisateur en faisant la liaison entre l'interface graphique et la base de données. Quant au XML, il permet notamment de décrire les interfaces graphiques.

Notre Application a été développée pour la version 2.1 d'Android baptisé Eclair. Toutefois, elle est utilisable sur la version 2.2 et sur les versions antérieures d'Android.

IV.3.6 La bibliothèque Jackson : [10]

Jackson est une bibliothèque Java permettant de transformer très facilement un fichier JSON en un objet Java et ainsi, par exemple, exploiter très facilement et rapidement les résultats d'un web service. Vous allez sûrement vous demander "***Pourquoi Jackson ?***".Par ce que **Jackson** est une bibliothèque :

- de Streaming (Lecture / Ecriture) ;
- rapide ;
- puissante ;
- possédant aucune dépendance ;
- open source ;
- configurable à souhait.

Le format JSON : (JavaScript Object Notation – Notation Objet issue de JavaScript) est un format léger d'échange de données. Il est facile à lire ou à écrire. Il est basé sur un sous-ensemble du langage de programmation JavaScript. JSON est un format texte complètement indépendant de tout langage, mais les conventions qu'il utilise seront familières à tout programmeur habitué aux langages descendant du C, comme par exemple : C lui-même, C++, C#, Java, JavaScript, Perl, Python et bien d'autres. Ces propriétés font de JSON un langage d'échange de données idéal.

Partie 2 : Présentation de l'application

IV.4. Implémentation de notre application :

IV.4.1. Partie stockage :

Pour l'implémentation de cette partie nous avons choisi de sauvegarder les données sous forme de ficher json, pour gérer ce format, on a utilisé la bibliothèque JACKSON, pour ce faire :

- Nous avons copié les deux archives jar **jackson-core-asl 1.9.4.jar** et **jackson-mapper-asl-1.9.4.jar** dans le répertoire libs
- Référencer les archives jar comme bibliothèque dans notre Project

