Chapitre1 : Architecture et fonctionnalités de base de la plate-forme Android.

I. Introduction

Les Smartphones sont des appareils extrêmement sophistiqués, qui fournissent des fonctionnalités en plus de celles des téléphones mobiles classiques comme la navigation sur le web, la consultation et l'envoi de courriers électroniques, la messagerie vocale et visuelle, etc. Les Smartphones exécutent tous divers logiciels et applications grâce à des systèmes d'exploitation spécialement conçus pour les mobiles. Les Smartphones peuvent être personnalisés en y installant des applications additionnelles telles que des jeux ou des utilitaires grâce aux magasins d'applications en ligne (stores).

Il existe certaines contraintes pour le développement et la programmation smartphone, qui ne s'appliquent pas au développement habituel. Au moment, la mémoire RAM sur les téléphones est limitée, ce qui implique qu'on peut lancer moins de logiciels à la fois et donc les logiciels doivent faire en sorte de réserver moins de mémoire. Il est aussi important de prendre en compte que nos applications (programs) doivent pouvoir interagir avec un system complet sans l'interrompre. C'est-à-dire il faut respecter une certaine priorité des tâches, par exemple les systèmes permettent de recevoir des messages et des appels pendant l'utilisation d'une autre application. La variation de la taille des écrans, doit être considérée encore lors de la réalisation des applications pour les smartphones.

II. la plate-forme Android

Il existe plusieurs systèmes d'exploitation spécifiques aux Smartphones. Les systèmes d'exploitation les plus utilisés sont **Android**, **iOS** et **Windows Phone**.

II.1. Qu'est-ce qu'Android ?

Android est un système d'exploitation open source pour smartphones, PDA et terminaux mobiles…. Il est développé par la société mondialement connue : Google pour intégrer les applications Google : Gmail, Google Maps, Google Agenda, YouTube et la géolocalisation….. Pour diffuser en masse son système, Google a fédéré autour d'Android une trentaine de sociétés (dont Sag, Motorola, HTC, Sony Ericsson, LG, ...) à l'intérieur de l'Open Handset Alliance (OHA). Le but de l'OHA est de favoriser l'innovation sur les appareils mobiles en fournissant une plate-forme véritablement ouverte, complète et gratuite.

Le SDK Android (Software Developement Kit) fournit les outils et les API (Applications Programming Interface) nécessaires pour développer des applications sur la plateforme Android en utilisant le langage de programmation Java.

La plateforme Android propose notamment :

• un framework permettant la réutilisation et le remplacement de composants,

• une machine virtuelle optimisée pour les appareils mobiles,

• un navigateur intégré basé sur le moteur open source WebKit,

• un moteur graphique optimisé, propulsé par une librairie 2D dédiée et un moteur 3D basé sur les spécifications OpenGL ES 1.0,

• le système de gestion de base de données SQLite pour le stockage de données,

• un support média pour les principaux formats audio, vidéo et images,

• la téléphonie GSM, les communications Bluetooth, 3G et WiFi,

• un accès à la caméra, au GPS, à la boussole et aux accéléromètres,

• un environnement de développement riche : émulateur, outils de débogage, …

L'Android Market, renommé Google Play Store en mars 2012, permet le téléchargement d'applications gratuites ou payantes. Il est aussi possible de les noter et de les commenter. Fin 2010, il y avait déjà plus de 100 000 applications sur l'Android Market, dont seulement 35,6 % payantes ; en septembre 2011, on dénombrait sur Google Play Store quelques 520 000 applications dont toujours environ 65 % gratuites... La progression du nombre d'applications sur Google Play Store est exponentielle ! Cette progression s'explique par le développement totalement ouvert d'Android, et les applications peuvent d'ailleurs être distribuées autrement que par ce biais. Pour simplifier le développement d'applications, Google a développé une interface web : App Inventor permettant de développer facilement une application qui pourra ensuite être mise à disposition sur le marché.

II.2. Origine

Android, qui se prononce Androïd, doit son nom à la startup du même nom (spécialisée dans le développement d'applications mobiles), rachetée par Google en août 2005. Nom qui vient lui-même d'« androïde » qui désigne un robot construit à l'image d'un être humain...Fondé sur le noyau Linux, le système d'exploitation Android de Google a creusé l'écart avec Apple dans le secteur des smartphones. Samsung reste actuellement le leader incontesté des ventes d'Android, avec 73,3 millions de smartphones vendus en trois mois et 39,1% de parts de marché.

II.3. Les versions

Les différentes versions ont des noms de dessert (qui suivent l'ordre alphabétique, de A à Z) qui sont sculptés et affichés devant le siège social de Google (Mountain View). D'où vient le nom de dessert apposé sur chaque version d'Android ? Au départ, il s'agit d'un petit délire dans l'équipe en charge du projet. L'idée c'est que les desserts, comme les smartphones et tablettes, sont là pour rendre nos vies plus agréables !!!!!

