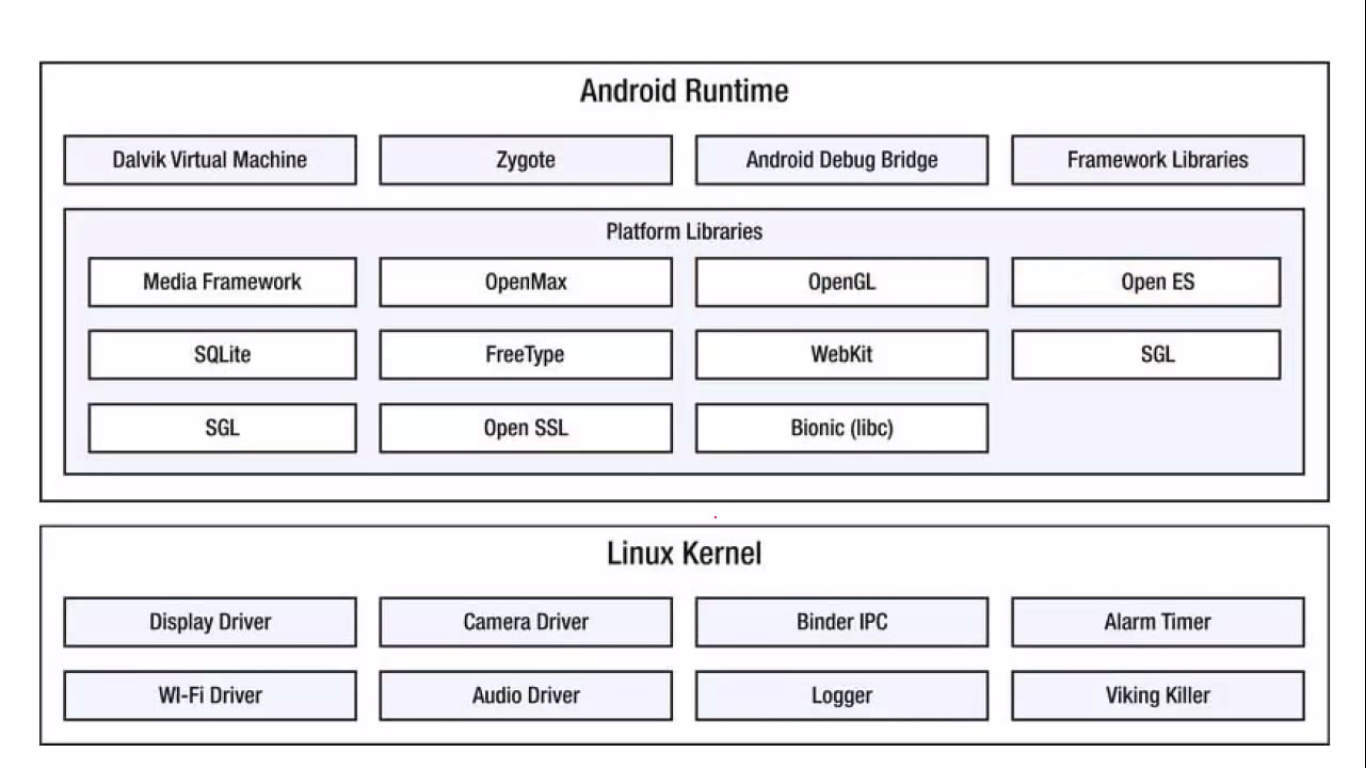
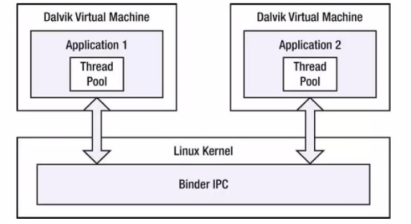
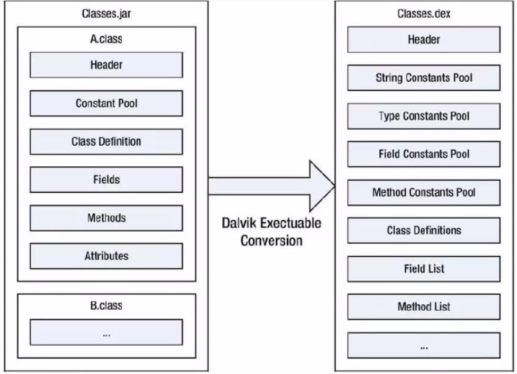
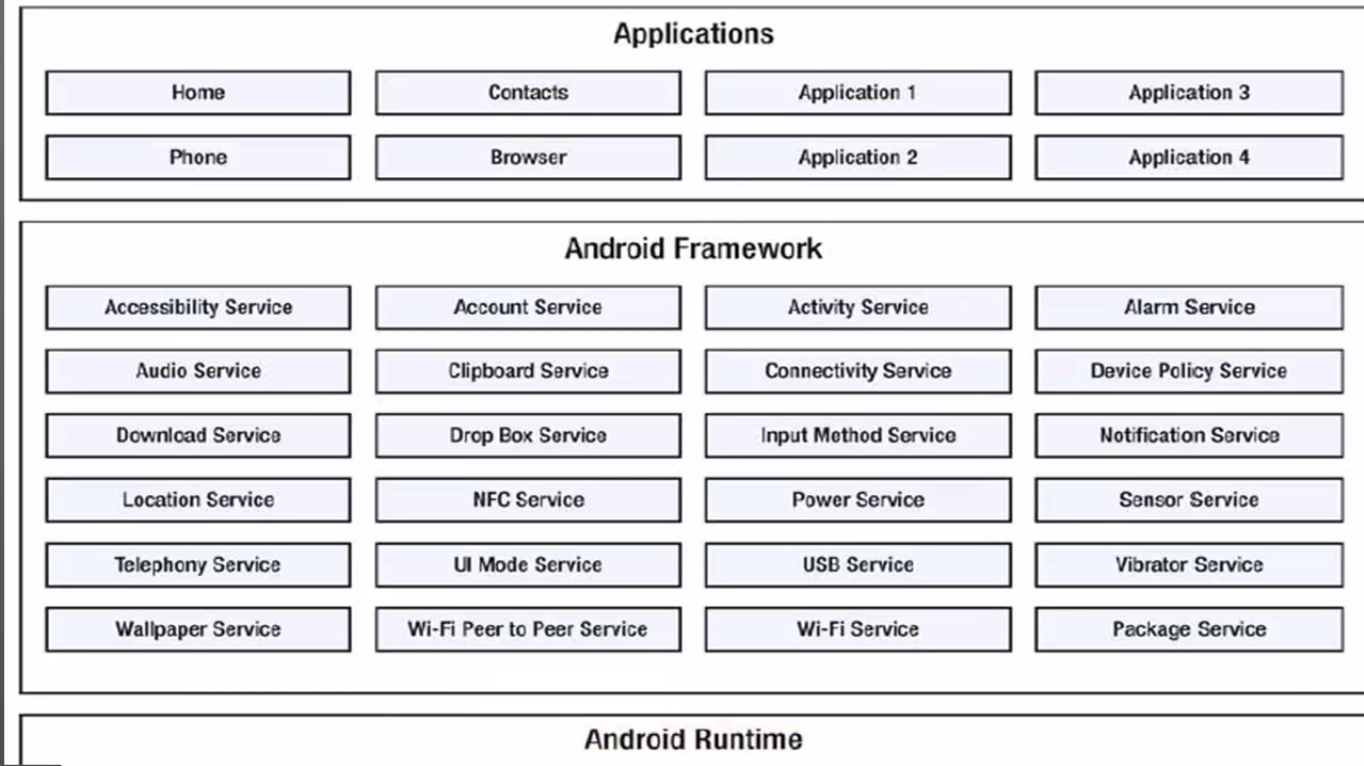
|  |
| --- |
| <https://rosarioterranova.github.io/> |

