INFO0306 - Programmation mobile

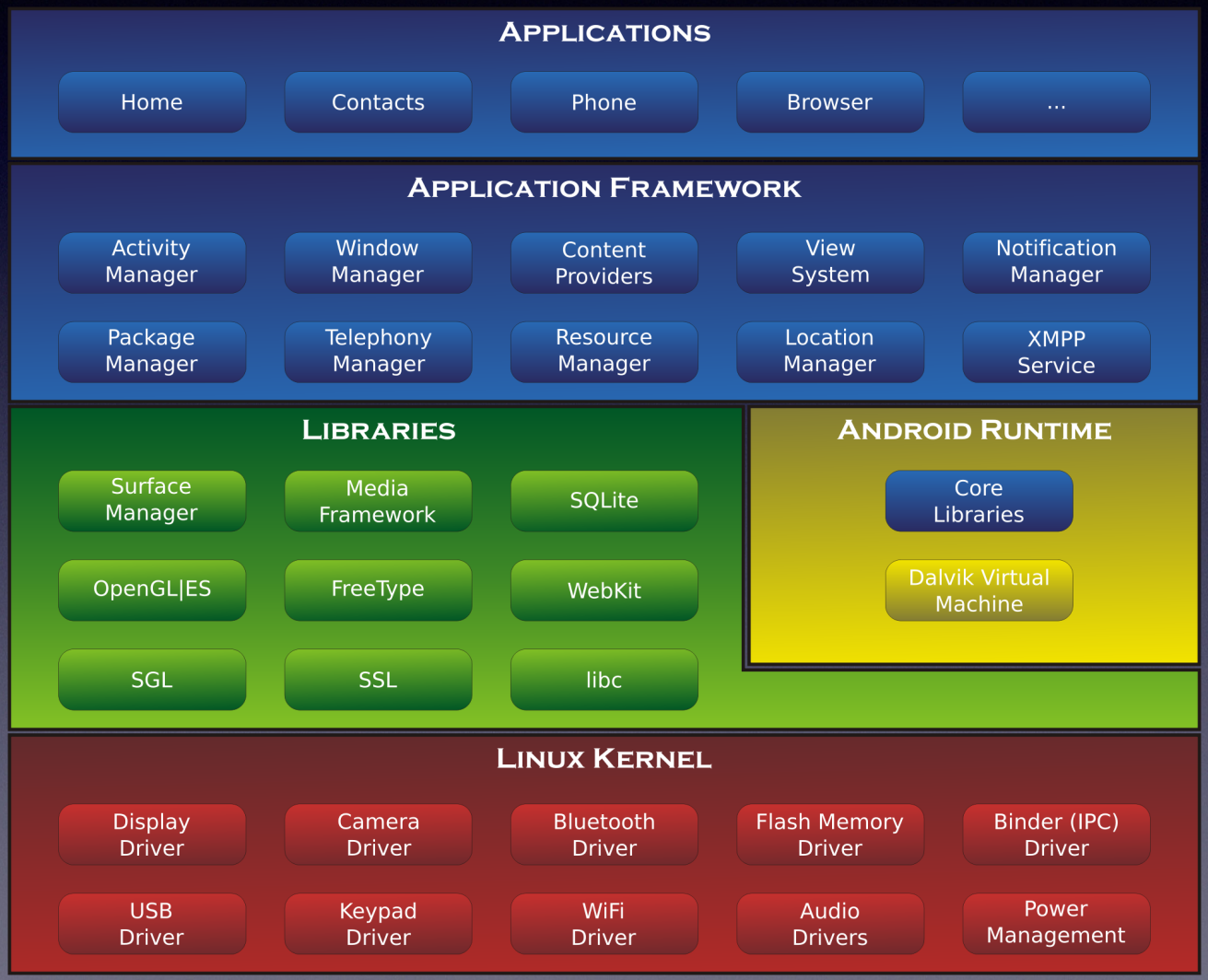
sommaire

- Rappels
- Architecture d'Android

Rappels

- Appareils mobiles de plus en plus nombreux
- Beaucoup sous Android
- Beaucoup d'avantages pour les mobiles
- Avec quelques un à prendre en compte
- Beaucoup d'application smartphones disponibles
- Android système ouvert très utilisé basé sur Linux
- Beaucoup de versions avec des évolutions rapides

