ANDROID

Laurent Provot <laurent.provot@uca.fr>

Septembre 2022

Introduction

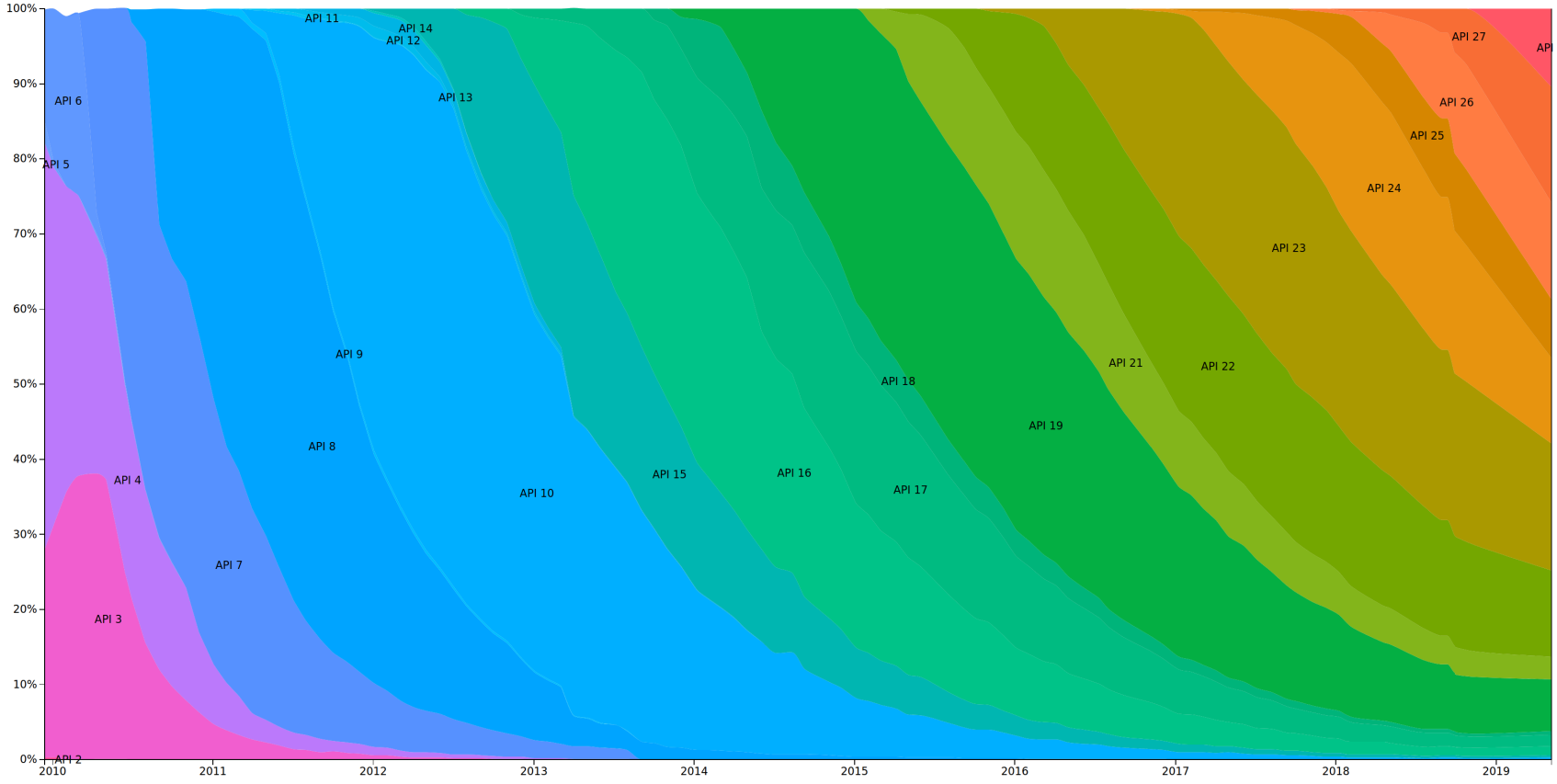
L'environnement

Spécificités du développement sur plateformes mobiles :

- Ressources limitées
- IHM variées
  - Écrans tactiles
  - GPS
  - Accéléromètres
  - Caméras
- Diversité des appareils
- Ultra-connectés

La plateforme Android

- Évolution très rapide



<https://www.bidouille.org/misc/androidcharts> (le site n'existe plus :'()

La plateforme Android

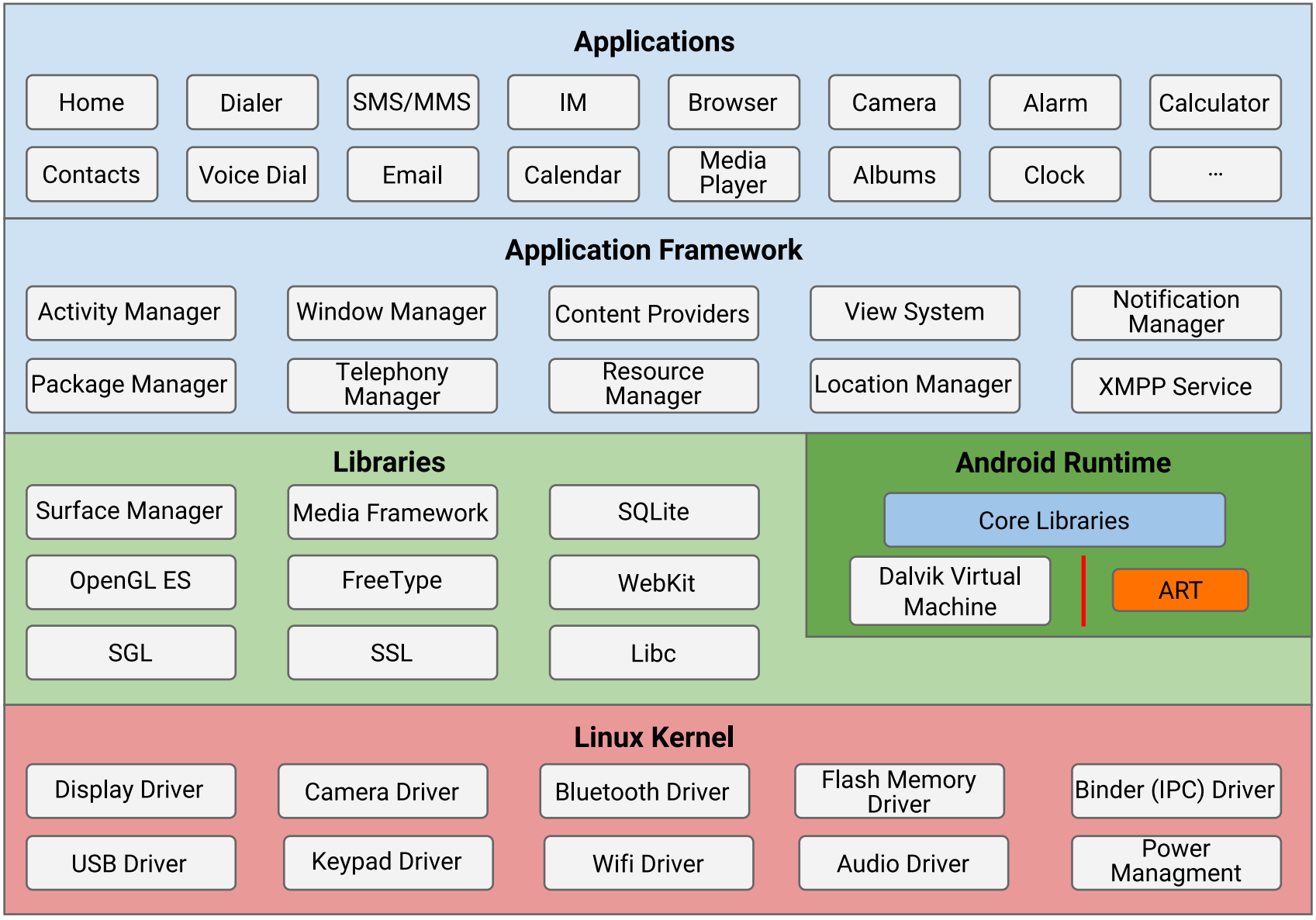
Fragmentation actuelle des versions du framework

