# Chapitre 1 : Architecture d'Android

Mousaab Bada m.bada@univ-batna2.dz
http://staff.univ-batna2.dz/bada_mosaab

24/01/2018 (rev. 1)

**Objectifs :**

- Comprendre l'architecture du système Android.
- Comprendre l'organisation d'une application Android .

# Partie 1 :

## 1. Introduction « systèmes embarqués »:

Un système embarqué est défini comme un système électronique et informatique autonome, souvent temps réel, spécialisé dans une tache bien précise. De tels systèmes sont nombreux dans des secteurs aussi variés que l'aéronautique, l'électroménager, le matériel médical, la téléphonie mobile, etc.

Les systèmes embarqués les plus basiques ne disposent pas d'une interface utilisateur sophistiquée « simples boutons ou Led ».

D'autres peuvent présenter un écran tactile ou un « joystick » qui permet de naviguer sur l'écran. D'autres enfin, sont connectés au réseau.

Les ressources d'un système embarqué sont généralement limitées. Cette limitation est généralement d'ordre spatial (encombrement réduit) et énergétique (consommation restreinte).

### 1.1. Contraintes Matériel et logiciel:

Les cahiers des charges des systèmes embarqués comportent plusieurs contraintes. On peut citer :

- Une puissance processeur faible;
- Une RAM limitée;
- Plusieurs type de résolutions de l'écran;
- Des coûts élevés de transferts de données;
- Des connexions réseau moins stables ;
- Logiciel :
  - Un système d'exploitation léger « Java Card, IOS, Android, Tiny OS, windows embarqué, ... »;
  - Optimisation du code afin qu'il soit rapide et réactif.

## 2. Introduction à l'OS embarqué « Android » :

Android est **un système d'exploitation mobile**. Tout comme Windows ou Linux ➔ il est composé de petits modules programmes, qui permettent de gérer le matériel et d'exécuter d'autres logiciels.

Le système d'exploitation Android est actuellement l'OS le plus utilisé dans le monde faisant tourner des **smartphones**, **tablettes**, **montres connectées**, **télévisions interactives**, et bien d'autres.

| Plateforme | Programmation | IDE recommandé(s) |
|---|---|---|
| *Windows Phone* | VB.Net, C# | Visual studio .Net |
| *iOS* | Objective-C, Swift | X-CODE |
| *Blackberry OS* | Java | MDS Studio |
| *Java ME* | Java | EclipseME (CLDC, MIDP) |
| *Android* | Java, code natif C++ | Android Studio / Eclipse + plug-in ADT |
| *Palm WebOS* | JavaScript, C/C++ | Eclipse + webOS plug-in |
| *Symbian OS* | C++ | Performance |
| *Brew MP* | C++ | Visual Studio + plug-in |
| *Bada* | C++ | badaIDE |
| *Sailfish OS (MeeGo)* | Qt C++ | QtCreator |
| *Firefox OS (B2G)* | HTML5/CSS3/JavaScript | Notepad++ et Firefox OS Simulator |
| *Ubuntu Mobile* | C/C++, JavaScript | Qt Creator |
| *Tizen* | HTML5/JavaScript, code natif C++ | Eclipse + plug-in Tizen |

ou

**Figure1 : Systèmes d'exploitation Mobile**

C'est un système, **open source** qui utilise le **noyau Linux**. Il a été créé par Android, Inc. qui fut rachetée par Google en 2005.