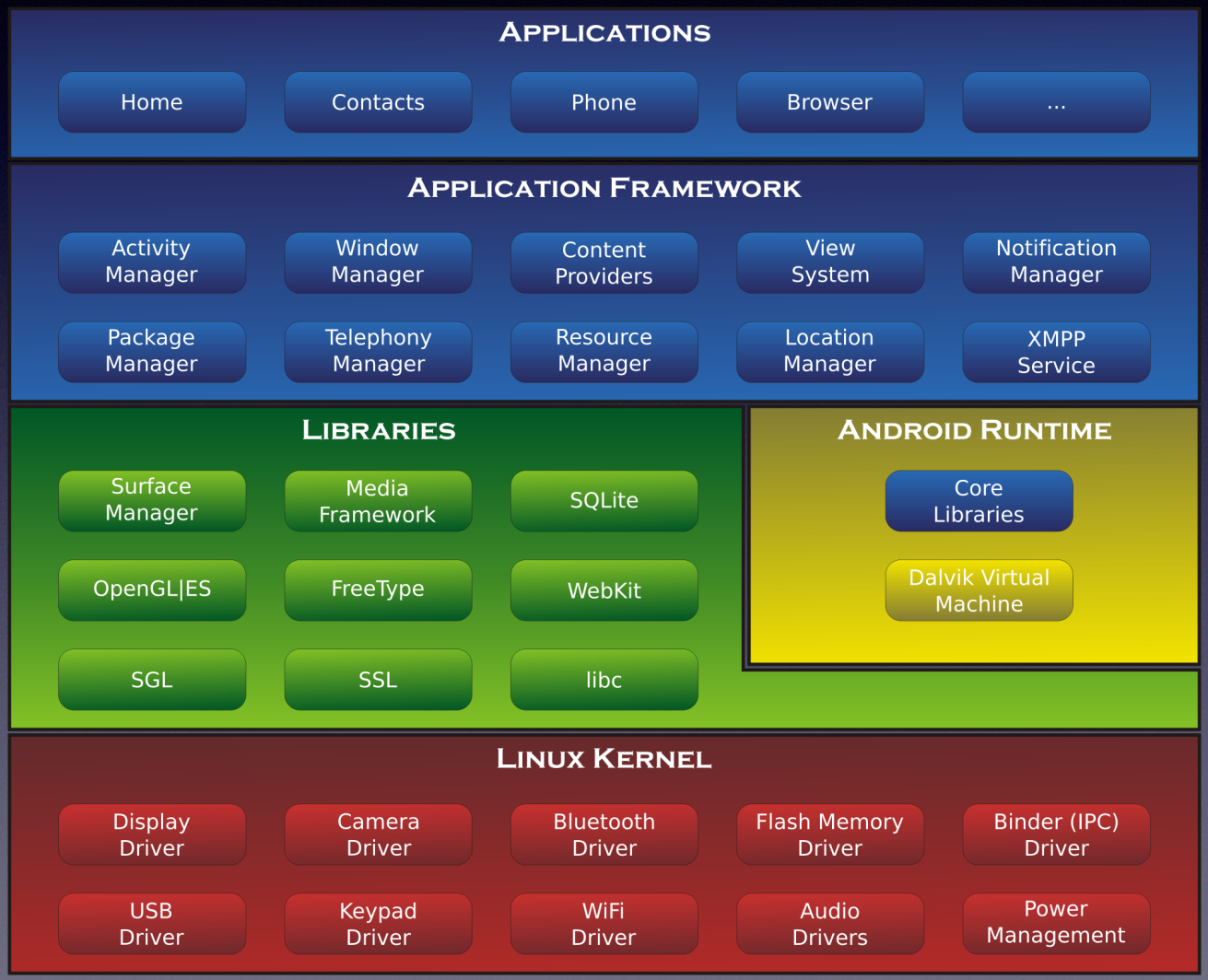
Architecture d'Android



Les couches du système Android

- 5 couches :
  - Noyau
  - Bibliothèques natives
  - Runtime
  - Framework
  - Application

Couche 1 : Noyau



Couche 1 : Noyau

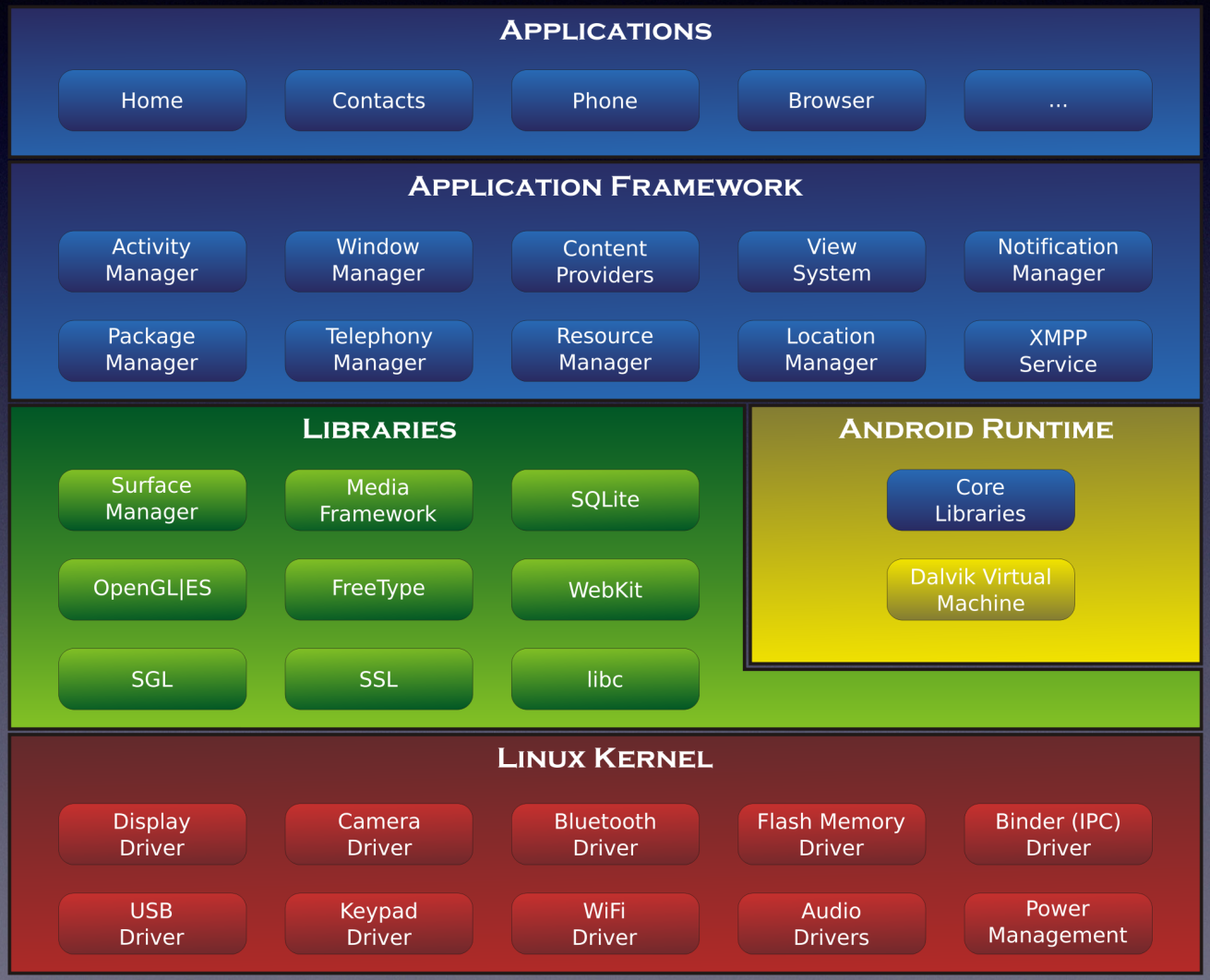
- Basé sur Linux mais modifié pour :
  - sa stabilité
  - sa maturité
  - l'ouverture de son code
- Principal Changement : Suppression des IPC SysV remplacé par Binder car économique en ressources
- Gestion de la mémoire différente (SHM POSIX mais simplifiée)
- Partage de mémoire entre processus via Binder
- Intégration d'un logger car système embarqué oblige…