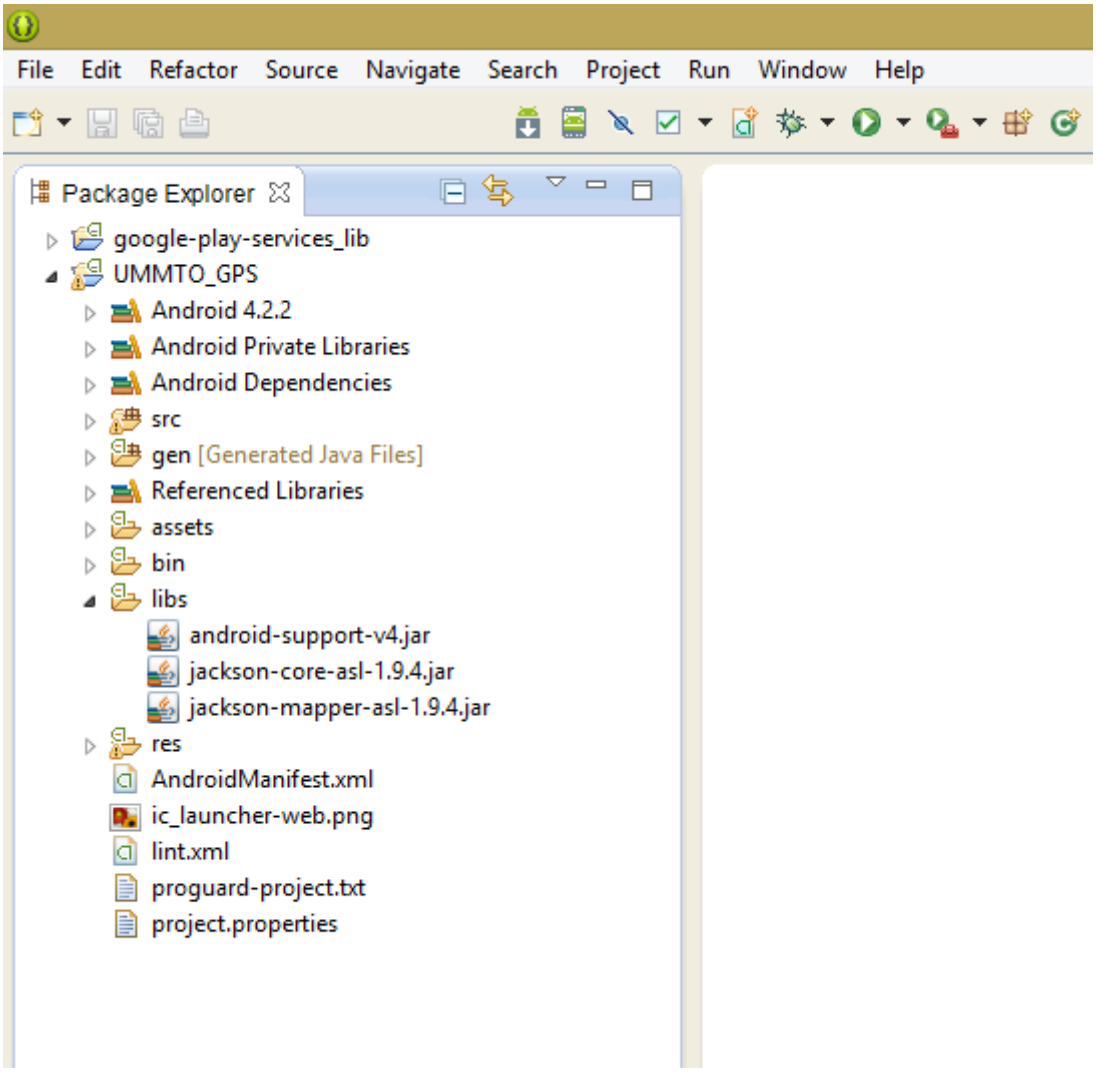

Figure 4.1. Utilisation de la bibliothèque jackson

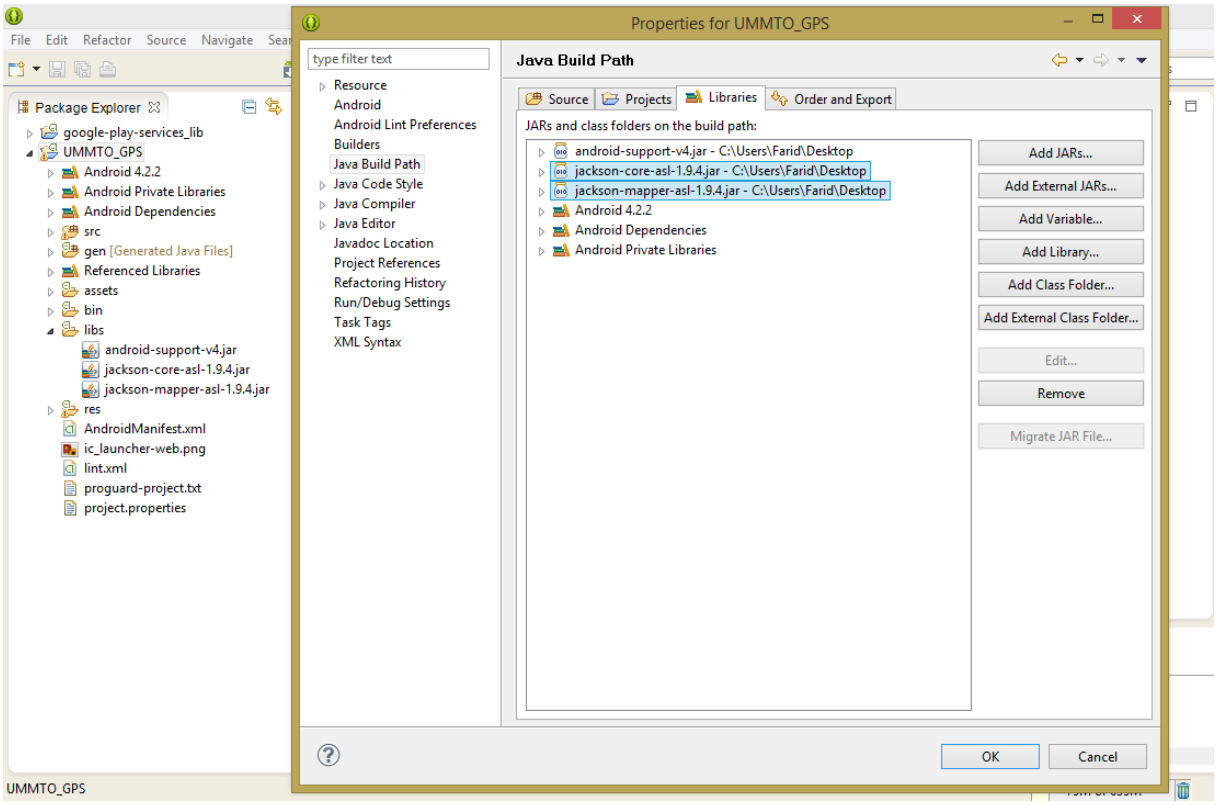

Figure 4.2. Référencer les archives jar comme bibliothèque dans notre project.

IV.4.2. Partie traitement :

Dans cette section, nous allons faire une brève description des différents packages implémentant notre application, ainsi les différentes classes qui la constituent.

IV.4.2.1 Structure de l'application :

Pour faciliter le développement et la modification prochaine de notre application nous avons choisi de la structurer en cinq packages séparés.

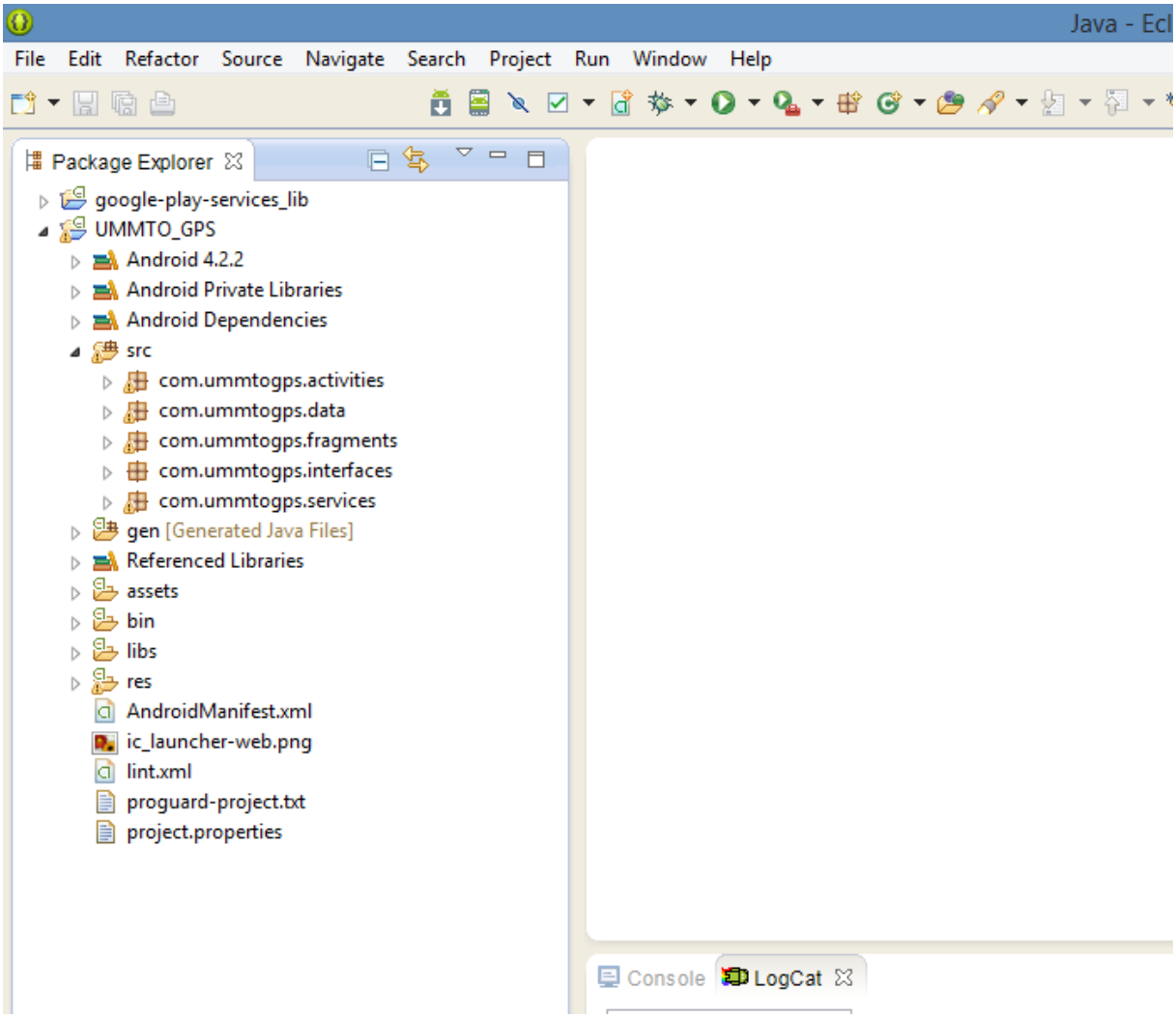

Figure 4.3. Structure de l'application.

IV.4.2.2 Le Package com.ummtogps.activities :

Ce package contient les activties de notre applications c'est-à-dire, les classes java qui étendent la classe Activity de Android

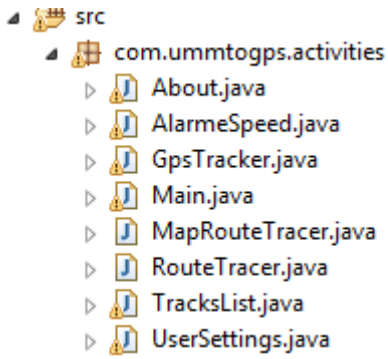

Figure 4.4. Package com.ummtogps.activities.