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.1 Jelly Bean	16	
4.2 Jelly Bean	17	99,9%
4.3 Jelly Bean	18	99,7%
4.4 KitKat	19	99,7%
5.0 Lollipop	21	98,8%
5.1 Lollipop	22	98,4%
6.0 Marshmallow	23	96,2%
7.0 Nougat	24	92,7%
7.1 Nougat	25	90,4%
8.0 Oreo	26	88,2%
8.1 Oreo	27	85,2%
9.0 Pie	28	77,3%
10. Q	29	62,8%
11. R	30	40,5%
12. S	31	13,5%

source : Android Studio Dolphin (2021.3.2)

Architecture d'Android

Architecture logicielle d'Android



Organisation du système de fichiers

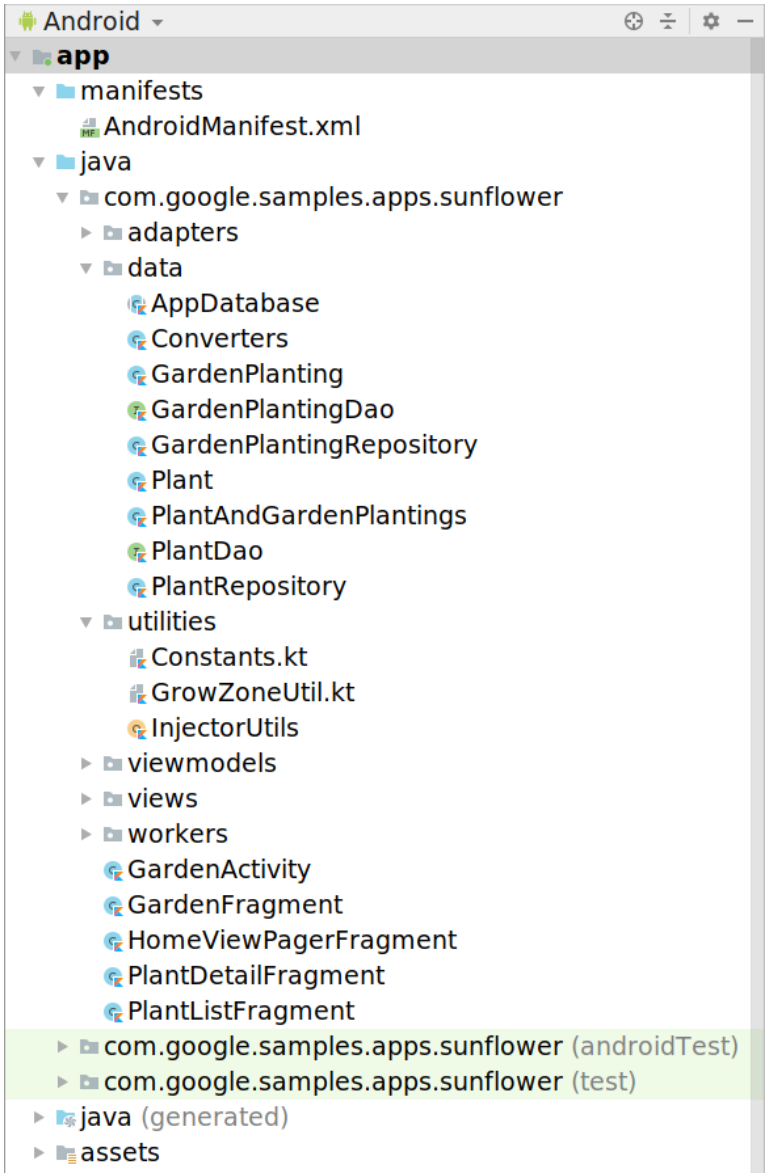
Partie système

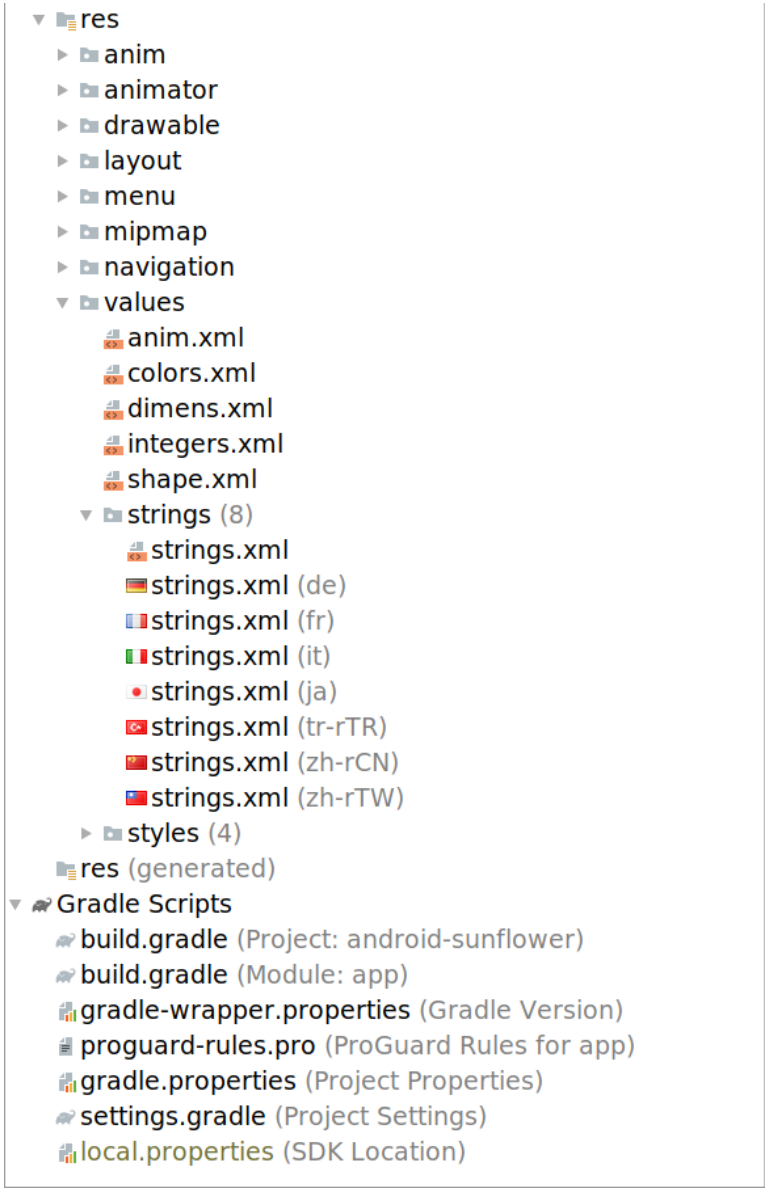
- Boot image : Noyau Linux et RAMDisk
- System image : OS Android et Applications
- Data image : Données utilisateur persistantes
- Recovery image : Utilisé pour recompiler / mettre à jour le système
- Radio image : Gère la partie « connectivité » de l'appareil

Partie utilisateur

- `/sdcard` : stockage interne ou externe

Architecture d'un projet Android





Les ressources d'une application Android

Dossier res : stockage des fichiers ressources

- anim : fichiers d'animations
- color : fichiers de description de listes de couleurs d'état
- drawable : images de résolutions différentes pour les écrans
- layout : fichiers XML décrivant la structure des vues de l'application
- menu : définition XML de menus
- raw : ressources définies dans des fichiers bruts (mp3, avi, ...)
- values : stockage clé/valeur de données simples (string, dimensions, couleurs, ...)
- xml : configuration d'objets avec des paramètres stockés en XML

Qualificatifs de ressources

1. Country code du système (*mcc310-mnc004*)
2. Code langage / region (*fr-rFR*)
3. Direction de lecture (*ldrtl*)
4. Tailles (*xlarge*)
5. Orientation de l'écran (*land*)
6. Densité de pixel (*xxhdpi*)
7. Version d'API (v11)

<http://developer.android.com/guide/topics/resources/providing-resources.html>

Le manifeste