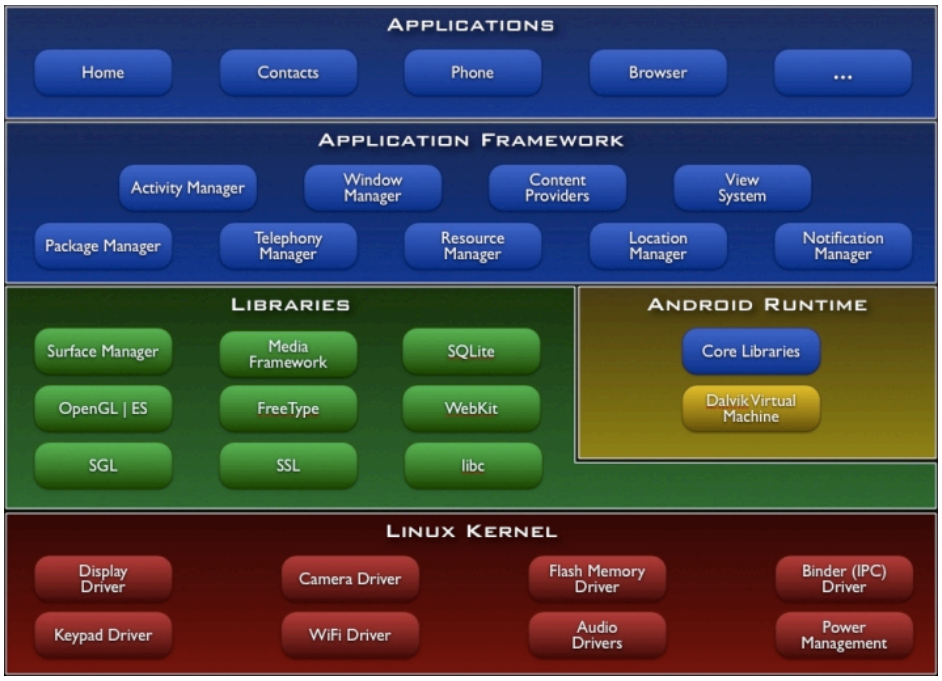
Le développement d'applications pour Android s'effectue généralement en **Java** ou **Kotlin** en utilisant des bibliothèques spécifiques.

## 2.1. Architecture :

Après quelques réflexions, je me suis dit que *Ce n'est peut-être pas* le «*moment idéal*» pour présenter l'architecture complète de A jusqu'à Z.

A ce stade, nous nous contentons de présenter abstraitement l'ensemble des couches ainsi que la partie « Applications ». Nous reviendrons sur le détaille de l'architecture à la fin de ce chapitre.

D'habitude la lecture de cette architecture se fait d'une manière ascendante (de bas en haut). Nous par contre nous allons la faire d'une manière descendante, commençant par la partie la plus proche des utilisateurs afin de simplifier la perception.




## A. La couche Applications :

Alors c'est la couche **Application**☺. C'est un ensemble des programmes écrits généralement en **java** *« avant l'apparition du langage **Kotlin** en 2017 qui est devenu officiellement le langage de programmation voulu et recommandé par le géant américain Google* ».

Elle sont généralement distribuées depuis des plate formes de téléchargement telles que **l'AppStore** (plateforme d'Apple), le **GooglePlay** (plateforme de Google / Android), ou encore le **Windows Phone Store** (plateforme de Microsoft).

Une application Android peut être composée des éléments suivants:

- **Activité « Activity »** : une interface graphique « une vue » (équivalent à une fenêtre ou boite de dialogue en java).
- **Service** : un programme qui s'exécute en arrière plan.
- **Fournisseurs de contenus « Provider »** : un mécanisme (une Class) permettant le **partage** d'informations entre applications.
- **Widgets** : une vue accrochée au bureau d'Android.
- **Notifications** : permet de notifier l'utilisateur de la survenue d'événements.
- **Intention « Intent »** : un mécanisme (une Class) peut être utilisé pour transiter des données entre deux activités.
- **Récepteurs d'Intents « BoradcastReceiver ».**

Jusqu'ici tout va bien !!! Reprenez votre souffle, car cette fois nous allons plonger dans des concepts un peu plus avancés.

**NBs :**

Une **« Activity/ Fenêtre/ Vue»** qui est **visible** à l'utilisateur se trouve dans un état d'« ***Exécution*** ».

Quand-t-on clique sur le bouton Home beaucoup de choses vont se produire :

- l'Activity passe à l'état « ***En Stop***».

  Ça veut dire quoi en Stop ?

  Ça veut dire qu'on va stocker les valeurs des différentes variables dans un espace mémoire bien déterminé (c'est l'opération du sauvegarde du contexte vu dans le cours SE).