IV.4.2.3 Le Package com.ummtogps.data :

Ce package contient les classes java utilisées pour enregistrer ou fournir des données à l'application.

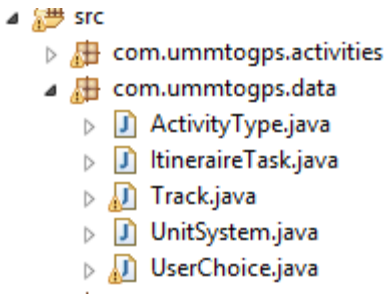

Figure 4.5 Package com.ummtogps.data

IV.4.2.4 Le Package com.ummtogps.fragments :

Ce package contient les classes java qui gèrent les fragments de notre application, tel que la liste des parcours enregistrer.

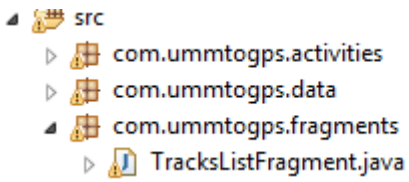

Figure 4.6 Package com.ummtogps.fragments

IV.4.2.5 Le Package com.ummtogps.interfaces :

Ce package contient deux interfaces, en parle pas d'interface graphique mais des class de type interface au niveau de programmation.

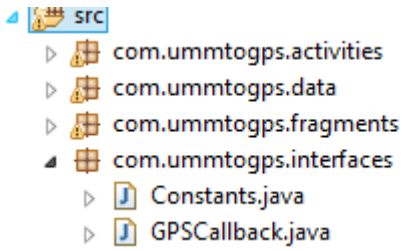

Figure 4.6 Package com.ummtogps.interfaces .

IV.4.2.6 Le Package com.ummtogps.services :

Ce package contient trois classes, les plus intéressantes, puisque c'est à ce niveau que la plus part des traitements de notre application sont réalisés.

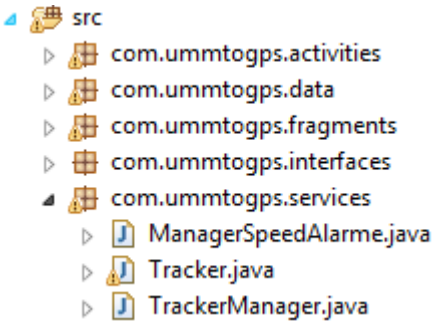

Figure 4.7 Package com.ummtogps.services

IV.4.3. Présentation de l'application:

Dans cette section nous allons vous présenter les écrans de notre application, ainsi que leurs fonctionnalités sous forme de zones de texte présentant les actions réalisées via cet élément.

- **Fenêtre de l'écran d'accueil :**

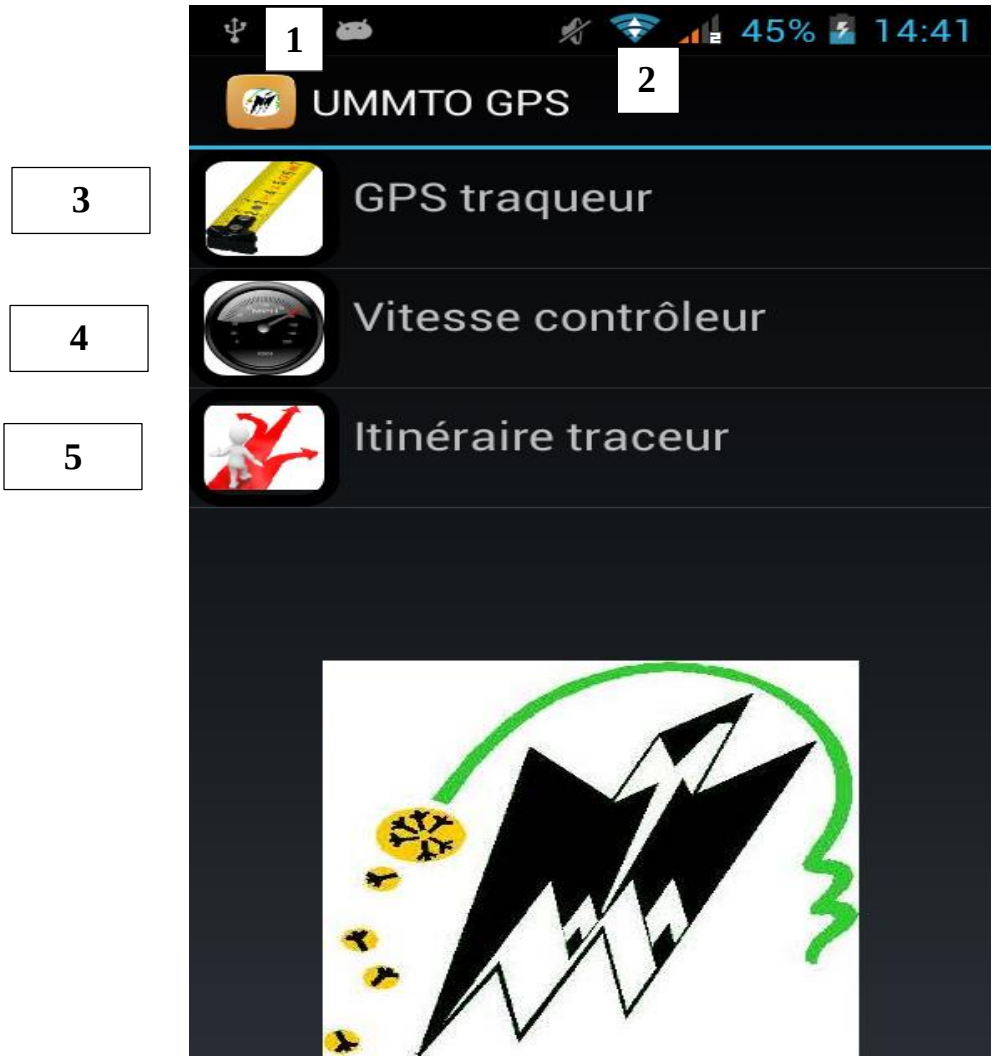

Figure 4.7. Écran d'accueil

1 : Icone de l'application

2 : Nom de l'application

3 : Option GPS traqueur.

4 : Option vitesse contrôleur.

5 : Option Itinéraire traceur.

