II.1 Introduction :

Les smartphones sont aujourd'hui omniprésents dans nos vies, et les tablettes commencent à se démocratiser. Ce sont de véritables mini-ordinateurs qui permettent d'exécuter des applications plus ou moins sophistiquées. Les éditeurs des systèmes d'exploitation pour ces appareils incitent les développeurs à créer des applications, et fournissent pour cela des outils pour faciliter le développement.

L'objectif de ce chapitre est de présenter le système d'exploitation Android, et l'outil de développement Android studio. Des outils que nous allons utiliser pour développer ce projet.

II.2 Présentation générale :

Android est un système d'exploitation open source édité par Google pour appareils embarqués et/ou mobiles, comme les smartphones ou les tablettes. On le retrouve aussi dans certains GPS, ordinateurs de bord de voitures, dans des télévisions, autoradios, et même des montres. De plus, de nombreux prototypes d'appareils électroménagers, comme des réfrigérateurs ou des machines à laver, fonctionnant sous Android ont été présentés ces derniers temps, permettant ainsi de lancer une machine à l'aide de son téléphone, ou encore d'être prévenu par son frigo lorsqu'il manque certaines choses.

Le système Android est basé sur un fork du noyau de Linux. Ce dernier a été modifié pour être plus adapté aux terminaux mobiles ayant peu de puissance de calcul, de mémoire et de batterie. De fait, certaines librairies standards ne sont pas supportées par le système, et des améliorations ont été apportées sur la gestion de l'énergie. Les applications sont écrites en Java, et fonctionnent au sein d'une machine virtuelle Dalvik. Cette machine virtuelle a elle aussi été modifiée pour être le plus adapté possible aux appareils de faible puissance. Ainsi, beaucoup d'efforts ont été fait sur la consommation de mémoire, qui a été largement diminuée par rapport à la machine virtuelle java classique.

D'autre part, ce système est open source, ce qui permet à n'importe qui de lancer sa propre version d'Android. De nombreuses personnes utilisent ainsi des "ROM custom", c'est à dire des versions modifiées par rapport au code de base fourni par l'éditeur. Toutefois, on peut se demander si Google joue le jeu : en effet, le code source de la version 3.0 (Honeycomb), dédiée aux tablettes, n'a jamais été fourni. Officiellement, c'était pour éviter que les constructeurs rajoutent une surcouche (c'est-à-dire un thème différent d'autres logiciels, etc...).

De plus, certaines librairies sont propriétaires, et les développeurs ne sont pas tous égaux devant le géant de Mountain View : un petit groupe est en effet privilégié et bénéficie des dernières mises à jour du SDK, tandis que les autres sont un peu oubliés.

II.3 Historique :

L'histoire d'Android commence en octobre 2003, où la société Android Inc. est créée. Officiellement, elle développe des logiciels pour mobiles. Mais en réalité, elle se préparait à sortir un tout nouveau système d'exploitation pour smartphones. En 2005, Google rachète cette entreprise, et sort une première béta en novembre 2007, avant de lancer la version 1.0 en septembre 2008 avec le HTC Dream. À partir de ce moment-là, le rythme des nouvelles sorties est très élevé : pas moins de 11 versions différentes sont sorties en 3 ans ! On remarquera au passage que chaque version d'Android porte le nom d'un dessert.

Figure II.1: *Les différentes versions d'Android.*

II.4 Etude de la plateforme Android :

Android est un système d'exploitation pour Smartphones, tablettes tactiles, PDA et terminaux mobiles. C'est un système open source, utilisant le noyau Linux, conçu par Android, une startup rachetée par Google, et annoncé officiellement le 5 novembre 2007. D'autres types d'appareils possédant ce système d'exploitation existent, par exemple des téléviseurs, des montres, des autoradios et même des voitures( ).

Android est gratuit, pour les constructeurs d'appareils souhaitant l'utiliser, et partiellement open-source. Concrètement, ce système d'exploitation s'appuie sur un noyau Linux optimisé pour un usage mobile, et sur une machine virtuelle Java assez fortement modifiée, nommée Dalvik JVM. Celle-ci n'exécute pas les (.class) habituels, mais des fichiers portant l'extension (.dex), compilés différemment et optimisés par le SDK Android.