•1.0 : Version connue des développeurs : sortie avant le premier téléphone Android (fin 2007).

•1.1 : Version incluse dans le premier téléphone, le HTC Dream

• 1.5 : Cupcake (Petit Gâteau - Avril 2009), API level 3

• 1.6 : Donut (Beignet - Septembre 2009), API level 4

• 2.1 : Eclair (Eclair - Janvier 2010), API level 7

• 2.2 : FroYo (Frozen Yogourt / Yaourt glacé - Mai 2010), API level 8

• 2.3 : Gingerbread (Pain d'épice - Décembre 2010), API level 9

• 3.0 : Honeycomb (Gâteau de Miel – Février 2011), API level 11

◦ 3.1 : Mars 2011, API level 12

◦ 3.2 : Juillet 2011, API level 13

• 4.0 : Ice Cream Sandwich (Sandwich à la crème glacée - Octobre 2011), API level 14

◦ 4.0.3 : Décembre 2011, API level 15

• 4.1 : Jelly Bean (Bonbon à la gelée / Dragée – Juillet 2012), API level 16

◦ 4.2 : Octobre 2012, API level 17

◦ 4.3 : Juillet 2013, API level 18

• 4 .4 : Kitkat, octobre 2013.

• 5.0 : Lollipop, novembre 2014.

• 6.0 : Marshmallow en fin 2015, Nougat, Oreo Puis ANDROID 9 Pie,ANDROID10,ANDROID11.

III. Architecture Android

Architecture en "pile logicielle" :

- **La couche "Applications" :** Android est utilisé dans un ensemble contenant déjà des applications natives comme, un client de mail, des programmes pour envoyer des SMS, d'agenda, de navigateur web, de contacts personnels…..

- **La couche "Application Framework" :** cette couche permet au programmeur de construire de nouvelles applications. Cette couche fournit la gestion :
    - des Views (= IHM).
    - des ContentProviders = l'accessibillité aux données des autres applications (ex: les contacts) et donc les partages de données.
    - des ressources = les fichiers non codes comme les images, les écrans (Resource Manager).
    - des Notifications (affichage d'alerte dans la barre de titre).
    - des Activitys = l'enchaînement des écrans.

- **La couche "Libraries" (bibliothèques) :** couche logicielle basse pour utiliser :
    - les formats multimédia : images, audio et vidéo.
    - les dessins 2D et 3D, bitmap et vectoriel.
    - une base de données SQL (SQLite).

- **L'environnement d'exécution (Android Runtime) :** toute application est exécutée dans son propre processus, dans sa propre Dalvik Virtual Machine. DVM est la machine virtuelle Java pour les applications Android, conçu pour exécuter du code Java pour des systèmes ayant des contraintes de place mémoire et rapidité d'exécution. Elle exécute du code .dex (Dalvik executable) qui sont des .class adaptés à l'environnement Android. Ecrit par Dan Bornstein d'où le nom est un village islandais dont sont originaires certains de ses ancêtres. Elle a été choisie par Google car plusieurs instances de la DVM peuvent être lancées efficacement. Le code de la DVM est open source.

- **Le noyau Linux :** sur lequel la Dalvik virtual machine s'appuie pour gérer le multithreading, la mémoire. Le noyau Linux apporte les services de sécurité, la gestion des processus, etc.