**Stack dell’architettura Android a basso livello**



* **Linux Kernel**: alla base di tutto a contatto diretto con l'hardware
  + **Display, Camera, WiFi e Audio Driver**: semplici driver per i relativi moduli
  + **Binder IPC** (legare internet process comunication): permette di collegare due app scambiandosi informazioni, o collegare un’app col sistema operativo (figura a destra). In ciascuna *dalvik virtual machine* abbiamo in esecuzione un’applicazione, che scambia dati con un'altra tramite il *Thread Pool* e il kernel.
  + **Alarm timer:** permette la granularità tra gli stati del dispositivo (spleep, on, ecc.), server a coordinarle.
  + **Logger:** registri di sistema dove vengono registrati gli eventi di app o del so.
  + **Viking Killer**: supercontrollore della ram che la matiene libera per le app. Quando la memoria è scarsa, il viking killer chiude tutte le app secondo una scala di importanza.
* **Android runtime**: livello sopra il kernel formato da
  + **Platform libraries**: librerie utilizzate dal runtime android scritte in C/C++
    - **Bionic**: supporto alle librerie standard del C.
    - **Media Framework**: librerie dedicate alla gestione di immagini, musica, video.
    - **SGL**: gestione grafica 2D
    - **OpenGL**: gestione grafica 3D
    - **OpenES**: versione snellita delle OpenGL
    - **FreeType**: sistema per il rendering dei font a video
    - **WebKit**: rendering delle pagine web
    - **OpenMax**: rendering dello streaming media con codec inclusi
    - **OpenSSL:** trasporto di dati con criteri crittografici per la sicurezza
    - **SQLite:** motore di database management system
  + **Dalvik virtual machine**: una java virtual machine modificata per essere resa più fluida e prestante, dunque ottimizzata, per dispositivi mobile. Il bytecode java non verrà però capito dalla dalvik perché è totalmente diverso, per questo è necessario convertire i file jar per essere letti da android in file dex (dalvik executable), unico con merge di più classi jar; il risultato è un risparmio di memoria di circa il 50%.
  + **Zygote**: componente che ottimizza la partenza della dalvik virtual machine in brevi tempi facendo dei fork (cloni) della dalvik originale
  + **Android debug bridge**: utility per modalità di debug dell’app via cavo o via emulatore a riga di comando composto da client a disposizione di un ide, un server e un demone in ascolto per emulare l’app sul device.