Fichier `AndroidManifest.xml` : fichier de description de l'application, présent obligatoirement à la racine de l'application.

- Nom du package (Android) de l'application
- Descriptions des entités avec leurs capacités et comportements, ainsi que la manière dont elles sont lancées (`launchMode`)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="fr.iut.pm.od"
  android:versionCode="1"
  android:versionName="1.0" >
…
</manifest>
```

Le manifeste

- Donne le point d'entrée de l'application (activité lancée en première)

```
<application
  android:icon="@drawable/ic_launcher"
  android:label="@string/app_name"
  android:theme="@style/AppTheme" >
  <activity android:name=".CrimeListActivity"
            android:exported="true">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
  </activity>

  <activity android:name=".CrimeActivity"
            android:label="@string/app_name" >
    <meta-data android:name="android.support.PARENT_ACTIVITY"
      android:value=".CrimeListActivity"/>
  </activity>
</application>
```

Le manifeste

- Identifie les périphériques nécessaires pour faire fonctionner l'application
  - Puce Caméra, NFC, GPS, ...

```
<uses-feature android:name="android.hardware.camera" />
```

- Identifie les permissions
  - INTERNET
  - ACCESS_FINE_LOCATION
  - CAMERA
  - READ_CONTACTS
  - WRITE_EXTERNAL_STORAGE
  - SEND_SMS
  - ...

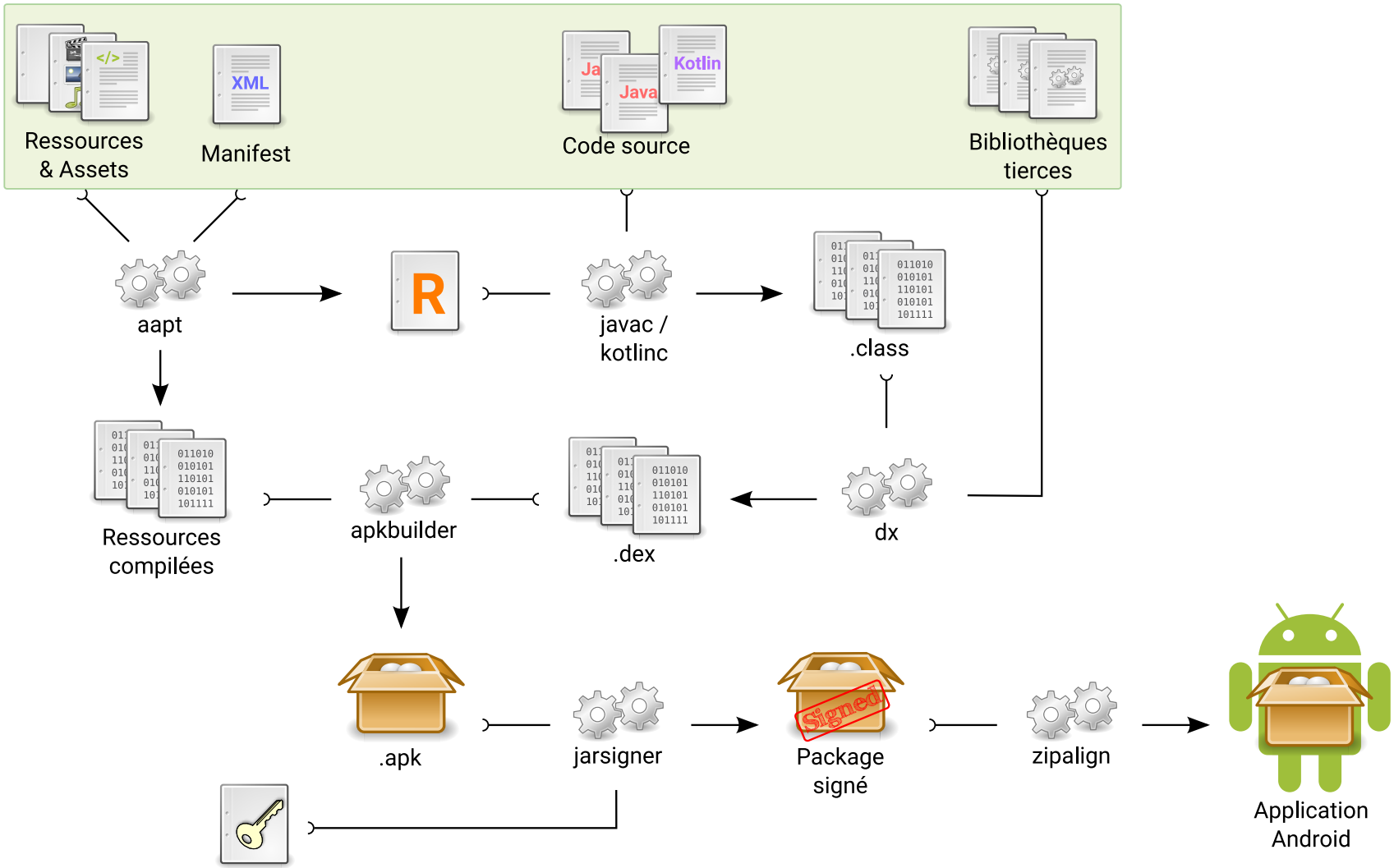
```
<uses-permission android:name="android.permission.INTERNET" />
```

Le manifeste

- L'attribut `<uses-sdk>`
- Donne les niveaux d'API de l'application (pas les versions)
  - `minSdkVersion` : niveau minimum d'API nécessaire pour le fonctionnement de l'application. S'il n'est pas renseigné, Android considère que l'application est compatible avec toutes les versions
  - `targetSdkVersion` : version où le fonctionnement normal de l'application a été testé
  - `maxSdkVersion` : attribut inutile depuis la compatibilité ascendante des API Android.