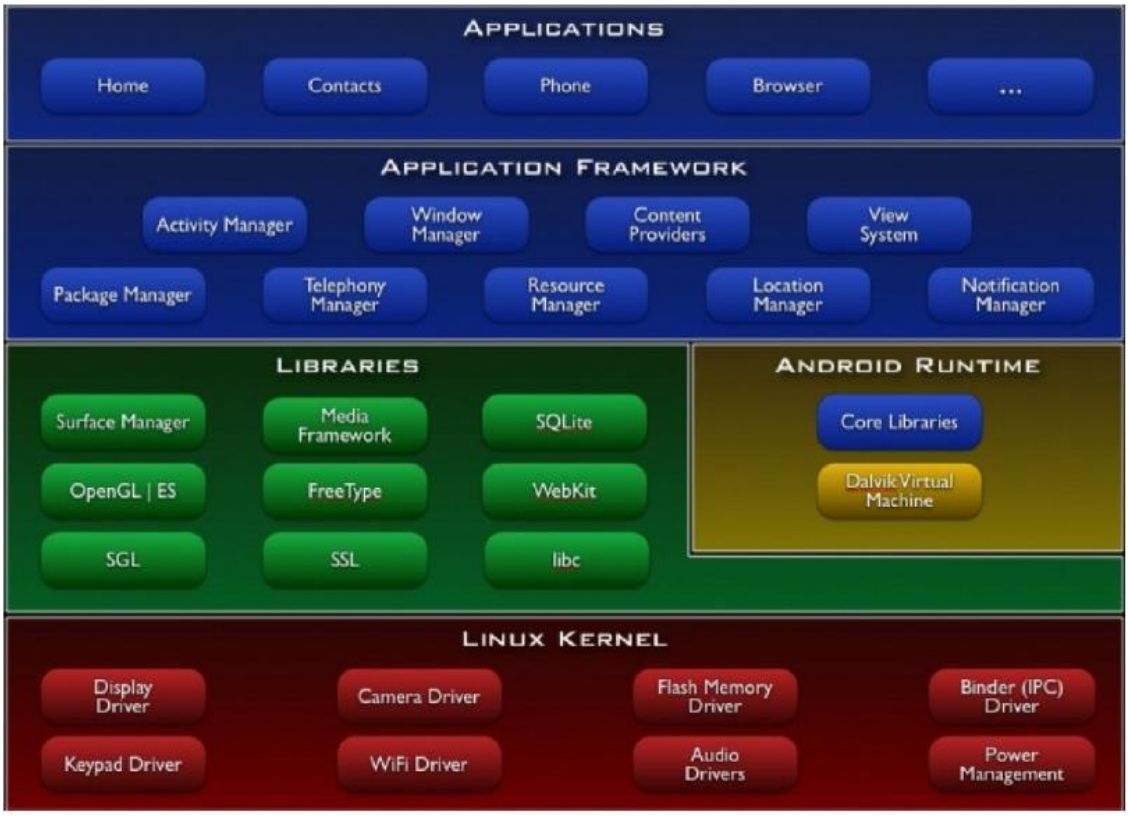

Figure1. Architecture en "pile logicielle".

Chapitre2 : Développement Android.

I. Les composants d'une application Android

Les composants fondamentaux d'une application Android sont : Activity, Service, Content Provider, Broadcast Receiver. Certains de ces composants communiquent entre eux à l'aide d'Intent :

- **Activity** (activité)"gère l'affichage et l'interaction d'un écran" : La plupart des applications se compose de plusieurs écrans. Chaque écran peut être réalisé par une **activité**. Si un nouvel écran s'ouvre, le système utilise une pile d'histoire pour stocker les écrans précédents et pouvoir reprendre l'état précédent ou enlever cet état.
- **Service** : Un Service est utilisé pour réaliser l'application en arrière-plan. C'est-à-dire, cette application peut marcher quand une autre application est en train de s'exécuter comme les services de lecture de musique.
- **Content Provider (**fournisseur de contenu): Gère des données partageables. C'est le seul moyen d'accéder à des données partagées entre applications. Exemple de fournisseur de contenu : les informations de contact de l'utilisateur du smartphone.
- **Broadcast Receiver** (récepteur d'informations) : est un composant à l'écoute d'informations qui lui sont destinées. Un tel récepteur indique le type d'informations qui l'intéressent et pour lesquelles il se mettra en écoute. Exemple : appel téléphonique entrant, réseau Wi-Fi connecté, informations diffusées par des applications.

  Un récepteur n'est pas une IHM mais peut en lancer une (éventuellement petite : une barre de notification), ou peut lancer un service traitant l'arrivée de l'information.

- Un événement (**intent)** est une "intention" à faire quelque chose contenant des informations destinées à un autre composant Android. C'est un message asynchrone. Les activités, services et récepteurs d'informations utilisent les Intents pour communiquer entre eux.

II. Cycle de vie d'une activité (Activity Life cycle) :

Pour développer une application sur Android, on doit comprendre le cycle de vie d'une activité. Le cycle de vie d'une activité est exprimé par la figure suivante :

- L'état **Active**/courant (Running) : L'activité marche en avant-plan.
- L'état **Paused** (en pause) : Cette activité est visible mais elle n'est pas active.
- L'état **Stopped**: Cette activité n'est pas visible. Si une activité est complètement masquée par une autre activité, elle est arrêtée et conserve tous les états. Cependant elle n'est plus visible pour l'utilisateur, sa fenêtre est cachée et elle sera souvent tuée par le système lorsque la mémoire est nécessaire ailleurs.
- L'état **Dead :** Cette activité est terminée ou elle n'a jamais été démarrée. Si une activité est en pause ou arrêtée, le système peut supprimer 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.