Couche 1 : Noyau

Couche 1 : Noyau

- Gestion de la sécurité
- Gestion de la mémoire
- Gestion des processus
- Gestion réseau
- Drivers, etc…
- Ce noyau agit comme une couche d'abstraction entre le matériel et le restes des couches applicative.
- Compatibilité Linux?
  - Pas de système X-Window nativement
  - Ne supporte pas toutes les libraires GNU standards
  - Difficile de porter toutes les applications/librairies compatibles Linux
  - —> Le Code de google n'est pas reversé dans le noyau Linux car Android forme un nouvel arbre de développement

Couche 2 : Bibliothèques Natives



Couche 2 : Bibliothèques Natives

- Elles fournissent un accès direct aux ressources du système
- Une couche d'abstraction au framework Java Android

Couche 2 : Bibliothèques Natives

Couche 2 : Bibliothèques Natives

- Android inclus un ensemble de librairies C/C++
- Utilisées par les applications Android
- Accessibles au développeurs via le SDK
- Quelques unes de ces librairies :
  - Librairie Système C : une implémentation dérivée de l'implémentation BSC des librairies standard C (libc)
  - LibWebCore : Un moteur de navigateur internet moderne utilisé autant pour navigateur android que pour les vues web intégrables
  - SQLite : un système de gestion de base de données relationnel léger et puissant disponible pour toutes les applications.