```
<uses-sdk android:minSdkVersion="8" android:targetSdkVersion="17"/>
```

Processus de construction d'une application



Les applications Android

Les entités principales

Les composants applicatifs :

- L'activité (`Activity`) : Écran d'interaction avec l'utilisateur pour effectuer une fonctionnalité (*modifier une note*, *numéroter un appel*)
- Le service (`Service`) : Tâche sans UI, non interactive (*jouer une musique*, *mettre à jour l'icône de température*)
- Le fournisseur de contenu (`ContentProvider`) : Gère l'accès aux données (*récupération des données d'un contact*)
- Le gaget (`AppWidget`) : vue miniature d'application pouvant être intégrée à une autre application (*gaget météo de l'écran principal*)

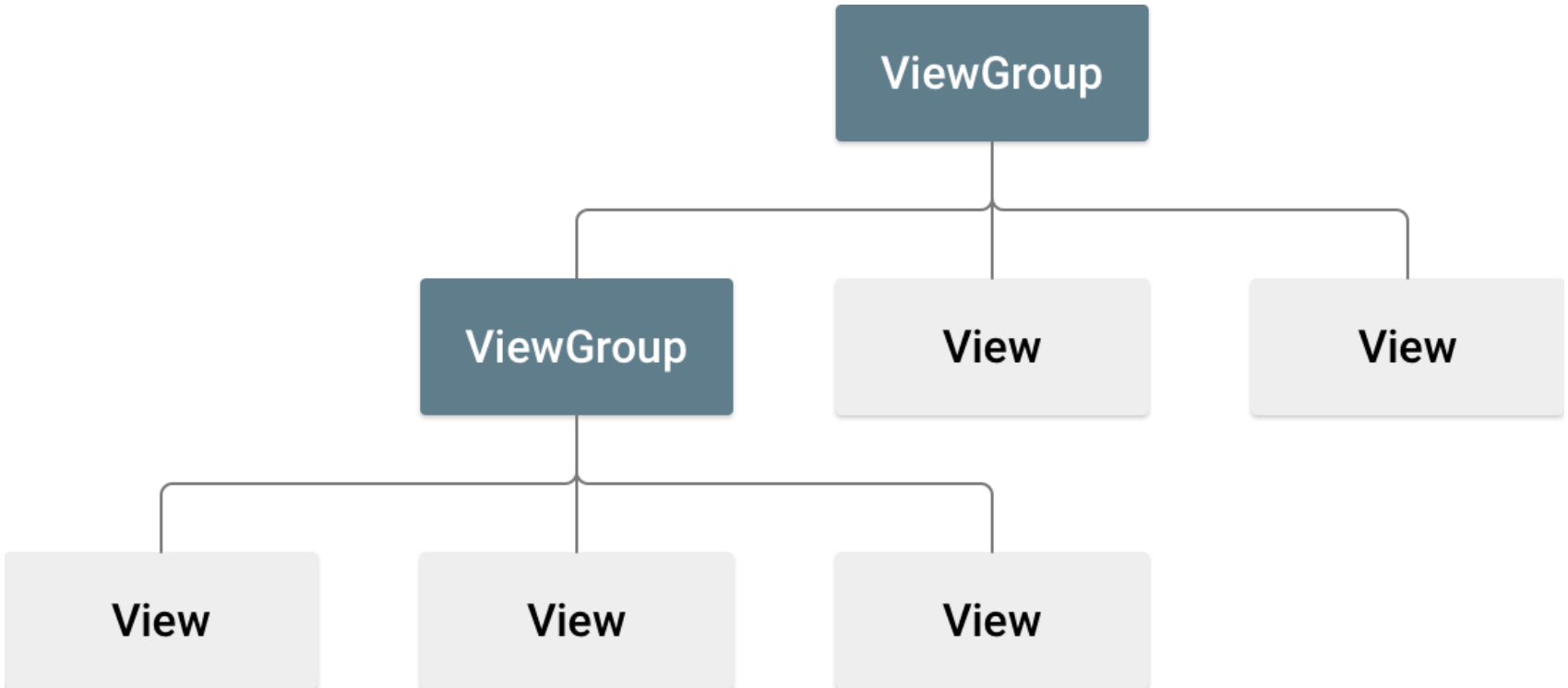
Les entités principales

Les composants d'interaction :

- L'intention (`Intent`) : Communications asynchrone entre les entités du système (*démarrer une application*)
- Le récepteur (`BroadcastReceiver`) : Reçoit et traite des messages (*déclencher une alarme à la réception d'un message*)
- Les feedbacks (`Notification` , `Toast` , `SnackBar`) : un message affiché dans la barre de notification ou une info-bulle temporaire (*arrivée d'un mail*, *traitement photo terminé*)

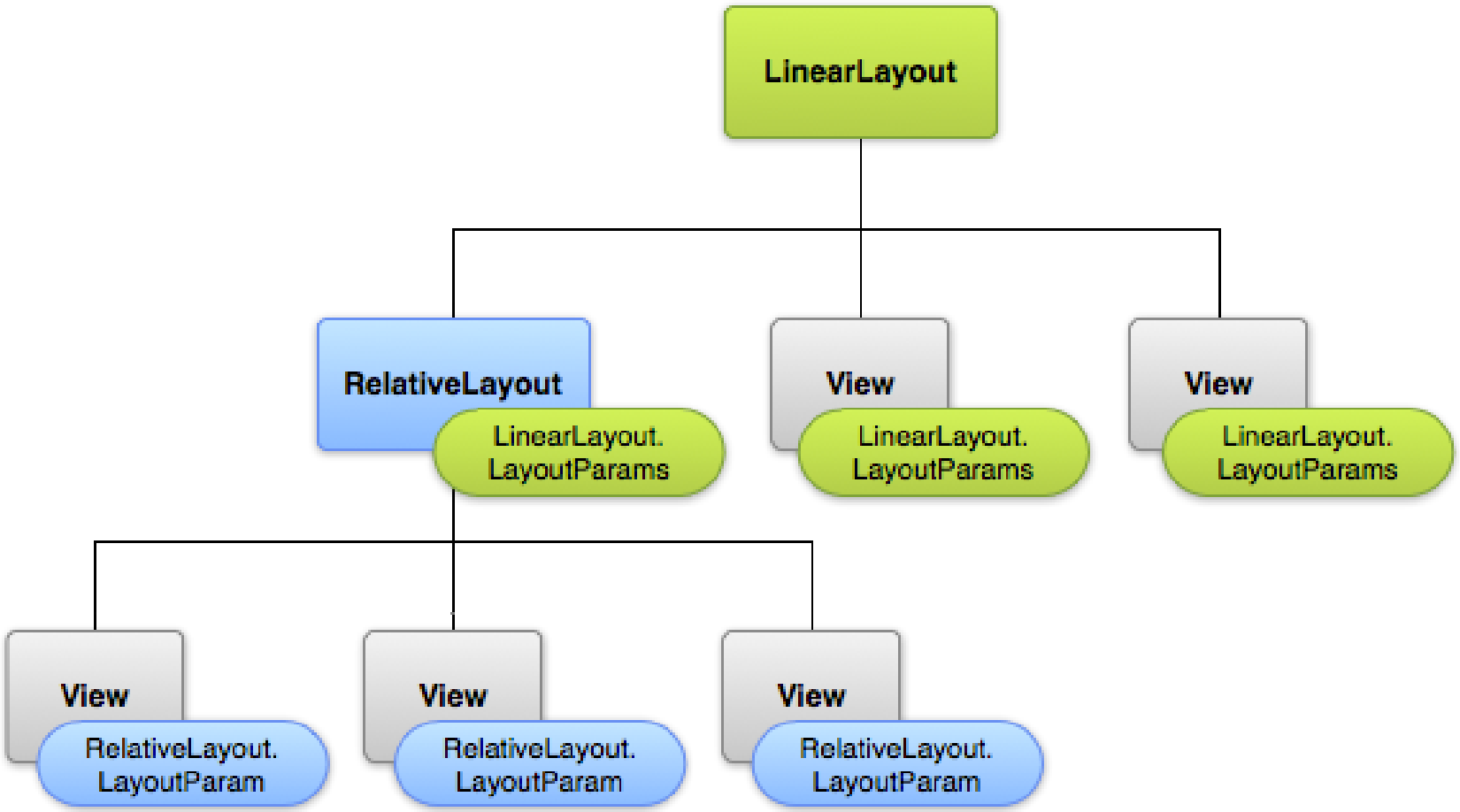
La partie visible des applications

- Gestion par design pattern Composite



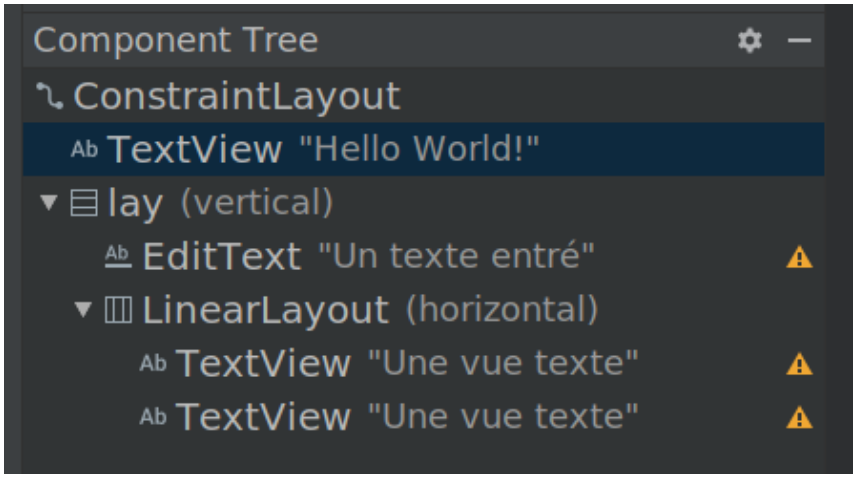
L'agencement des vues