➢ **Fenêtre d'écran GPS Traquer :**

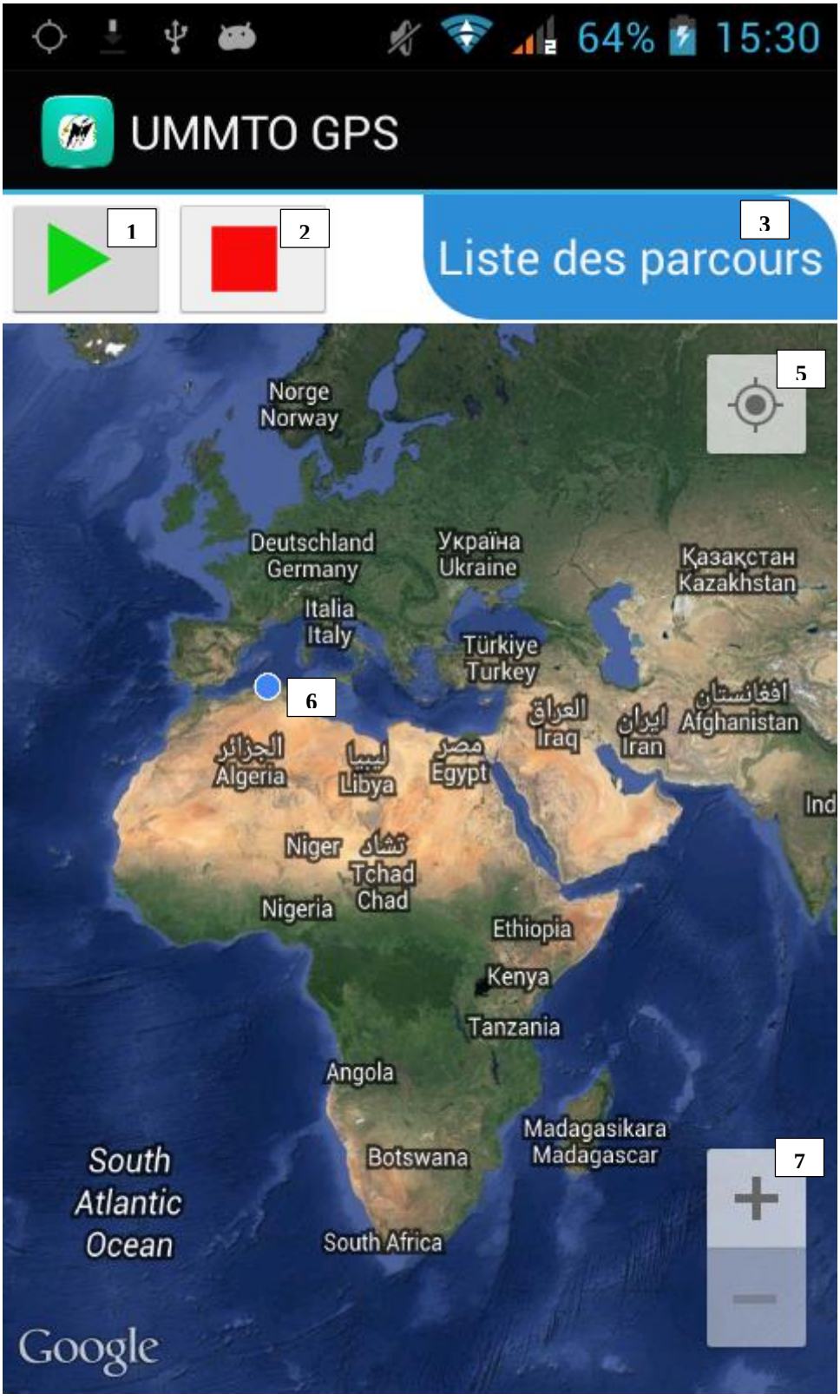

Figure 4.7. Écran GPS traqueur

1 : Démarre l'enregistrement d'un parcours.

2 : Arrête l'enregistrement.

3 : Affiche la liste des parcours enregistrés précédemment.

4 : Carte Google map.

5 : Bouton permis la localisation de l'utilisateur.

6 : Un point bleu indiquent la position de l'utilisateur.

7: Bouton de zoom sur la carte.

➢ **Fenêtre du choix type d'activité :**

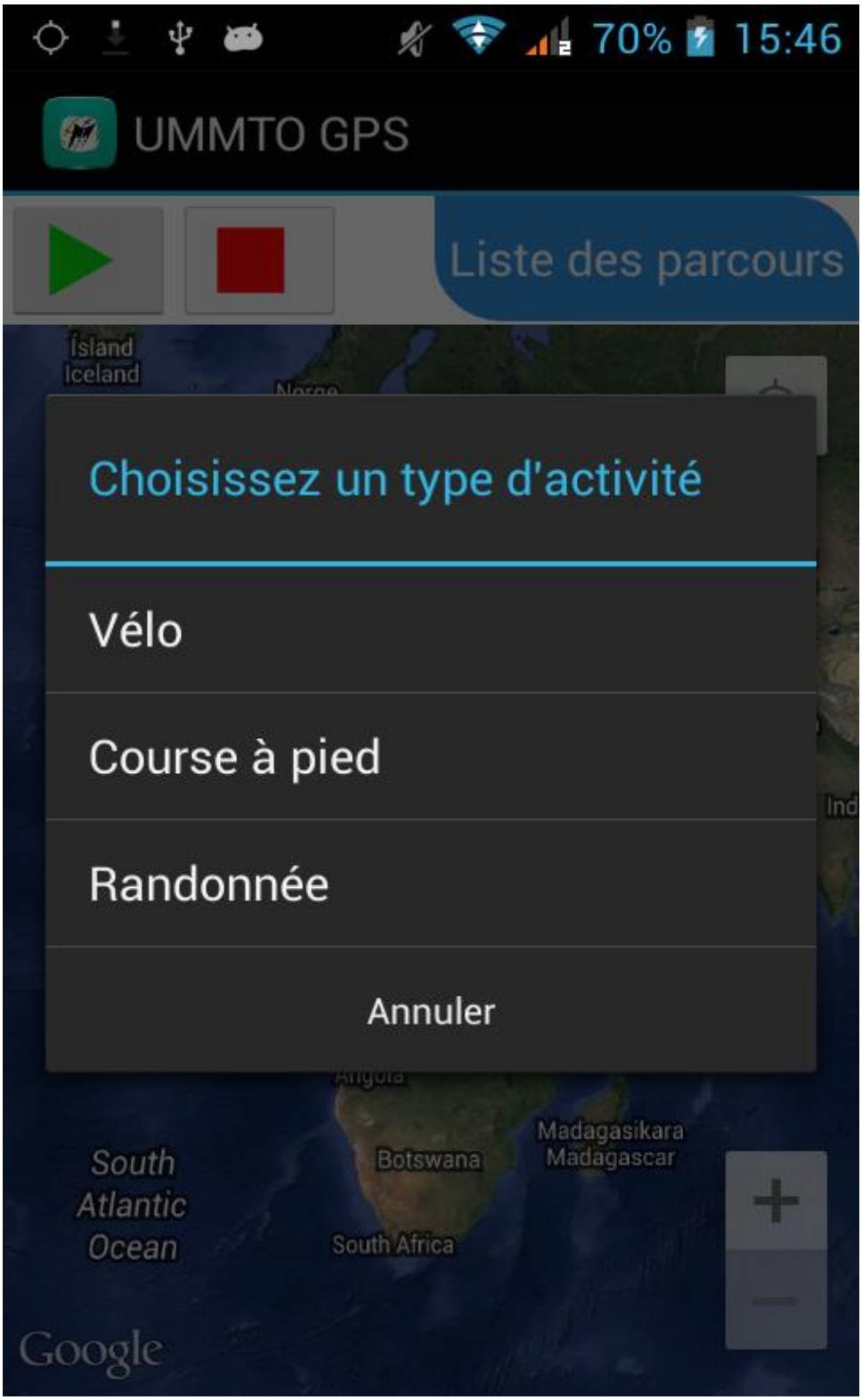

Figure 4.8 Fenêtre du Choix de type d'activité

1 : Boite de dialogue pour choisissez le type d'activité sportif.