Le développement se fait donc en Java, mais sur cette JVM spécifique à Android. Il n'est, du fait, pas possible d'utiliser n'importe quelle librairie Java dans une application Android.

II.4.1 Architecture de la plateformeAndroid :

L'architecture de la plateforme Android se décline, selon une démarche bottom up, en quatre principaux niveaux que sont :

- le noyau linux
- les librairies et l'environnement d'exécution,
- le module de développement d'applications
- les différentes applications.

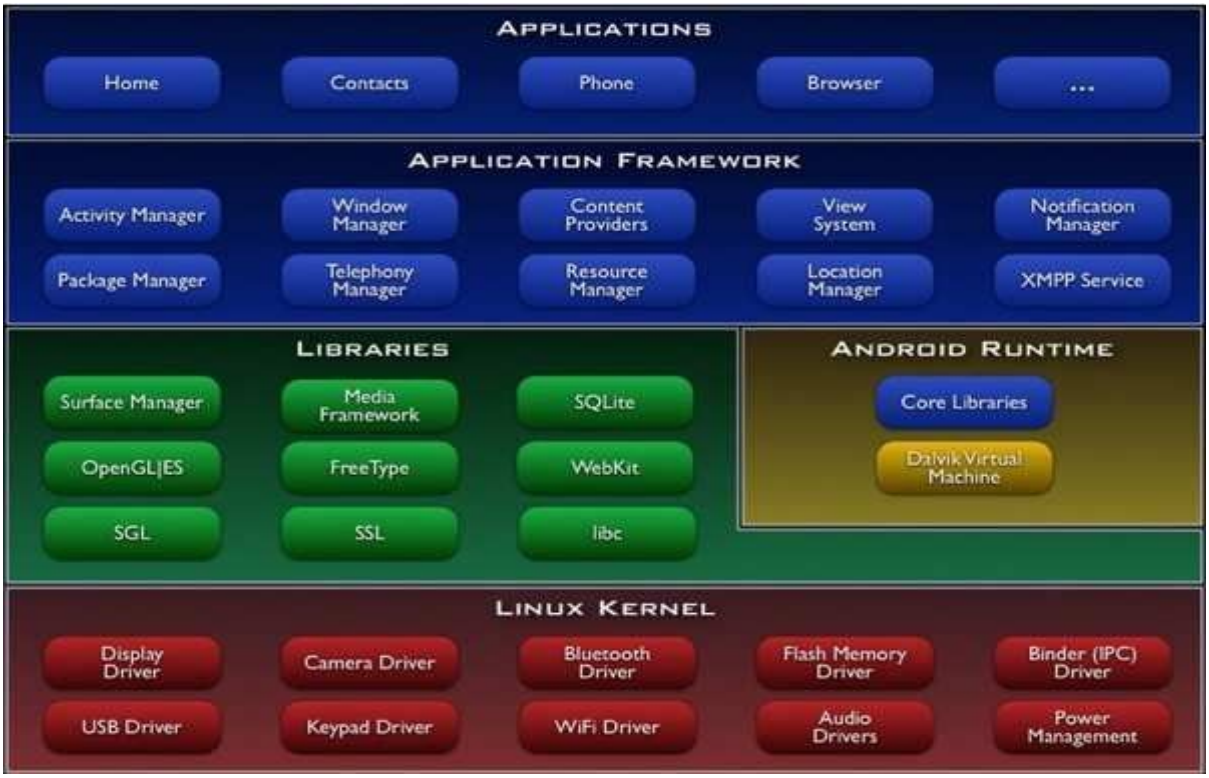

Figure II.2 :*L'architecture de la plateforme Android.*

II.4.1.1 Premier niveau : noyauLinux

Android s'appuie sur un noyau (Kernel en anglais) dont la version est Linux 2.6. Pour être précis, le noyau est l'élément du système d'exploitation qui permet de faire le pont entre la partie matérielle et la partie logicielle. D'ailleurs, si vous regardez attentivement le schéma, vous remarquerez que cette couche est la seule qui gère le matériel. La version du noyau utilisée avec Android est une version conçue spécialement pour l'environnement mobile, avec une gestion avancée de la batterie et une gestion particulière de la mémoire.