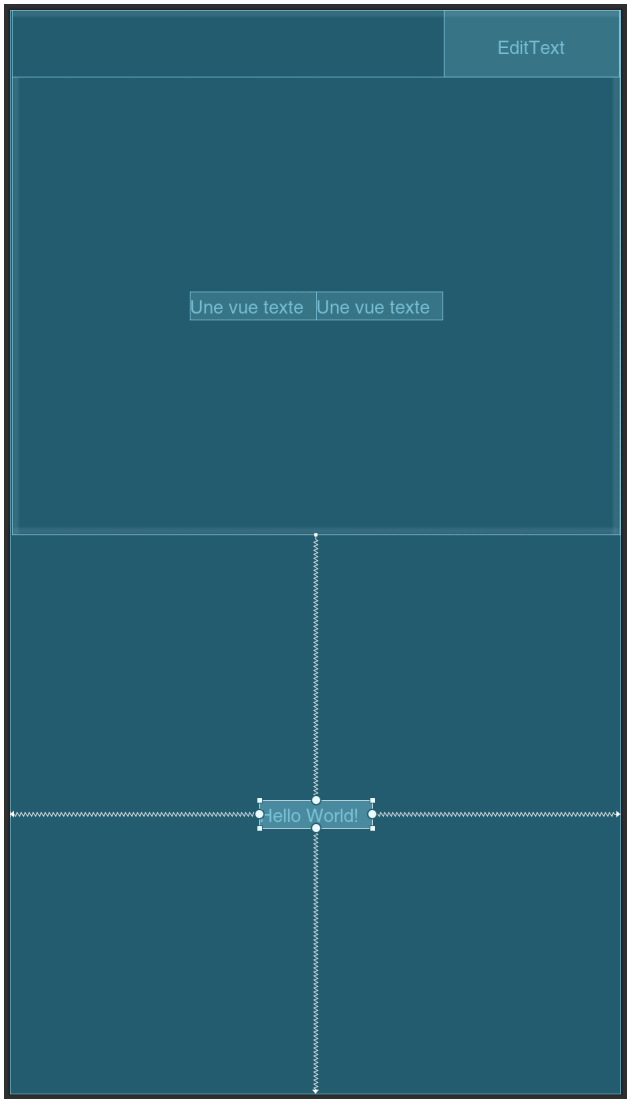
- Organisations par *Layouts* (comme en JavaFX, mais pas les mêmes)



L'agencement des vues

- Vue «Blueprint» classique





Navigation sous Android

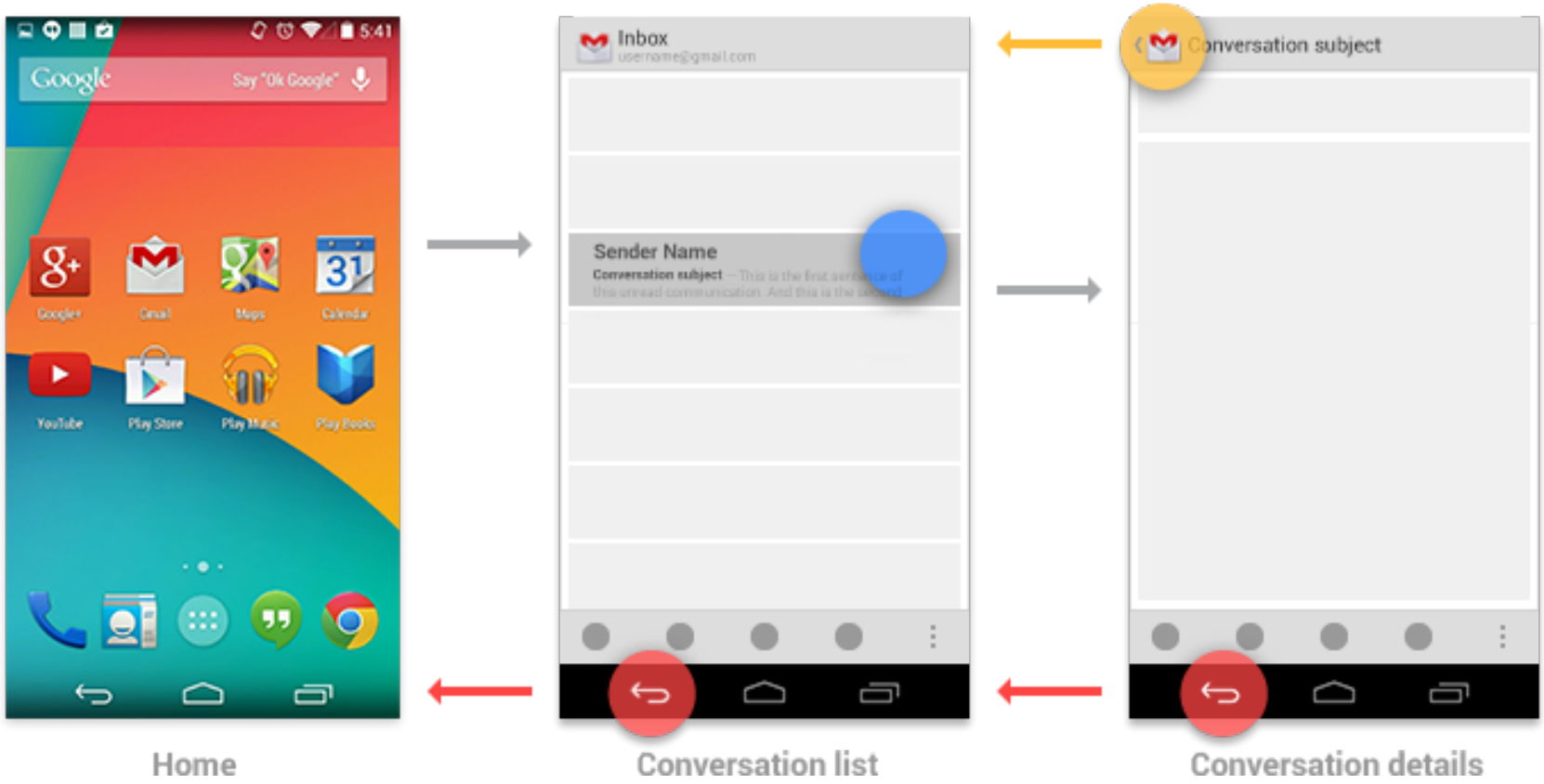
- 2 patterns classiques : *Back Navigation* et *Up Navigation*

1. Back Navigation
   - avec la touche  (ou  pour les ancienne versions)
   - navigation chronologique (en fait à travers les backstacks)
2. Up Navigation
   - avec l'App Bar 
   - navigation hierarchique entre les activités

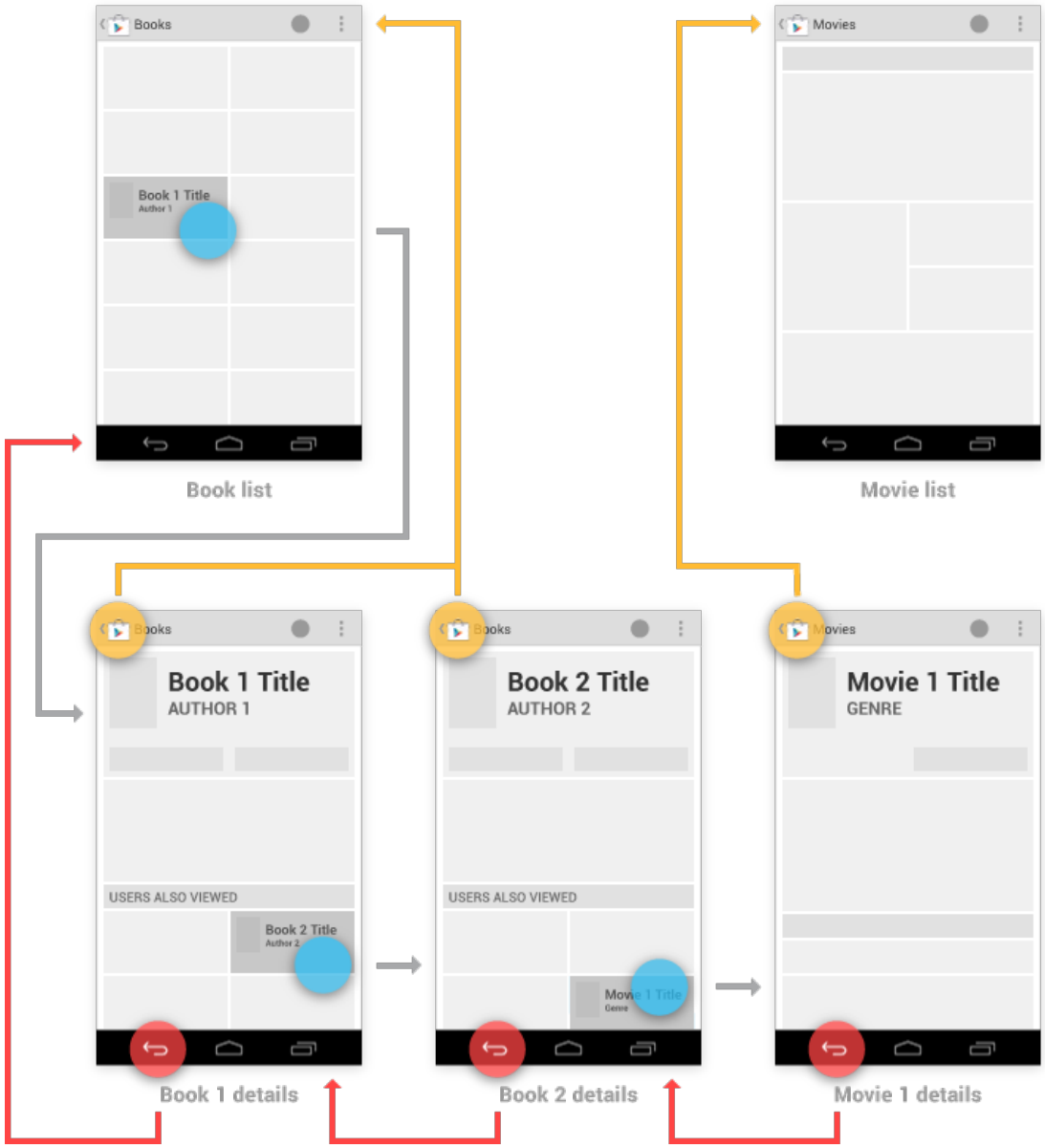
- Mais avec JetPack ça a un peu changé :-/

<https://developer.android.com/guide/navigation/navigation-principles>

Navigation sous Android (avant JetPack)