2 : annuler l'enregistrement

➢ **Fenêtre du démarrage du l'enregistrement :**

Figure 4.9 du démarrage du l'enregistrement.

1 : Notification générer par l'application dans la barre d'taches indiquent que l'enregistrement est en cours.

2 : l'application nous fournit un message (notification) de confirmation du démarrage de l'enregistrement.

➢ **Fenêtre Arrêter l'enregistrement et lui donner un nom :**

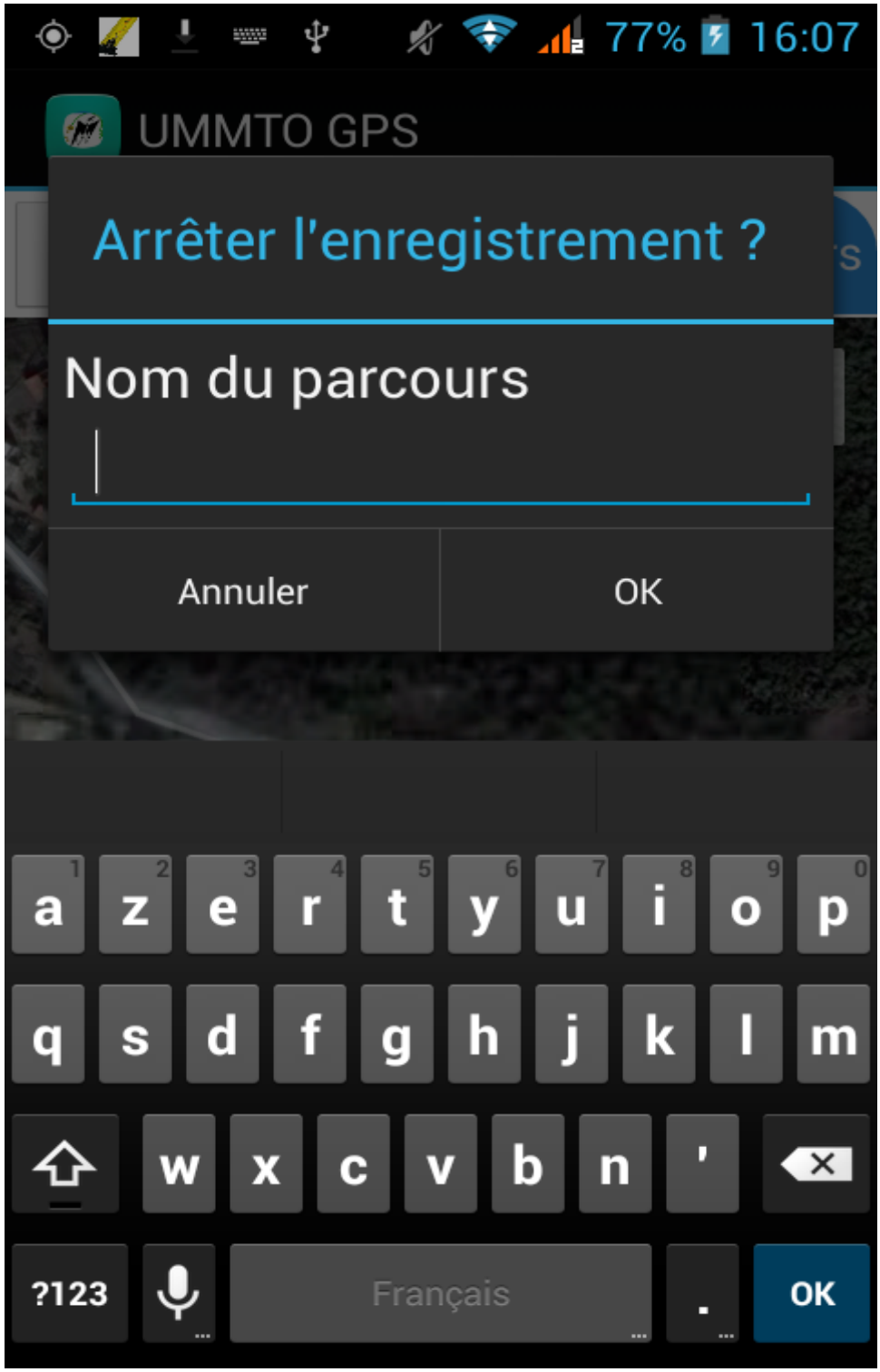

Figure 4.10 Fenêtre Arrêter l'enregistrement.

1 : Donner un nom au parcours.

2 : Confirmer l'arrêt de l'enregistrement.

3 : Annuler l'arrêt de l'enregistrement.