II.4.1.2Deuxième niveau : librairies et environnementd'exécution.

II.4.1.2.1 Leslibrairies :

Ces bibliothèques proviennent de beaucoup de projets open-sources, écris en C/C++ pour la plupart, comme SQLite pour les bases de données, Web Kit pour la navigation web ou encore OpenGL, afin de produire des graphismes en 2D ou en 3D.

II.4.1.2.2 L'environnementd'exécution :

Le runtime Android est basé sur le concept de machine virtuelle, utilisée en Java. Étant donné les limitations des dispositifs (peu de mémoire et la vitesse du processeur), il n'a pas été possible d'utiliser une machine virtuelle Java standard. Google a pris la décision de créer une nouvelle machine virtuelle Dalvik, afin de mieux répondre à ces limitations. Dans cette partie, qui est l'environnement d'exécution, on retrouve essentiellement deux parties :

II.4.1.2.2.1 CoreLibrairies :

Les Cores librairies fournissent le langage Java, disponible pour les applications. Le langage Java fournit avec Android reprend en grande partie l'API JSE 1.5. Il y a des choses qui ont été mises de côté, car cela n'avait pas de sens pour Android (comme les imprimantes, swing, etc.) et d'autres APIs spécifiques requises pour Android ont étérajoutées.

II.4.1.2.2.2 Dalvik :

Les applications Java, développées pour Android, doivent être compilées au format Dalvik exécutable (.dex) avec l'outil (dx). Cet outil compile les (.java) en (.class) et ensuite il convertit ces (.class) en (.dex).

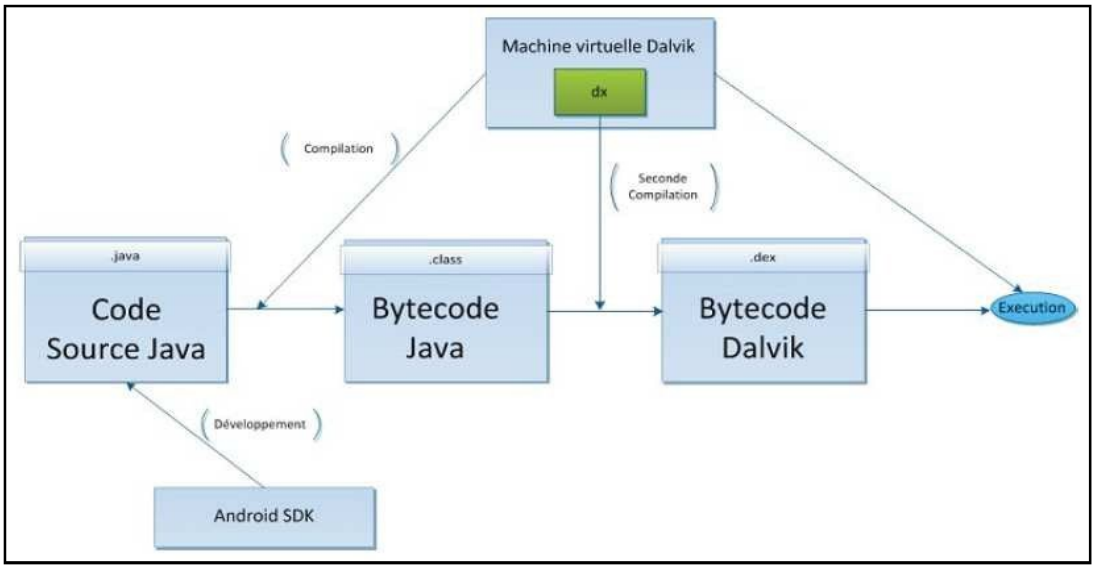

Figure II.3: *La machine virtuelle Dalvik.*

II.4.1.3Troisième niveau : module de développementd'applications

Le Framework est situé au-dessus de l'Android Runtime et des librairies. Il fournit des API permettant aux développeurs de créer des applications riches. A ce niveau on distingue deux types de service :

II.4.1.3.1 Core PlateformServices :

Android introduit la notion de services. Un service est une application, qui n'a aucune interaction avec l'utilisateur et qui tourne en arrière-plan. Les services cœurs de la plateforme (Core Plateform Services) fournissent des services essentiels au fonctionnement de la plateforme :
Activity Manager : gère le cycle de vie des applications et maintient une pile de navigation permettant d'aller d'une application à une autre.
Package Manager : utilisé, par l'Activity Manager, pour charger les informations provenant des fichiers (.apk) (Android package file);

Window Manager : il gère les fenêtresdesapplications(quelle fenêtredoitêtre affichée devant une autre à l'écran).