  Alors, Pourquoi on aura besoin de sauvegarde le contexte ?

  hmm... c'est pour revenir à l'état où on était avant de cliquer sur le bouton Home (restauration du contexte).

- Ah bon !!! Tout se passe derrière nous ☹. Nous ne voyons rien de tous ca.

  Ce que nous voyons c'est juste l'interface de l'activity qui est disparu, et une autre qui vient de la remplacer. ➔ Un changement de dessin et de pixel, c'est tout!

**Mais qui est le responsable de tout ça ?**
**C'est la deuxième couche : Application FrameWork**

## B. La couche Application FrameWork :

Puisque nous n'avons pas encore commencé la partie programmation et pour simplifier un peu les choses, j'aimerai bien utiliser le nom « couche des gestionnaires » au lieu de « Application FrameWork ».

Pourquoi utilisons-nous un tel nom?

Tout simplement parce que cette couche représente ensemble **d'outils (programme système au niveau du smart phone, outil de développement au niveau du PC, ensemble de class Java, etc... )** qui se limitent à la **gestion** des différentes parties de votre application (activity, ressource, content, package, Telephony,...) de la couche application.

Pour ne pas compliquer les choses, nous allons donner un exemple pour chaque type de gestionnaire :

### a) Acitivty manager vs Window Manager :

Pour comprendre le sens du nom gestion, Prenons l'exemple de **Window Manager** et **Activity Manager :**

- Le gestionnaire des fenêtres : est un ***« programme système »*** responsable de l'organisation de l'écran. Il permet d'allouer la surface, et décider où les Applications vont être placé et superposés.

  Il est aussi le responsable de la partie affichage/modification des pixels (traçage de l'interface).

- Le gestionnaire d'activités : est un **« *programme système* »** responsable de la gestion des différentes états d'une activité : (démarrage, exécution, pause, stop, destruction).

  Ex : sauvegarde/restauration du contexte

### b) Gestionnaire de téléphone:

Cette fois **la gestion** signifier qu' : il existe **une class Java** développée par la société Android qui s'appel ***TelephonyManager***.

Cette class contient des méthodes qui permettent de gérer les appels téléphonique :

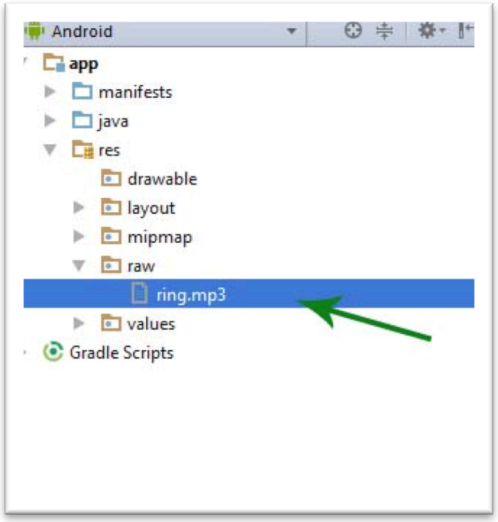
- récupérer l'état de l'appel : sonné, occupé, raccroché, etc...
- récupérer le numéro de série de la carte SIM.
- Récupérer la puissance du signal.
- Etc...

### c) Gestionnaire des ressources :

- Q1 : Tout d'abord il faut définir ce que signifier une ressource ?
  - R1 : N'importe quelle application utilise des **Fichiers** externe tel que : Image, animation, son, vidéo, xml, etc ...

    Nous appelons ces fichiers : **Ressources**.

En Android, il faut **organiser (placer)** les **Ressources** de votre projet dans un ensemble de dossiers et sous dossiers bien déterminée.




Apres avoir placer et organiser les fichiers, il faut donner un **identificateur** (id) pour chaque ressource (fichier).

- Q2 : Ou sont stocker les identificateur ?
  - R2 : pour chaque type de ressource il existe une **sous class java** (ex : ***R.drawable, R.raw, R.layout,*** **etc**.... ).

    Dans chacune des sous classes, il existe des **variables static** de type **entier** qui permettent de stocker l'identificateur d'une ressource.