- Il existe trois boucles principales :

- La durée de vie d'une activité se passe entre le premier appel à OnCreate () et l'appel à onDestroy (). Une activité met en place tous les états globaux dans la méthode onCreate () et libère toutes les ressources restantes à onDestroy ().
- La durée de vie visible d'une activité se passe entre un appel à onStart () et un appel correspondant à onStop (). Dans ce temps, l'utilisateur peut voir l'activité sur l'écran, même si elle n'est pas à l'avant et à l'interaction avec l'utilisateur. Entre ces deux méthodes, les ressources qui sont nécessaires pour montrer l'activité de l'utilisateur sont conservées.
- La durée de vie d'une activité en avant-plan se passe entre un appel à onResume () et un appel correspondant à onPause (). Dans ce temps, l'activité est en face de toutes les autres activités afin d'interagir avec l'utilisateur. Une activité peut souvent changer son état entre l'état de reprise et l'état en pause.

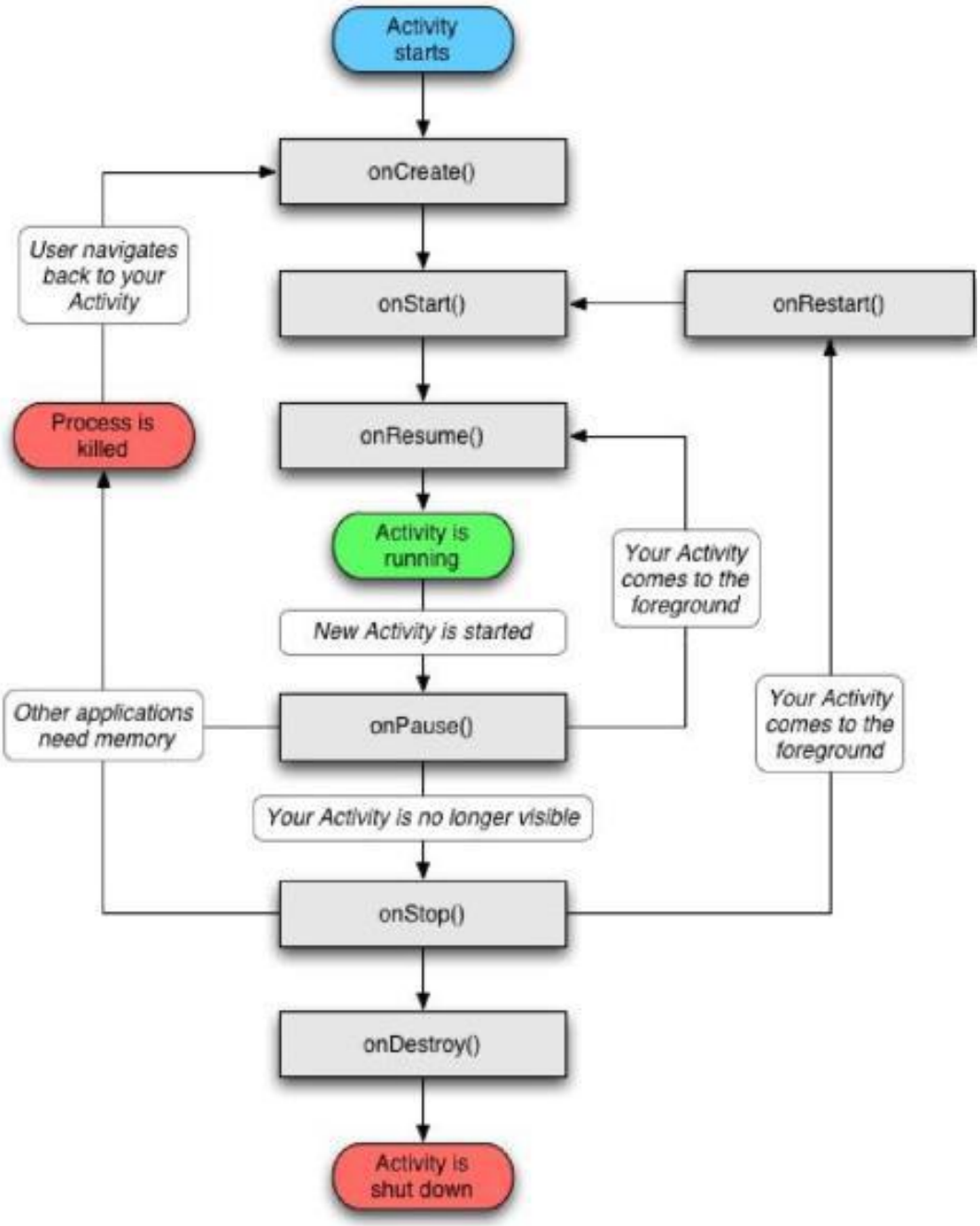

Figure1. Cycle de vie d'une activité.