Resource Manager : gère tous ce qui n'est pas du code, toutes les ressources (images, fichier audio, etc.).

Content Provider : gère le partage de données entreapplications.

View System : fournit tous les composants graphiques : listes, grilles, boutons,etc...

II.4.1.3.2 HardwareServices :

Les services matériels (Hardware Services) fournissent un accès vers les API matérielles :

Telephony Service : permet d'accéder aux interfaces téléphoniques (GSM, 3G, etc.).

Location Service : permet d'accéder au GPS.

Bluetooth Service : permet d'accéder à l'interface Bluetooth.

Wi-Fi Service : permet d'accéder à l'interface Wifi.

USB Service : permet d'accéder aux interfaces USB.

Sensor Service : permet d'accéder aux capteurs (capteur de luminosité, de proximité, d'accélération,etc.).

II.4.1.4 Quatrième niveau :applications

Il s'agit tout simplement d'un ensemble d'applications que l'on peut trouver sur Android, par exemple, les fonctionnalités de base incluent un client, pour recevoir et envoyer des emails, un programme, des SMS, un calendrier, un répertoire,etc…

II.5 Le Développement sousAndroid :

II.5.1 L'environnement de développement sousAndroid :

Afin de développer des applications sous Android, un ensemble d'outils est nécessaire. Vue que les procédures d'installation de ces outils sont assez longues et fastidieuses, alors les décrire, pas à pas, risquerai de prendre énormément de place et ainsi beaucoup de pages. Alors, on se contentera juste d'évoquer les outils et leurintérêt.

II.5.2 Le JDK (Java DeveloppmentKit) :

Les applications développées pour Android étant essentiellement écrites en langage java, un langage de programmation orienté objet qui a la particularité d'être très portable. Cela signifie qu'un programme Java, fonctionnant sur Windows (par exemple), pourra facilement tourner sur Mac ouGNU/Linux.

Cette petite prouesse vient du fait que Java s'appuie sur une machine virtuelle pour s'exécuter (appelée la JVM). Pour avoir une JVM sur votre ordinateur, il vous faut télécharger le JRE. Ce dernier contient, en plus de la JVM, des bibliothèques Java standards.

La JVM ne lit pas directement le code Java. Elle lit un code compilé (le byte code). Pour passer du code Java, que le développeur écrit, au code compilé, lu par la JVM, des outils spéciaux sont nécessaires. Ces outils sont inclus dans le JDK. De plus, le JDK contient le JRE (et donc la machine virtuelle), ce qui est bien pratique

Pour résumer, on dira que :

Pour un simple utilisateur de Java : il doit avoir le JRE;

Pour un développeur : il aura besoin des outils du JDK(22).

II.5.3 Le SDK *(*Software Development Kit*)*Android :

Un SDK, c'est-à-dire un kit de développement logiciel, est un ensemble d'outils que met à disposition un éditeur afin de permettre de développer des applications pour un environnement précis. Le SDK Android permet, donc, de développer des applications pour Android et uniquement pour Android.