    Ex : R.raw.ring ( **ring** c'est une variable de type entier qui se trouve dans la sous class **R.raw** qui représente le dossier physique (res/raw). Ce dossier permet de stocker les fichiers multimédia (MP3, etc....) .

- Q3 : c'est qui le responsable de cette gestion/organisation des ressources ? Qui va générer ces identificateurs ?
  - R3 : c'est le gestionnaire des ressources : **l'outil aapt** (aapt tool)

## C. La couche d'exécution Android + Library:

Je vous avais déjà dit qu'une application Android est écrite en Java. Cela signifier qu'on aura besoin d'un un moteur d'exécution pour android (JRE pour android = DALVIK).

Rappels :

Q1 : quelle est la différence entre un JRE et un JDK ?

R1 :

- **JDK= JRE + outils de développement**
- Outils de développement = (Compilateur « xxx.java➔xxx.class », debugger, ...)

Q2 : alors qu'est ce qu'un JRE ?

R2 :

- Un JRE est un moteur d'exécution (« *runtime system* » en anglais) est *un programme qui permet l'exécution d'autres programmes*.
- **JRE = JVM + Bibliothèque « API »**
- JVM= est ensemble d'outils qui permettent d'exécuter un programme java (ensemble de fichier xxx.class) dans une machine ou un smart phone.
- On trouve dans la JVM:
  - le chargeur : Class Loader.
  - le vérificateur des fichiers class (pour vérifier si le fichier xxx.class n'a pas été modifié).
  - récupérateur d'espace mémoire : Garbage collector.
  - Interpréteur

**Vous vous rappeler maintenant ? Très bien ☺**

Mais attend !!! Nous avons dit aussi qu'Android est à la base de linux. Et par conséquence il est à la base du langage C.

- Q1: Comment ca ? Le langage C pour le système et le Java pour la programmation ? comment faire le lien entre les deux ? Et pour la partie librairies, elle est écrite en Java ou en C ?

- R1 : Trop de questions ! Mais ne vous inquiétez pas☺. Le grand secret se trouve dans la partie **Librairies** et les **API**.
  - Les bibliothèques sont écrite en C/C++. Il existe deux types de bibliothèque :
    - Bibliothèque système : offrent un accès bas niveau
    - Bibliothèques de manipulation des medias, du WebKit, SQLite, etc...
  - Les bibliothèques sont exposées aux développeurs par le biais des **API**.

- Q2 : Qu'est ce qu'un API ?
- R2 : Est ce que vous vous souvenez de la notion du Lien hypertexte ? un lien c'est un simple texte qui vous amène vers une autre page, un fichier, une image, etc...

La cible d'un lien hypertexte est **créé par une autre personne**, et peut être un fichier Word, PDF ou même un fichier MP3. Donc **la cible n'est pas toujours une page web.**

C'est la même chose avec les APIs. **Un API est une sorte de lien** qui vous **permet d'accéder** à un **programme** écrit par une autre personne avec un le même langage ou **un langage différent.**

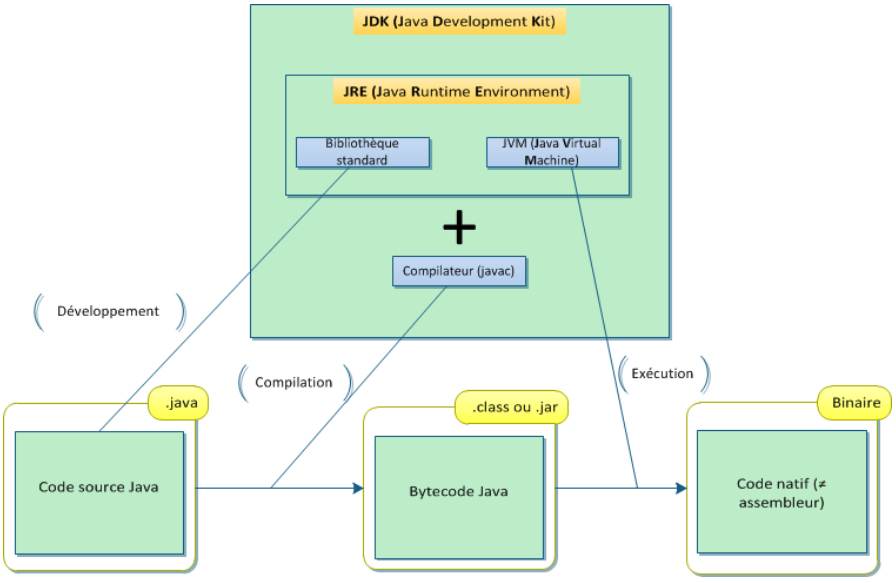
API (Application Program Interface) : si on se concentre sur la lettre « I : interface ». L'interface : signifie l'entête d'une méthode, qui vous aide à connaitre les paramètres d'entrés/sorties d'une méthode.