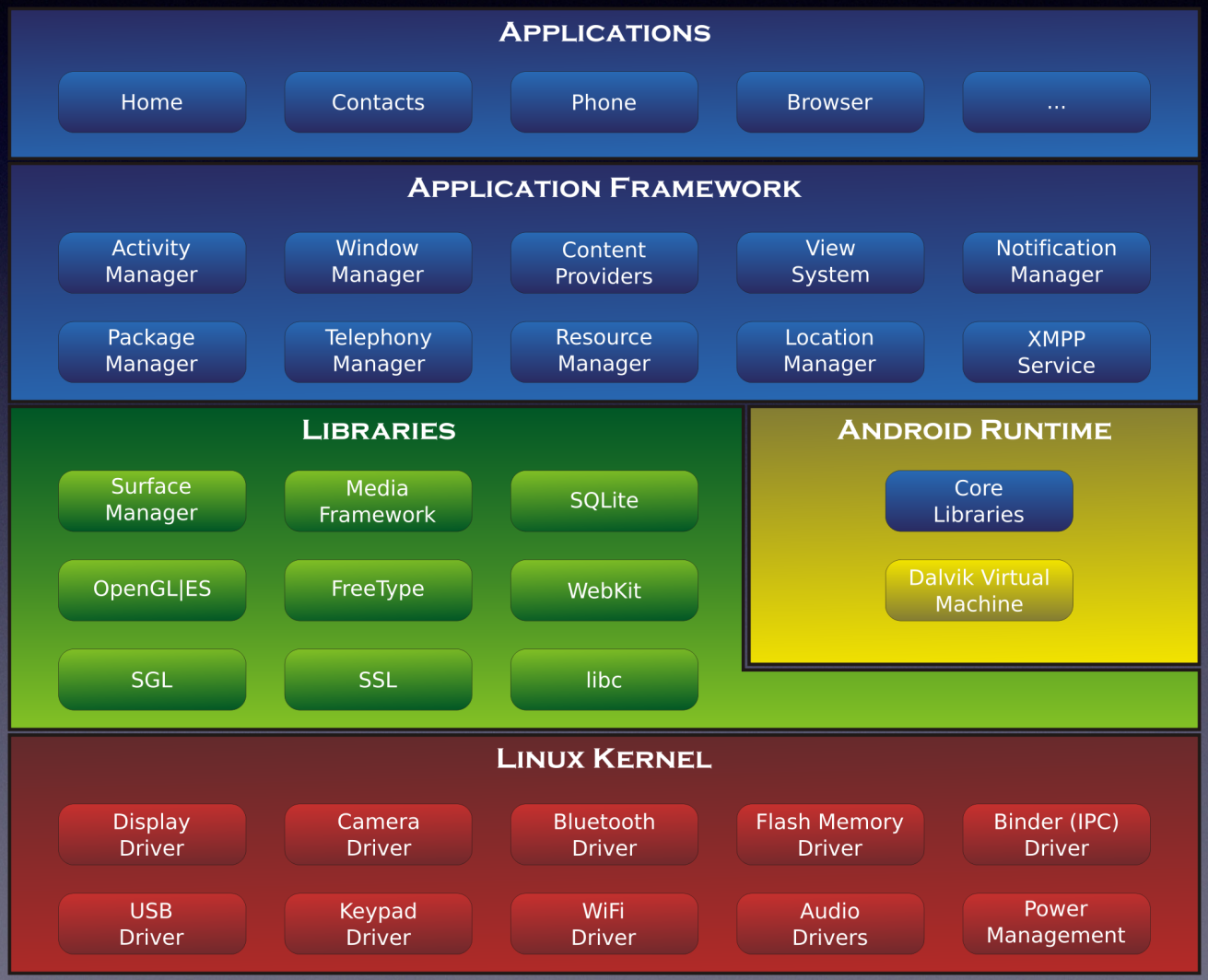
Couche 2 : Bibliothèques Natives

- Bibliothèques connues :
  - Bionic Libc :
    - Elle ne repose pas sur la classique GNU Libc.
    - Sa propre bibliothèque C appelée Bionic Libc.
    - Pas l'ensemble des fonctions POSIX.
    - Bionic Libc ne prend en charge que les architectures ARM et x86.
    - Bon support ARM et non Power PC ou MIPS
    - Les threads sont incompatibles avec POSIX
  - WebKit : intégrer un moteur de rendu de pages Web
  - FreeType : intégrer un moteur de rendu de police de caractères
- Bibliothèques développées par Google :
  - Le media framework : codec, compression, lecture, écriture.
  - Surface Manager : Dessiner à l'écran via l'interface

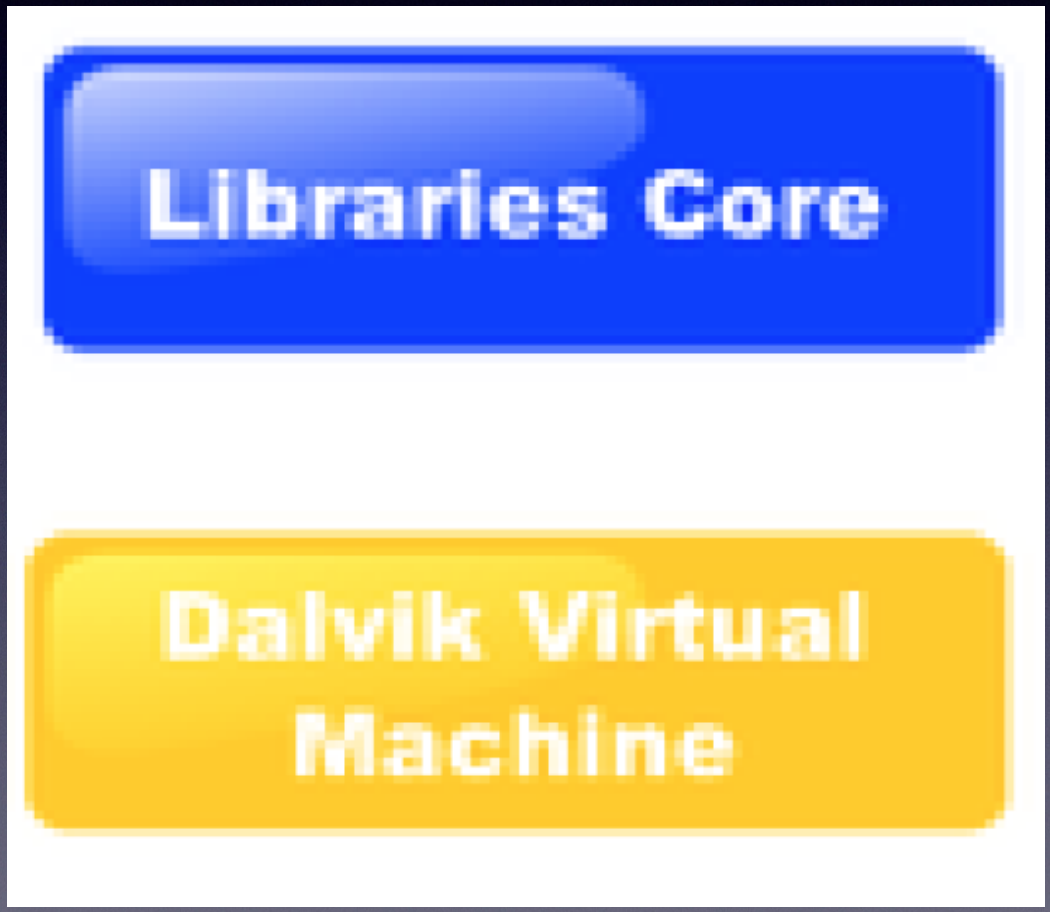
Couche 2 : Bibliothèques Natives

- Autres librairies :
  - Librairies MultiMedia : Intègrent le support de la lecture et de l'enregistrement de nombreux formats audio, vidéo et image (MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG...)
  - Surface Manager : Gère l'accès et l'affichage des différentes vues (2D ou 3D) composant les applications
  - SGL : Le moteur de rendu pour l'imagerie 2D
  - Librairie 3D : Une implémentation basée sur l'API OpenGL ES 1.0, et intégrant à la fois l'accélération matérielle (si disponible) et l'accélération logicielle.

Couche 3 : Android Runtime



Couche 3 : Android Runtime



Couche 3 : Android Runtime

- **DVM : Dalvik Virtual Machine**
  - Ecrite par Dan Bornstein
  - Dalvik : village de pêcheurs en Islande
  - Une sorte de JVM optimisée pour les systèmes limités en mémoire et en puissance.
  - Exécute les applications ".dex" compilés depuis le code automatiquement par le SDK avec l'outil "dx"
  - Utilise du ByteCode spécifique et non du ByteCode Java
  - Optimisée également pour être "multi-instance" sur un seul terminal.