Au premier lancement du SDK, un écran semblable à la figure suivante s'affichera comme dans la figure suivante :

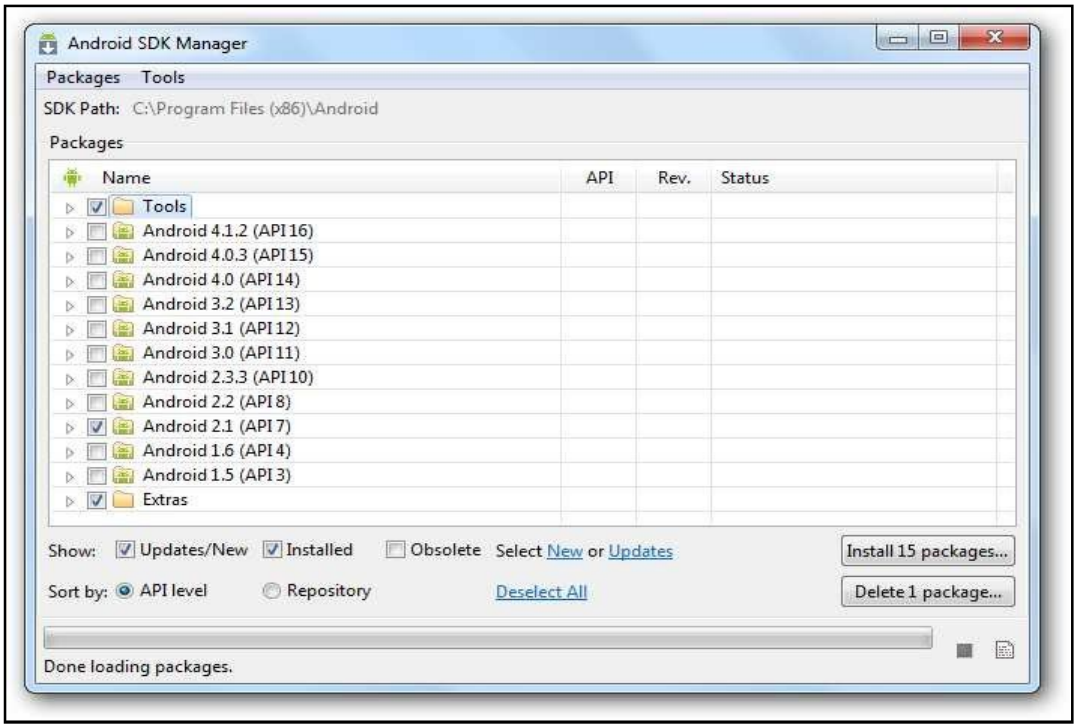

Figure II.4: *Le SDK Android.*

En regardant bien le nom des paquets, vous remarquerez qu'ils suivent tous un même motif. Il est écrit à chaque fois : Android [un nombre] (API [un autre nombre]). La présence de ces nombres s'explique par le fait qu'il existe plusieurs versions de la plateforme en circulation. Le premier nombre correspond à la version d'Android et le second à la version de l'API Android associée. Quand on développe une application, il faut prendre en compte ces numéros, puisqu'une application développée pour une version précise d'Android ne fonctionnera pas pour les versionsantérieures.

II.5.4L'IDEEclipse :

Eclipse est un environnement de développement intégré. C'est un logiciel qui permet d'écrire un programme beaucoup plus facilement qu'avec le simple Bloc-notes. Outre la coloration du code, il permet d'apporter des outils très pratiques pour compiler vos programmes, les déboguer, etc. Il peut être utilisé pour programmer avec n'importe quel type de langage, mais nous l'utiliserons pour faire du Java.

De plus, Eclipse est conçu pour pouvoir être complété avec des plugins (extension). Ainsi, il existe un plugin pour développer des applications Android que nous verrons dans la partie suivante.

II.5.4.1 Le plugin ADT pourEclipse :

Google fournit un plugin pour Eclipse, nommé ADT (Android Development Tools), la fonction principale de ce plugin est de créer un pont entre Eclipse et le SDK Android.

II.5.5 La migration de l'éclipse vers Android Studio :

Jusqu'à Mai 2013, pour développer des applications pour Android, Google mettait en avant l'utilisation d'Eclipse couplé avec le Plugin ADT (Android Development Tools). Cette première solution a tout de même permis à Google de posséder le Store d'application le plus riche. Eclipse est un IDE qui a été développé par IBM puis est passé open source en 2001, la Fondation Eclipse gère maintenant l'IDE. Eclipse possède les avantages d'être modulable ainsi que multi plate-forme.