Ex : double Add (double,double)

Donc c'est a travers cette interface « I » (lien) qu'on peut écrire notre code Java qui permet d'accéder à une routine (méthode) écrite en C/C++.

Maintenant, l'ensemble « API » signifie un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels

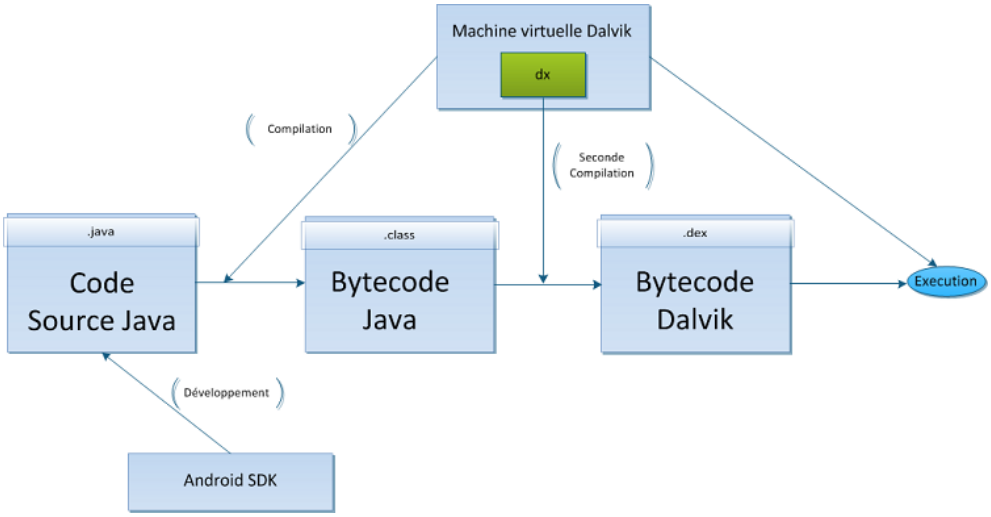
Source : wikipedia.



- Q1 : Est ce qu'on utilise « Java SE » pour le développement des applications Android ?
  - R1 : Non. La version de Java qui permet le développement Android est une version réduite (pas de swing, etc...)

- Q2 : Et pour le JRE ?
  - R2 : il n'existe pas vraiment un JRE en Android.

- Q3 : Wooow !!!! Et comment ca fonctionne ? Je crois que nous avons développé notre application en Java.
  - R3 : Il existe ***une nouvelle machine virtuel*** pour les systèmes embarqués qui s'appel **DALVIK.**
  - Cette machine virtuelle est optimisée pour mieux gérer les ressources physiques du système.
    Elle permet par exemple d'utiliser **moins d'espace mémoire** ou d'utiliser **moins de batterie** qu'une machine virtuelle Java.
  - Cette machine virtuelle ne peut pas exécuter un bytecode !!!! il faut donc convertir le bytecode (xxx.class) en un fichier (xxx.dex) pour l'exécuter par la machine virtuelle DALVIK




## D. La couche noyaux linux:

Le noyau est l'élément du système d'exploitation qui permet d'exploiter le matériel de votre smart phone.

- Par exemple, le driver WiFi permet de contrôler la l'antenne WiFi de votre smart phone.
- Quand Android veut activer le WiFi, on peut imaginer qu'il utilise la fonction « ActiverWifi() »
- Cette fonction est implémentée par le constructeur de la carte wifi.

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