Couche 3 : Android Runtime

- Anecdote : Oracle Vs. Google
  - Oracle (Java) porte plainte en Aout 2010 envers Google pour leur implémentation de Dalvik qui serait basée sur le code source de java
  - Non respect de 7 brevets API / copie de 9 sur 15 millions de lignes :

```
private static void rangeCheck(int arrayLen, int fromIndex, int toIndex) {
  if(fromIndex > toIndex)
    throw new IllegalArgumentException("fromIndex(" + fromIndex + ")
       > toIndex(" + toIndex + ")");
  if(fromIndex < 0)
    throw new ArrayIndexOutOfBoundsException(fromIndex);
  if(toIndex > arrayLen)
    throw new ArrayIndexOutOfBoundsException(toIndex);
}
```

  - Demande 6 milliards de dollars et puis 2 milliards de dommages
  - Le 31 mai 2012 : le juge statue en faveur de Google (open source)
  - Le 9 mai 2014 : Oracle l'emporte en appel
  - Le 26 Mai 2016 : Google a fait appel de la décision et a gagné (fair use)
  - Le 27 Mars 2018 : Oracle a fait appel et a gagné (non fair use)
  - Un nouveau procès pour évaluer les dommages et Google qui fait appel à la court suprême (à suivre…)

Couche 3 : Android Runtime

- Processus de Compilation :
  - Deux passages :
    - JAVA vers .CLASS
    - Concaténation des .CLASS en .DEX
  - Une application c'est :
    - Le bytecode DEX
    - des ressources (images, sons...)
    - Le tout regroupé dans un package .APK

Couche 3 : Android Runtime

- Android inclus un ensemble de librairies de base proposant ainsi la quasi totalité des fonctionnalités disponibles dans le langage de programmation Java.
- Chaque application sous Android utilise sa propre instance d'une DVM :
  - Pas de problème d'interaction entres les applications
  - Espace protégé
  - Pas de risque de plantage général

Nécessité d'une VM optimisée !

Couche 3 : Android Runtime

- Au démarrage d'Android :
  - Une machine virtuelle est lancée afin de pré-charger presque 2000 classes.
  - Nom : **Zygote**
- Les instances de Dalvik initiées par le lancement d'applications sont des forks de Zygote.
- Un cache est mis en place dès le démarrage pour accélérer le chargement du bytecode DEX.
- Une machine virtuelle JAVA reposant sur un système
- GNU/Linux ne serait pas utilisable.
- Un mécanisme de compilation à la volée (JIT) permet d'accélérer l'exécution.
- Les Core Libraries intègrent l'API standard JAVA J2SE 1.5.

Couche 3 : Android Runtime

- Des fonctionnalités sont enlevées : toolkit SWING, fonctions d'impression
- Code Natif :
  - Codage via le Android NDK
  - JNI permet de faire le pont entre le natif et Dalvik
  - Peu utilisé sauf pour les jeux (habitude de programmeurs)
  - Permet des gains de performance parfois, mais cela dépend de l'application.

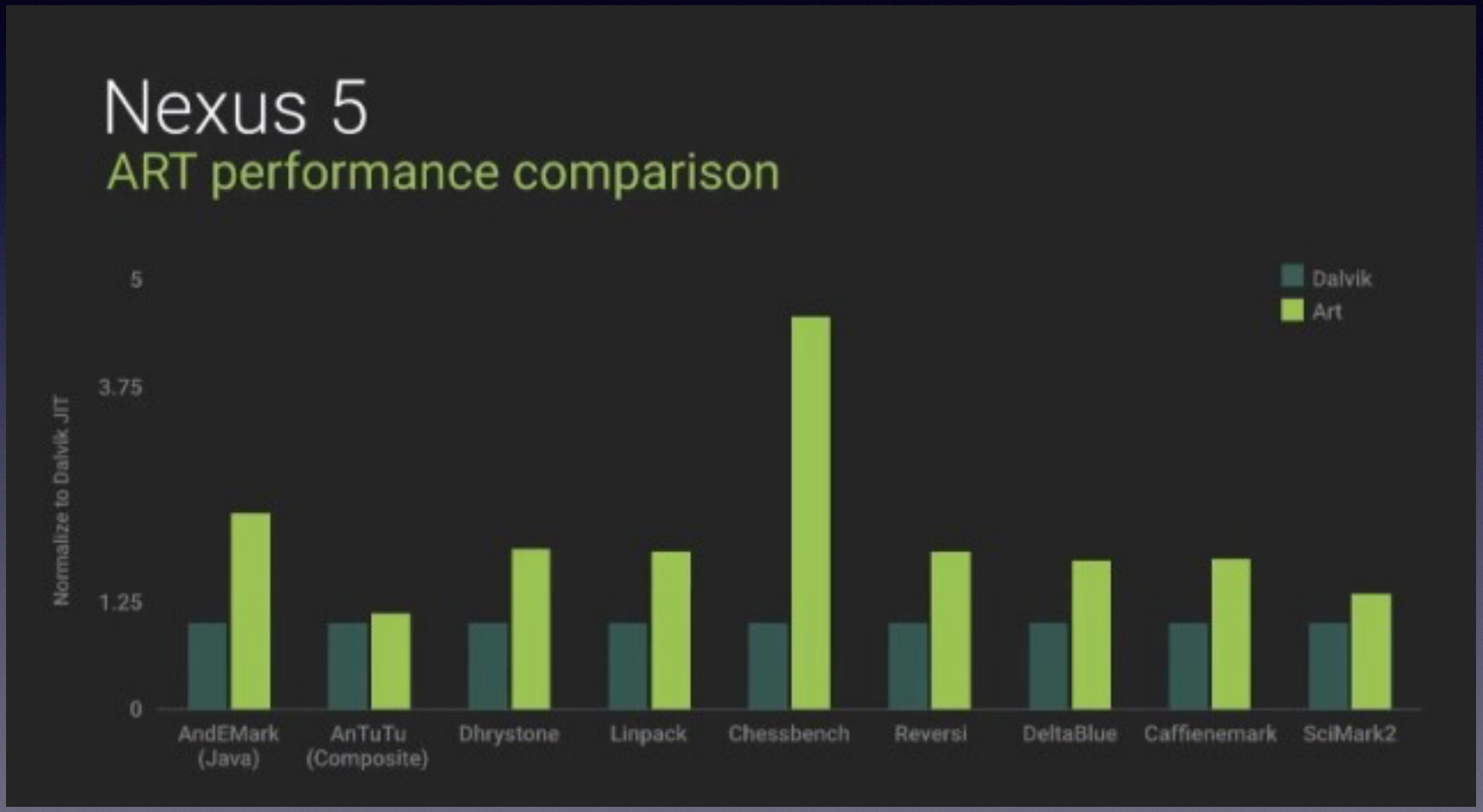
Couche 3 : Android Runtime - DALVIK VS ART