C'est durant la Google I/O de 2013, que Google a montré la première version d'Android Studio. En Access preview au départ pour sa version 0.1, puis passé en béta en juin 2014 pour la version 0.8, jusqu'à l'arrivé à la version 1.5 utilisée dans notre projet, cet IDE n'a pas été développé de zéro mais est basé sur l'IDE de JetBrains, IntelliJ IDEA. Cette société propose de nombreux IDE pour différents langages (PHP Storm, RubyMine, ...) mais qui sont tous payant. Dans sa dernière

version, Android Studio offre toutes les possibilités nécessaires pour développer une application Android complète.

Android Studio permet principalement d'éditer les fichiers Java et les fichiers de configuration d'une application Android.

Il propose entre autres des outils pour gérer le développement d'applications multilingues et permet de visualiser la mise en page des écrans sur des écrans de résolutions variées Simultanément.

Figure II.5: *Vue globale d'Android studio*.

II.5.5 Architecture d'un projet sous Android :

Les projets ont changé d'architecture depuis le passage à Android Studio, où l'on utiliserait plusieurs projets sous Eclipse, avec des dépendances entre chacun, nous parlerons maintenant d'un seul Projet, contenant plusieurs modules.

Les modules ont la forme suivante :

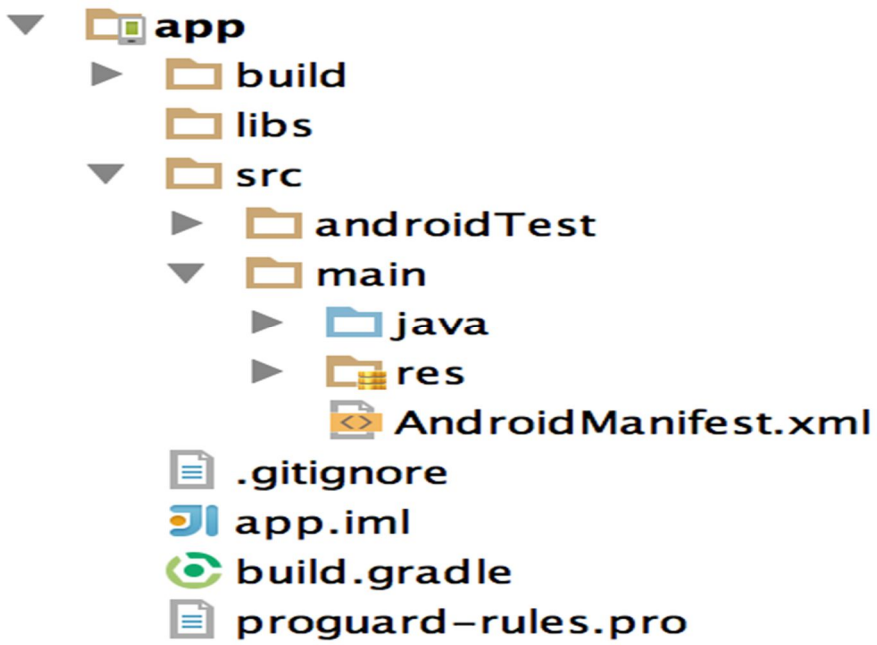

Figure II.6 : *Architecture d'un projet sous Android.*

Tout comme bon nombre de technologies actuelles, les sources d'une application Android possèdent une structure bien définie qui doit être respectée. Ces arborescences permettent non seulement de rendre les projets plus lisibles et organisés, mais aussi de simplifier le développement. Lors de la création d'un nouveau projet, voici l'arborescence qui est automatiquement générée :

AndroidManifest.xml : fichier XML décrivant l'application et ses composants, tels que les activités, les services, etc. Lors de la création d'une activité, une erreur courante pour un premier projet Android est d'oublier de la déclarer dans le fichier Manifest. C'est une étape indispensable pour le fonctionnement de l'application. Le Manifest est, en quelque sorte, la carte d'identité de l'application, et permet d'autoriser l'exécution des activités et autres actions del'application.

res : répertoire contenant toutes les ressources telles que les images, les vues de l'interface graphique, etc., nécessaires à l'application. Ce répertoire est structuré par défaut de la manière suivante:

res/drawable : contient les ressources de type image.

res/layout : contient les descriptions des interfaces graphiques au format XML (les vues).

res/xml : contient les fichiers XML supplémentaires (non présents par défaut).

res/menu : contient la description des menus, composants très courants d'une vue.

res/values : contient diverses ressources, telles que les textes, qui sont empaquetées sans aucuntraitement.