➢ **Fenêtre liste des parcours effectué :**

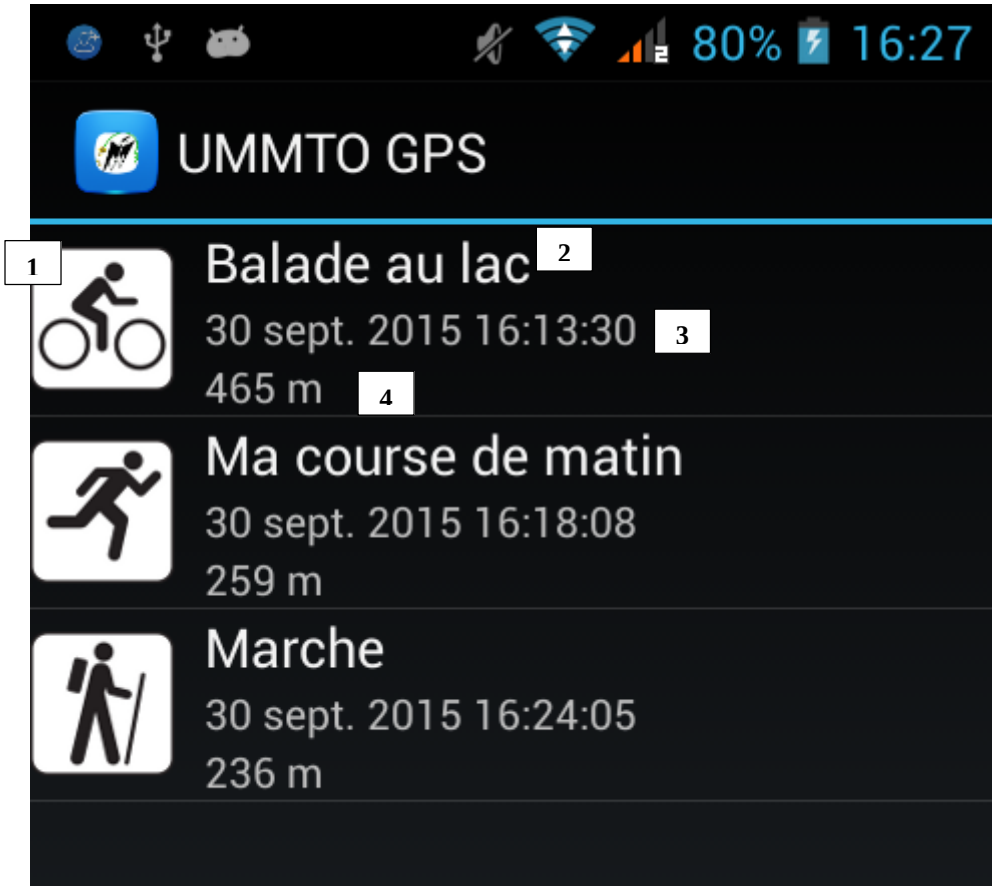

Figure 4.10 Fenêtre liste des parcours.

1 : Icone indiquent le type d'activité a là qu'elle se parcourt a été effectué.

2 : Nom de parcours.

3 : la date et l'heur de l'enregistrement du parcours.

4 : la distance parcourue du point du démarrage d'enregistrement jusqu'à à son arrêt.

➢ **Fenêtre de l'écran vitesse contrôleur :**

Figure 4.11 _Ecran vitesse contrôleur

1 : Boutin switch pour activer ou désactiver le système d'alarme.

2 : Champ de texte pour saisir la vitesse à ne pas dépasser.

3 : Vitesse à ne pas dépasser.

4 : Vitesse de déplacement de l'utilisateur en temps real**.**

➢ **Fenêtre de l'écran itinéraire traceur :**

Figure 4.12 Fenêtre de l'écran itinéraire traceur.

1 : Champ de texte pour saisir le point de départ.

2 : Champ de texte pour saisir le point d'arrivé.

3 : Bouton pour lancer la recherche de l'itinéraire.

➢ **Fenêtre de l'écran itinéraire trouvé :**

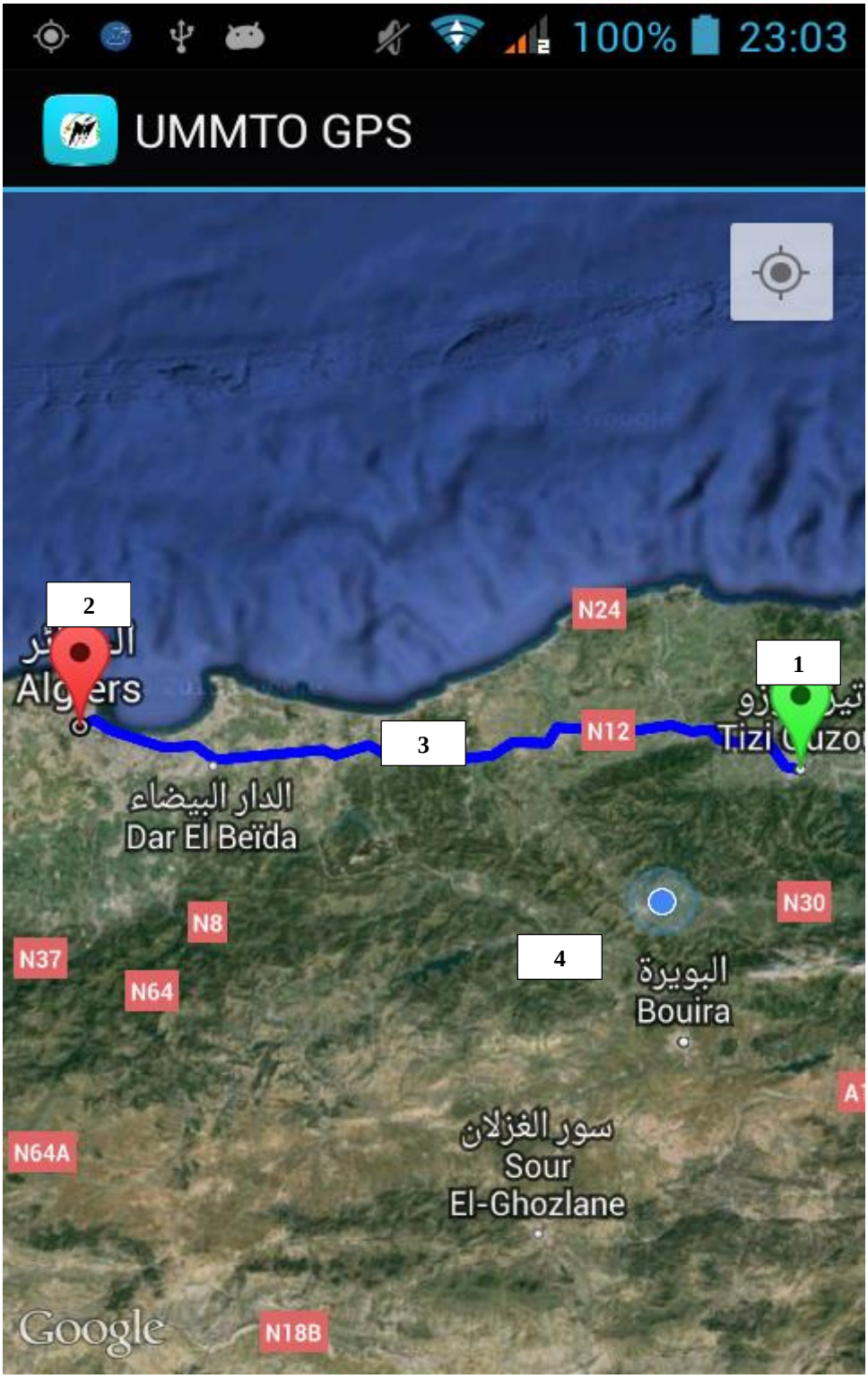

Figure 4.13 l'écran itinéraire trouvé.

1 : Point de départ.
2 : Point d'arrivé.
3 : La trajectoire à suivre.
4 : Position de l'utilisateur.