- ART pour Android RunTime
- KitKat (Android 4.4) a introduit dans le menu caché “développeur” la possibilité d’essayer ART
- Maintenant la norme
- La différence entre Dalvik et ART :
  - Moment d’interprétation du code :
    - Dalvik :
      - Interpréter le bytecode : Langage intermédiaire entre les instructions machines et le code source
      - A la volée pour être exécuté (JIT pour Just In Time)
    - ART
      - Compiler le bytecode (avant que vous en ayez besoin) une fois pour toutes
      - Avant son utilisation (AOT pour Ahead-Of-time)

Couche 3 : Android Runtime - DALVIK VS ART

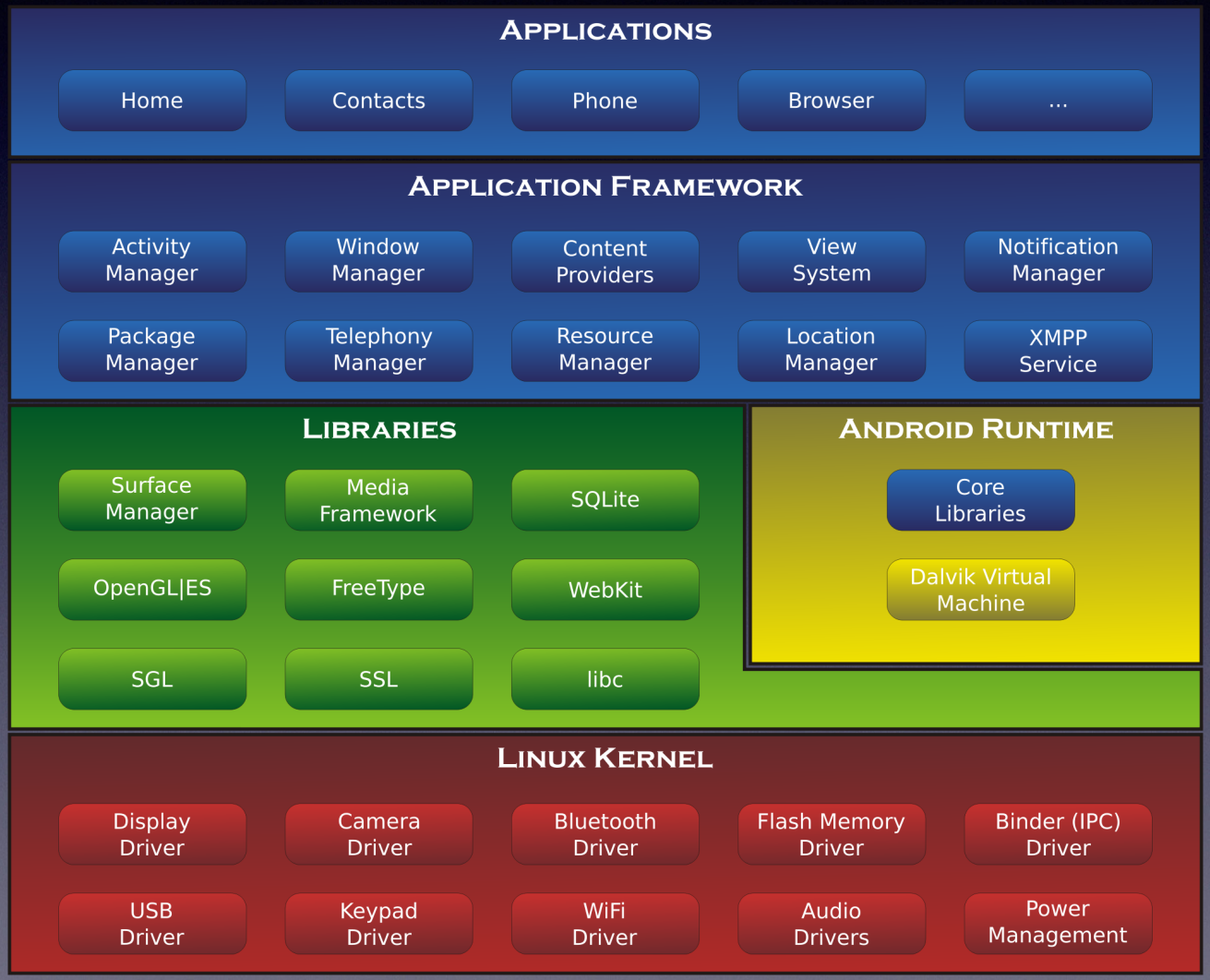
- ART plus performant :
  - en termes de rapidité d'exécution
    - DVM :
      - Le code interprété (bytecode) prend plus de temps à être exécuté
      - Une consommation électrique plus importante au niveau du processeur
    - ART :
      - Chaque application Android est compilée lors de l'installation
      - Plus de place dans la mémoire
      - Plus de temps pour l'installation mais beaucoup moins ensuite lors de l'exécution