Le fichier build.gradle: sert dans la configuration pour le nouveau moteur de production nommé Gradle, qui sera utilisé pour construire notre application afin de la déployer sur notre smartphone ou sur le Play Store.

Au moment de la compilation du projet, l'application finale est générée au format APK, dans le répertoire bin de l'arborescence. C'est ce fichier qu'il faut ensuite déployer sur les équipements, afin de pouvoir faire tourner l'application.

II.5.6 Composantes d'une applicationAndroid :

Une application Android se compose de plusieurs éléments. Dans ce qui suit, nous allons essayer de découvrir les plus importants :

II.5.6.1 LesActivités :

Une activité est la composante principale pour une application Android. Elle représente l'implémentation et les interactions desinterfaces.

Plusieurs choix se proposent pour mettre en place l'interface visuelle :

Utiliser un fichier XML pour décrire l'interface.

Créer les éléments de l'interface à l'intérieur du codejava.

II.5.6.2 LesServices :

Un Service est, en fait, un programme tournant en tâche de fond et n'ayant pas d'interface graphique. L'exemple commun illustrant cette notion, est celui du lecteur mp3. Un lecteur mp3 ne nécessite pas, pour la plupart du temps, d'interface graphique et doit tourner en tâche de fond, laissant la possibilité aux autres applications de s'exécuter librement. Un service peut être lancé à différents moments:

Au démarrage du téléphone.

Au moment d'un événement (arrivée d'un appel, SMS, mail, etc…).

Lancement de l'application.

Action particulière dansapplication.

II.5.6.3 LesBroadcastReceivers :

Un BroadcastReceiver, comme son nom l'indique, permet d'écouter ce qui se passe sur le système ou sur votre application et de déclencher une action que vous aurez prédéfinie. C'est souvent par ce mécanisme que les services sontlancés.

II.5.6.4 Les Contentproviders :

Les Content Provider sont, comme l'exprime leur nom, des gestionnaires de données. Ils permettent de partager l'information entre applications. Vous pouvez accéder :
Aux contacts stockés dans le téléphone.
A l'agenda.
Aux photos.
Ainsi que d'autres données depuis votre application grâce aux contentproviders.

II.5.6.5 LesIntents :

Les Intents sont des objets permettant de faire passer des messages contenant de l'information entre composants principaux. La notion d'Intent peut être vue comme une demande de démarrage d'un autre composant, d'une action à effectuer.

II.5.7 Cycle de vie d'une application Android :

Le diagramme d'état suivant présente les principaux états du cycle de vie d'une activité Android, il est suivi d'une description des principales méthodes événementielles du cycle de vie d'une activité.