➢ **Fenêtre des fonctions cachée de l'application (menu) :**

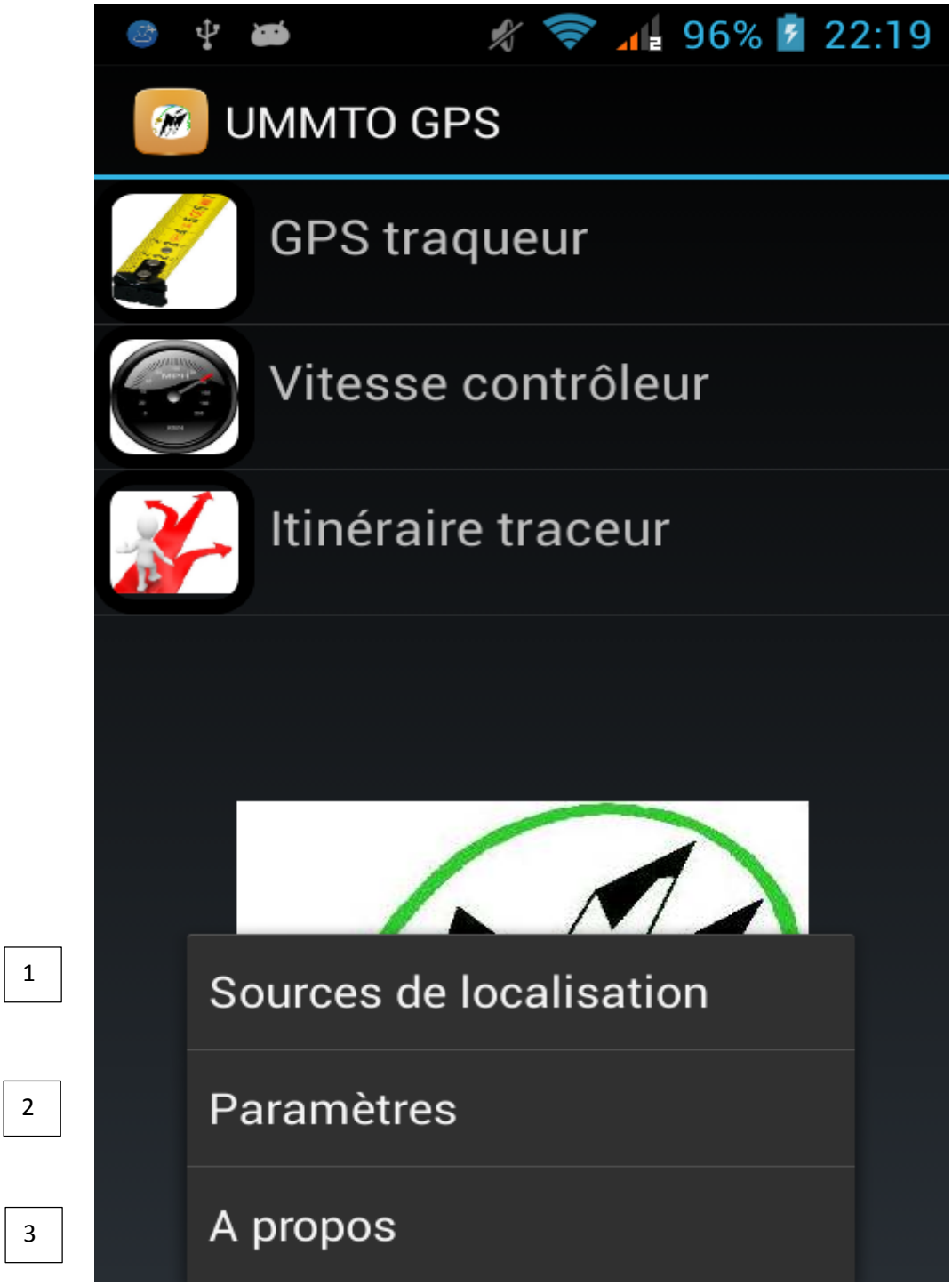

Figure 4.14 Les fonctions cachée de l'application (menu)

1: Ouvrir l'interface « paramètres de sécurités et de localisation » du l'appareil.

2 : Ouvrir l'interface des paramètres propres à l'application.

3 : Ouvrir une fenêtre web, qui donne des informations relative à l'application ainsi l'aide.

IV.5. Conclusion

Dans ce chapitre nous avons présenté les outils utilisés pour le développement de notre application, ensuite nous avons donné une description du fonctionnement de notre application en présentant ces différentes interfaces et les différents traitements que peut faire l'utilisateur.

Conclusion Générale :

L'élaboration de notre travail était dans le but de concevoir une application dédiée aux terminaux mobiles ou autre appareil disposant de la plateforme Android. Cette application permet au propriétaire du l'appareil d'enregistrer ses déplacements, de calculer la distance parcouru, de consulter l'historique des parcours effectué, de contrôler sa vitesse de déplacement, et de demander de tracé un itinéraire. Pour ce faire, nous avons recouru à différentes technologies et outils de localisation et d'orientation jugés nécessaires pour aboutir à l'objectif de notre application.

Notre application ainsi réalisée permet de :

- Démarrer l'enregistrement d'un parcours.
- Choisir un type d'activité.
- Calculer la distance parcourue.
- Exécuter l'application en arrière plant.
- Récupérer l'application à travers une notification.
- Arrêter l'enregistrement.
- Enregistrer un parcourt.
- Afficher la liste détaillée des parcours effectués.
- Déclencher une alarme si on dépassé la vitesse donner.
- Tracer un itinéraire d'un point de départ a un point d'arrivé.
- Modifier la source de localisation

Ce projet nous avait donnés de plus l'occasion d'acquérir des nouvelles connaissances à propos d'UML, Eclipse, et de maitriser le langage de programmation java qui sera certes utiles dans notre future vie professionnelle.

Enfin, on peu pas sans doute affirmer que notre travail est complet d'où plusieurs améliorations peuvent être apportées. Mais, nous espérons au moins que nous avons réussi à réaliser une simple application mobile Android fonctionnelle qui satisfait aux besoins des futurs utilisateurs et qui convient à leurs attentes.

Les perspectives :

En guise de perspectives nous envisageons d'enrichir notre application avec de nouvelles fonctionnalités comme :

1. **Elargir les cibles potentielles :**

   Tout d'abord, il serait intéressant d'ajouter un module de reconnaissance vocale pour permettre à des personnes déficientes visuelles d'utiliser l'application. De plus, proposer à l'utilisateur le choix de la langue peut être très utile.

2. **Ajout de nouvelles fonctionnalités :**

   Il est envisageable d'enrichir l'application par des services utiles à chaque étudiants.

   Une version widget de l'application, permettrait un accès plus rapide à une des fonctionnalités de l'application car il est accessible directement via le bureau de son smart phone. Par exemple ce widget pourrait servir à récupérer le nom du parcours et la distance parcouru et son chemin. Il s'agit en fait de déterminer quelle est la fonctionnalité la plus utilisée de l'application et d'en faire un widget.

3. **Mode Offline :**

   Un deuxième mode sera envisageable, afin que l'utilisateur puisse utiliser l'application sans se soucie s'il est connecté sur internet ou pas.

Référence bibliographique :

[1] : Programmation Android, Damien Guignard, Julien Chable, Emmanuel Robles, Nicolas Sorel, Eyrolles, 1 juil. 2011

[2] : développez pour Android, Cyril Mottier, Ludovic Perrier, Digit Books 2011

[3] : Algorithme de géolocalisation intérieure par différenciation de signaux wifi, Yacine Mezali, 2012

[4] : Guide pratique du GPS, Paul Correia, Eyrolles, 2012, édition 6

[5] : Les media géolocalisés, Nicolas Nova, FYP éditions 2009

[6] : Portail des développeurs Android [En ligne]http://developer.android.com/.