Couche 3 : Android Runtime - DALVIK VS ART



Source : https://blog.newrelic.com/2014/07/07/android-art-vs-dalvik/

Couche 4 : Framework Applicatif



Couche 4 : Framework Applicatif

Couche 4 : Framework Applicatif

- Framework écrit en Java.
- Fournit tout ce que les applications ont besoin.
- API du framework décrite dans la documentation du SDK
- Eléments du framework :
  - Activity Manager : cycle de vie des applications (backstack) / Assure le multi tâche
  - Package Manager : Manipulation du format .apk
  - Window Manager : utilise Surface Manager / Gérer l'affichage des fenêtres.
  - Ressource Manager : Tout ce qui n'est pas du code (images, vidéos, etc.).
  - Content Manager : Partage des données entre processus (Contacts, etc.).
  - View System : équivalent au toolkit GTK+ / Gère le rendu HTML (Boutons, etc.)
  - Telephony Service : fournit l'accès aux services GSM, 4G, 3G, GPRS
  - Location Service : fournit l'accès à la gestion du GPS.
  - Bluetooth Service : permet d'accéder à l'interface bluetooth.
  - Wifi Service : permet d'accéder à l'interface Wifi.
  - Sensor Service : permet d'accéder aux capteurs (détecteurs de luminosité, etc.)

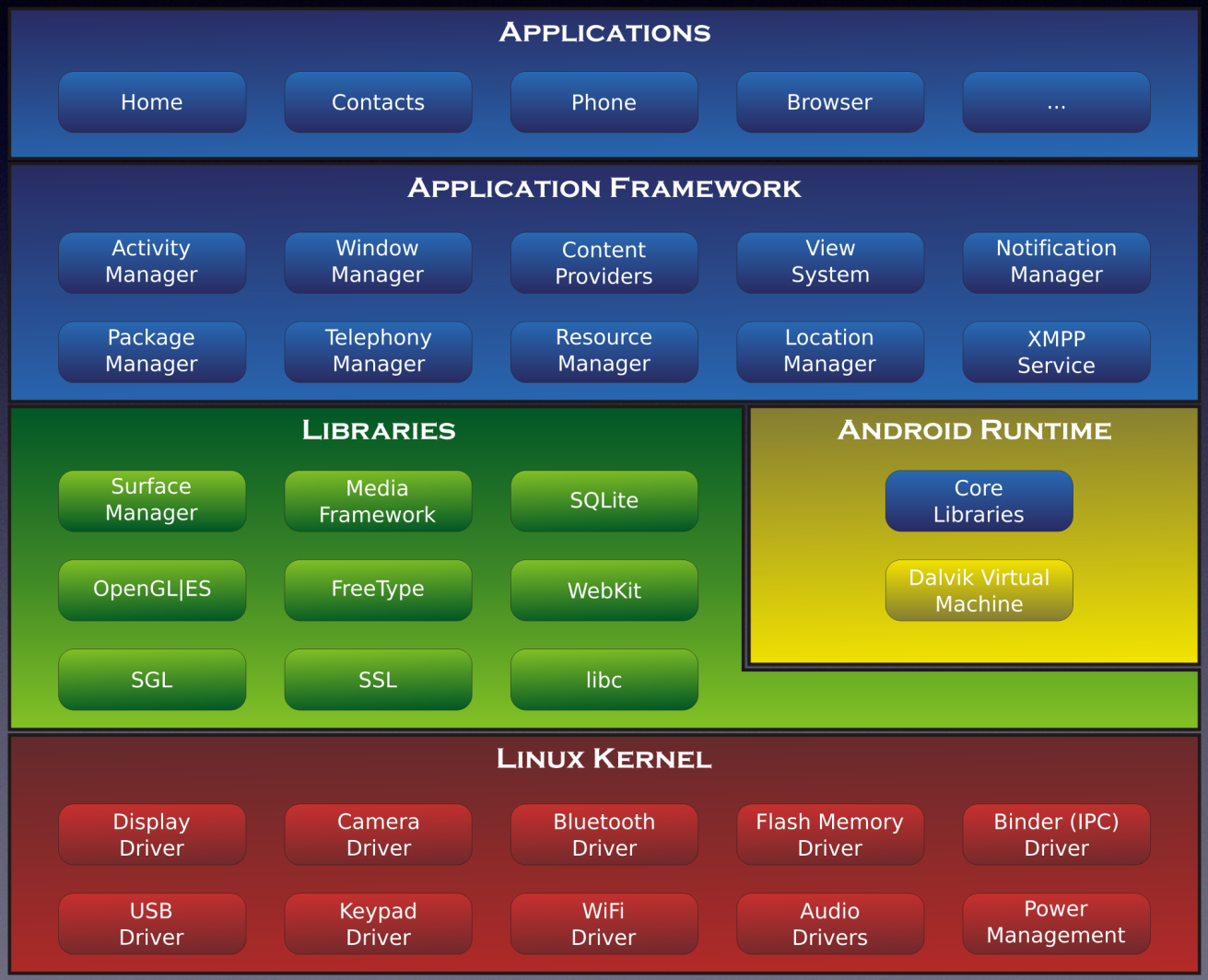
Couche 4 : Framework Applicatif

- Plateforme de développement Ouverte :
  - Permet de développer des applications riches et variées
  - Accès au matériel
  - Accès aux informations de localisation
  - Lancement de services de fond
  - Mise en place d'alarmes, de notifications,
  - Etc.
- Architecture conçue pour simplifier la réutilisation des composants
- Publication des capacités des applications
- Les autres applications peuvent utiliser ces capacités
- Chargement facile des apps.