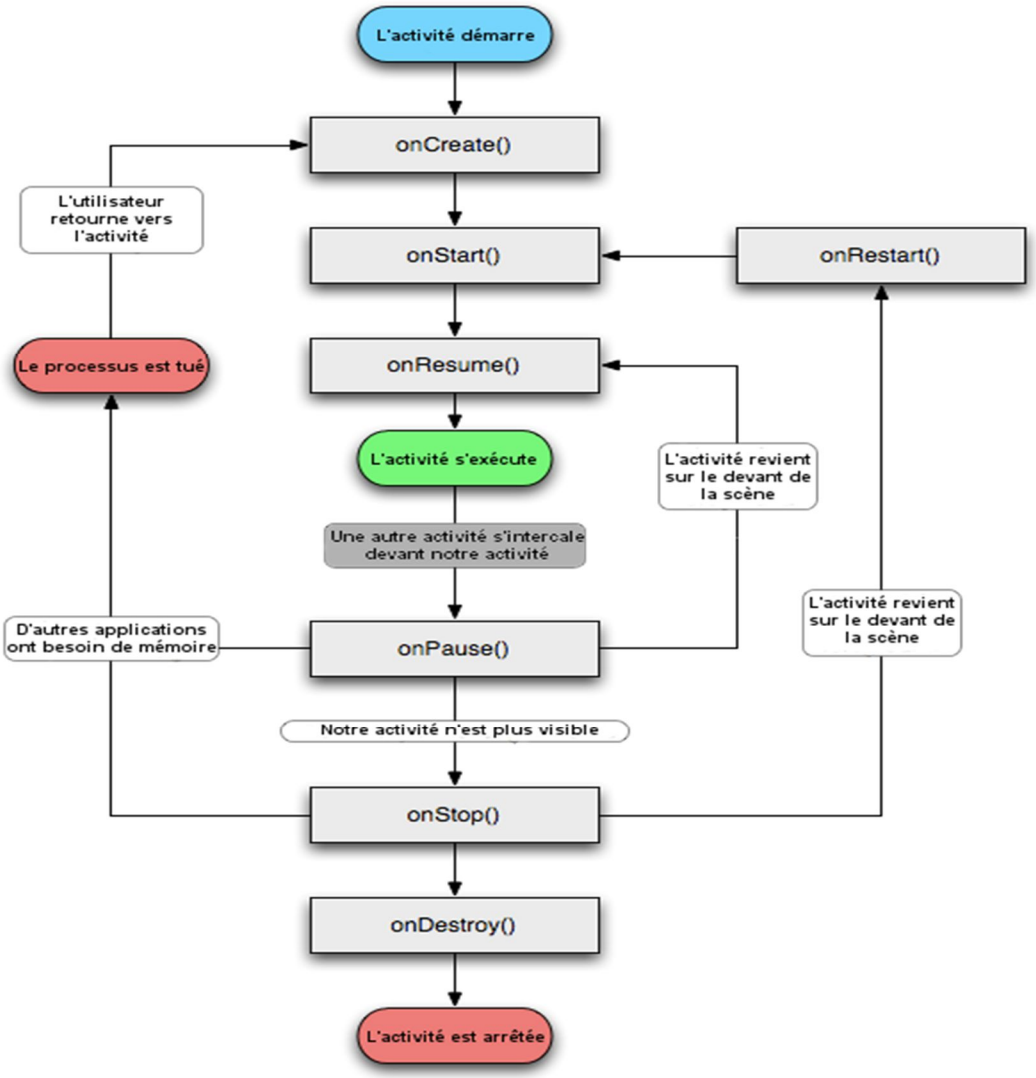

Figure II.7: *Le cycle de vie d'une Activité.*

La méthode onCreate () est appelée à la création de votre activité. Elle sert à initialiser l'activité ainsi que toutes les données nécessaires à cette dernière. Quand la méthode

OnCreate () est appelée, on lui passe un Bundle en argument. Ce Bundle contient l'état de sauvegarde enregistré lors de la dernière exécution de l'activité.

La méthode onStart () signifie le début d'exécution de l'activité (début du passage au premier plan). Si l'activité ne peut pas aller en avant plan, pour une quelconque raison, elle sera transférée à OnStop ().

La méthode onResume () est appelée lorsque l'activité commencera à interagir avec l'utilisateur juste après avoir été dans un état de pause.

La méthode onPause() est appelée au passage d'une autre activité en premier plan. L'intérêt d'un tel appel est de sauvegarder l'état de l'activité et les différents traitements effectués par

l'utilisateur. A ce stade, votre activité n'a plus accès à l'écran, vous devez arrêter de faire toute action en rapport avec l'interaction utilisateur (désabonner leslisteners).

La méthode onStop() est appelée quand l'activité n'est plus visible quel que soit la raison. Dans cette méthode vous devez arrêter tous les traitements et services exécutés par votre application.

La méthode onDestroy () est appelée quand votre application est totalement fermée (Processus terminé). Les données non sauvegardées sont perdues.

II.6 Conclusion:

Dans ce chapitre nous avons présenté les outils de conception et développement sous Android. Avec l'architecture de la plateforme, les composantes d'un projet en cours de conception ainsi que le cycle de vie associé avec les activités du projet. Notre projet vise l'interaction avec la Air Interface des réseaux mobiles, la localisation GPS et les connections sans fils du type wifi. Dans le chapitre suivant nous allons aborder les librairies Android qui permettent d'exploiter ces interactions.