[7] : Référence du SDK Android[En ligne] http://developer.android.com /sdk/ndk/1.5_r1/ index.htm.

[8] : UML2 pour les développeurs, Xavier Blanc, Isabelle Mounier, Eyrolles 2006.

[9] : Penser objet avec UML et java, Michel Lai, Dunod 2004.

[10]: http://jackson.codehaus.org/

Webographie

[W1]: http :// www.eclipse.org/downloads/. Accéder en janvier

[W2]: http :// developer.android.com/index.html.

[W3]: https :// dl-ssl.google.com/android/eclipse/.

[W4]: http :// www.pointgphone.com/android/emulateur.

[W5]: http: // code.google.com/intl/fr-FR/android/maps-api-signup.html.

[W6]: http :// www.mti.epita.fr/blogs/2010/08/03/introduction-a-la-programmation-sous-android/.

[W7]: http :// developer.android.com/guide/topics/location/index.html. Accéder en juin

[W8]: http :// www.anddev.org/.

Annexe A : Manuel d'installation et de mise en marche.

Installation de l'Eclipse [W1].

2. Installation du SDK Android [W2].

3. Installation du plugin ADT sous Eclipse [W3].

4. Création d'un Android Virtual Device(AVD) :

La Configuration d'un nouveau AVD se fait en remplissant les champs suivants lors de sa création:

- ✓ Name : Le nom à donner à votre émulateur (sans espace).
- ✓ Target : La version du SDK Android sur lequel l'émulateur doit fonctionner (Dans notre cas elle doit être de type Google APIs pour pouvoir faire fonctionner le programme).
- ✓ SD Card: Configuration de la SD Card (Taille, etc.). Ce champ est facultatif.
- ✓ Skins : Choix du thème de l'émulateur. Des émulateurs préconfigurés se trouvent dans Built-in.
- ✓ Hardware: Cette partie permet de rajouter le matériel et de le personnaliser.

Annexe B : Obtention d'une clé pour utiliser Google Maps.

J'en ai besoin de cette configuration pour toute la partie concernant la manipulation de carte via l'API Google Maps. Une étape supplémentaire sera alors nécessaire : l'obtention d'une clé de licence.

Pour cela, la première étape consiste à se rendre sur le site proposé par Google (http://code.google.com/android/maps-api-signup.html.). Il nous faudra également un compte Google pour compléter l'opération. Enfin, pour obtenir cette clé, l'interface du site nous demande une empreinte MD5 d'un certificat :

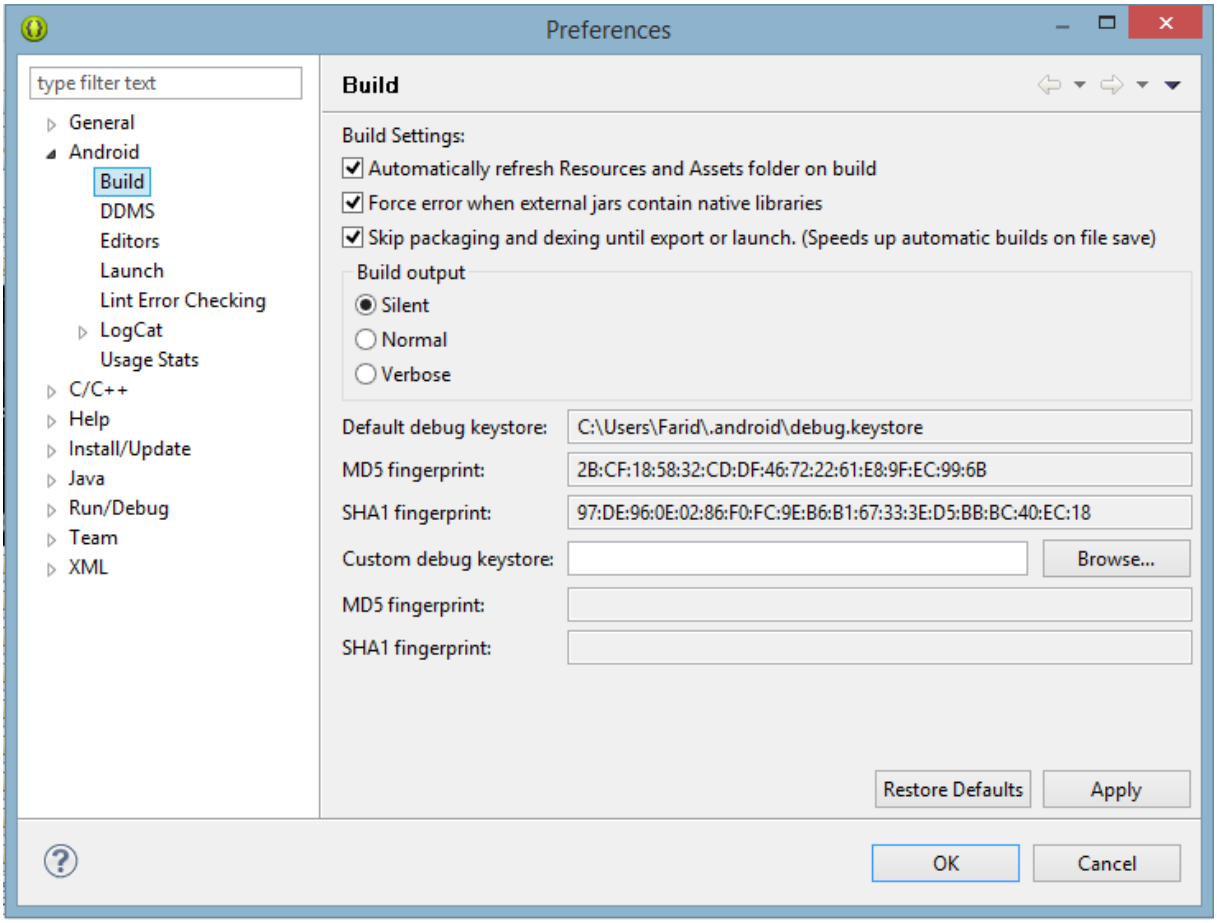

Figure .A.1 fenêtre pour récupérer l'empreinte MD5

Android n'autorise l'installation que des applications signées. Avant d'installer une application via l'émulateur, Eclipse signe notre application en utilisant un certificat de débogage qui est fourni avec le SDK d'Android.

Eclipse crée le fichier ***debug. Keystore*** lors de la compilation du projet, ce fichier étant stocké dans le répertoire ***.android*** dont le chemin varie en fonction du système d'exploitation.

La clé générée pour utiliser l'API Google Maps est basée sur ce certificat de débogage.

Une fois l'empreinte saisie dans le formulaire et envoyée au site, une clé est générée ainsi qu'un morceau de code XML montrant un exemple d'utilisations.