Navigation sous Android (avant JetPack)

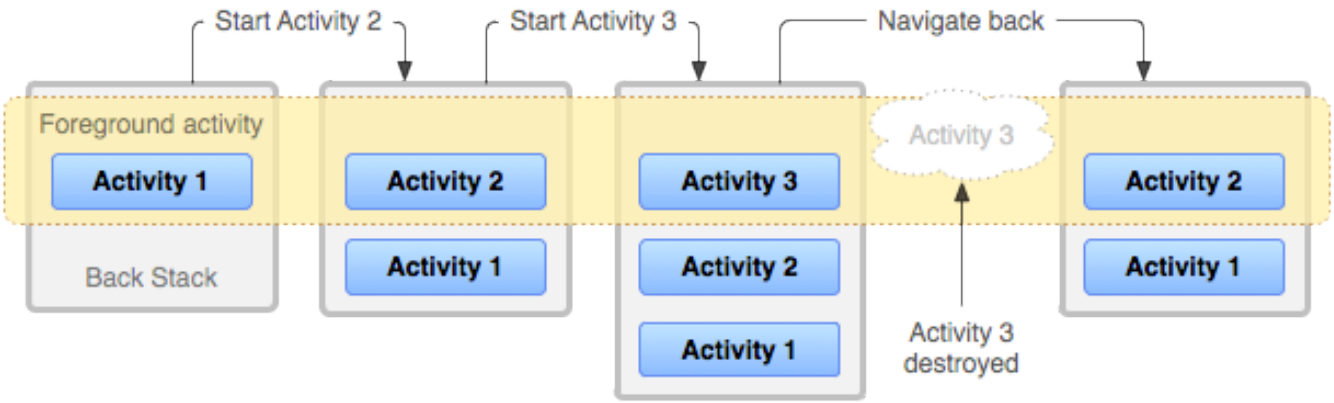


Gestion des applications

- Android **N**'est **PAS** mono-processus
- Chaque application est lancée dans sa propre machine virtuelle ART (ou Dalvik)
  - Mémoire propre
  - Espace disque réservé pour une application
  - UID propre
- Une application n'accède pas directement à l'espace disque d'une autre

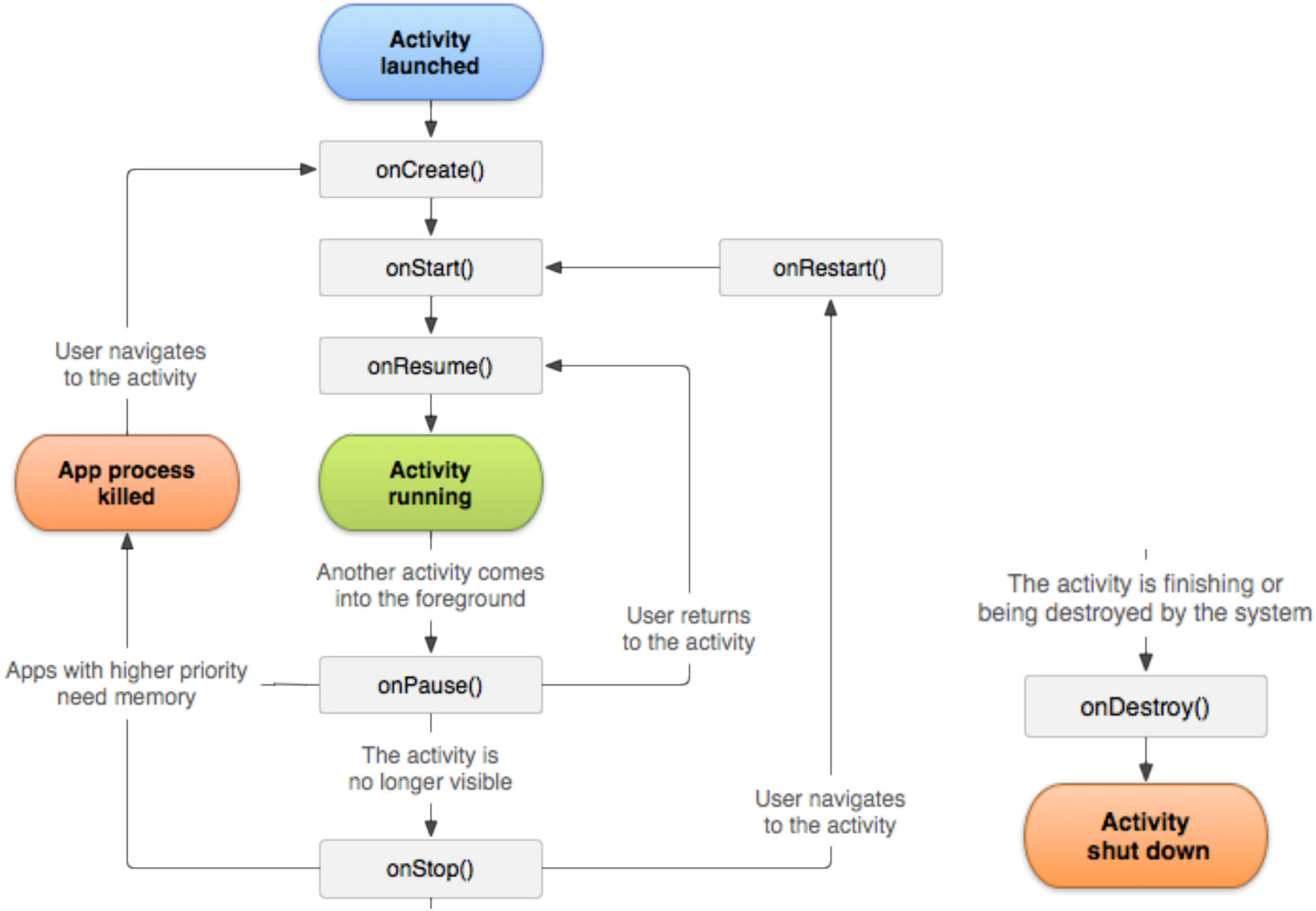
Fonctionnement des applications Android

- Le système Android gère des tâches (*tasks*)
- Chaque tâche possède sa propre pile d'activités (*back stack*)
  - Gérée par défaut comme une structure LIFO
- Appui sur `Back` : dépile l'activité au sommet de la pile et la détruit