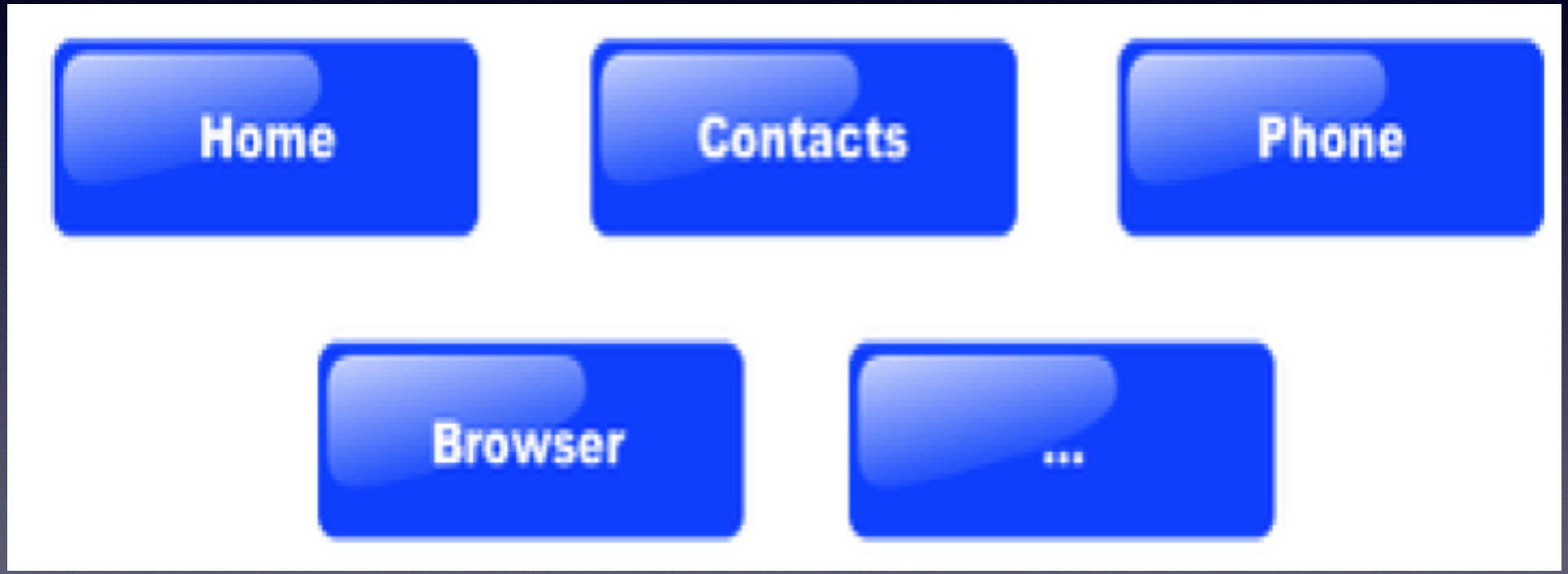
Couche 4 : Framework Applicatif

- Une application est composée d'un ensemble de services et de systèmes incluant :
  - Un ensemble de vues "Views" utilisées pour construire l'application (listes, grilles, zone de saisies, boutons ou encore navigateur web intégrable)
  - "Content Provider" permettant aux applications d'accéder aux données d'autres applications (Contacts...) ou de partager leur propres données.
  - "Resource Manager" permettant d'accéder a des ressources telles que des chaines de caractères, des images ou des "layout" (le tout est paramétrable selon de multiples critères : taille de l'écran, internationalisation…)
  - "Notification Manager" permettant à chaque application d'utiliser la barre de statut générale pour y intégrer ses propres informations.
  - "Activity Manager" : composant qui gère le cycle de vie d'une application et fournit les outils de navigation applicative.

Couche 5 : Applications



Couche 5 : Applications



Couche 5 : Applications

- 2 parties :
  - Les activités : des fenêtres interactives
  - Les services : tâches de fond.
- Les applications tournent dans leurs espaces pour la sécurité
- Communications entre applications : Les ”intent”
- Intent = intention : formulation une demande
- Plusieurs composants peuvent répondre à un ”intent”
- Dernière couche sur Android
- Plusieurs sont intégrées dans le système :
  - Ecran ”Home”
  - Gestion des Emails / SMS / MMS
  - Gestion de la téléphonie
  - Google Maps...
  - Application supplémentaires installables
  - Toutes les applications sont écrites via le même SDK

Introduction à la Programmation mobile

Architecture d'Android

Post-it

- Android est structuré en 5 couches.

- Il permet une abstraction complète du matériel disponible dans le Smartphone.

- Le noyau est basé sur Linux avec des importantes modifications.

- Les bibliothèques natives sont fournies pour la gestion du matériel.

- Le Runtime permet une utilisation performante et économique des applications en parallèle à travers des machines virtuelles DVM

- Google remplace progressivement les machines Dalvik par l'environnement ART plus performant

- Le framework permet de faciliter le le développement et le chargement des applications

- Android à la base est conçu par des développeurs pour les développeurs d'applications diverses et variées.