- Appui `Home` : retourne à l'écran d'accueil et met la tâche en attente en arrière plan



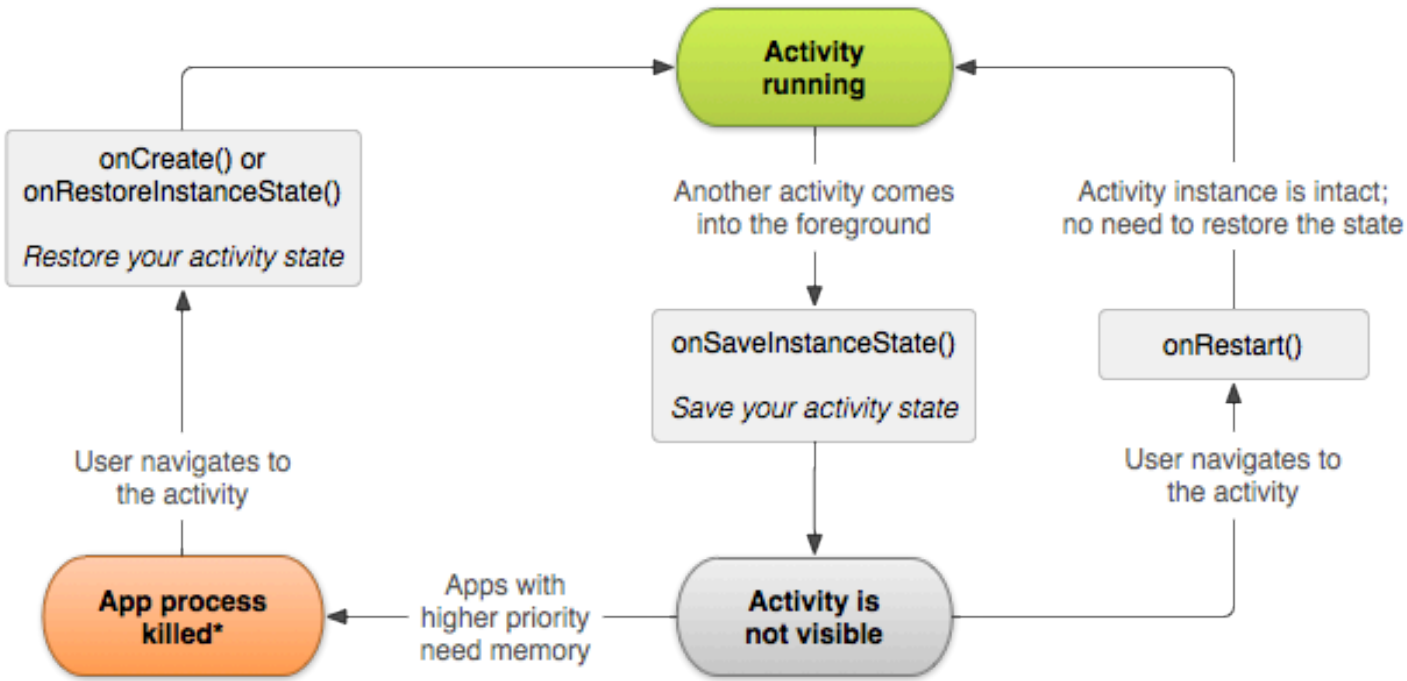
Tâches et navigation

Cycle de vie d'une activité



Cycle de vie d'une activité

- Si besoin de ressources, une tâche mise en attente peut être détruite
- Attention à bien gérer les données transitoires de l'application qui doivent être sauvegardées !
- `onSaveInstanceState(Bundle outState)` et `onRestoreInstanceState(Bundle savedInstanceState)`



Cycle de vie d'une activité

- Implémentation par défaut des `View` : sauvegarde automatique si elles possèdent un ID (`android:id`)
- Pour données transitoires et petites !
- Si données persistantes alors `onPause()`