**Stack dell’architettura Android ad alto livello**



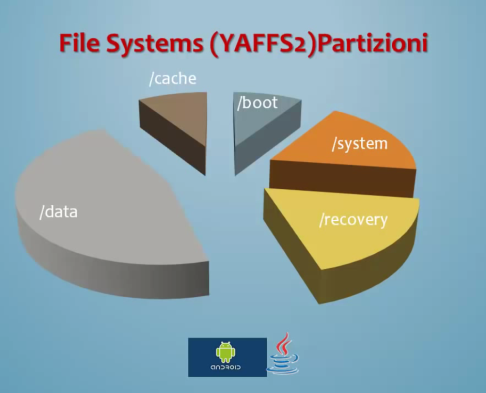
* **Android Framework**: contiene moduli utilizzabili tramite delle classi java pronte all’uso per gestire completamente le varie funzioni del device.
* **Applications**: qui troviamo le app vere e proprie che attingono i loro dati dalll’android framework.

**Cartelle e file di un apk**

* **Cartella progetto**: file necessari a creare il progetto, tutti questi file non vengono trasferiti nel device
  + **Bin**: troviamo il file *app.apk* dove vengono impacchettati tutti i file necessari al funzionamento dell’app.
    - **NomeApp.apk**: troviamo il file *classes.dex*, ovvero il bytecode della dalvik virtual machine, il file *resources.arsc* un indice di tutte le risorse della cartella *res* dove risiedono i file collegati. Il file *AndroidManifest.xml* utilizzato per i dati e informazioni sull’app.

**File System Android**

Android utilizza il file system YAFFS2 (Yet another file flash system) ottimizzato per memorie flash. Esso permette di aggiornare facilmente il device, sicurezza e stabilità grazie a partizioni separate del disco.

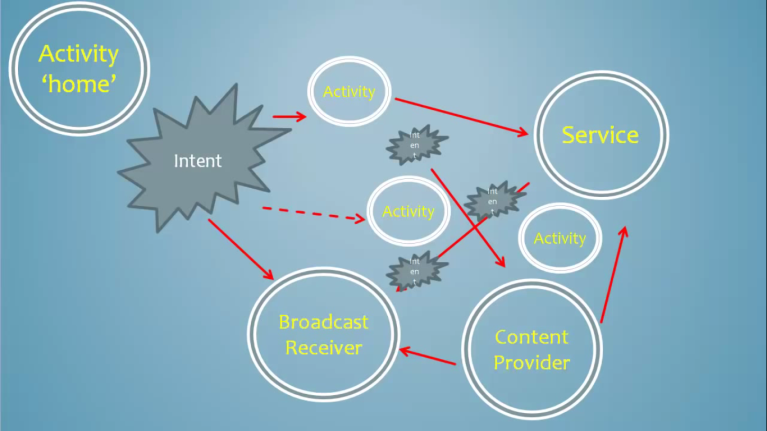
**/boot:** vi sono i file che permettono l’avvio del device, inaccessibile dall’utente e dalle app in scrittura;

**/system:**  sono concentrati tutti i dati di sistema android comprensive delle app preinstallate (non accessibile tranne il market);

**/recovery:** contiene un’immagine di sistema in modo da poter riportare il device alle condizioni di partenza in caso di ripristino;

**/data:** contiene i dati dell’utente e delle app installate (accessibile da tutti);

**/cache:** contiene una copia dei file che sono maggiormente usati.

**Logica delle app**

Una **Activity** corrisponde ad una schermata presentata all’utente, e la schermata principale o home è considerata l’activity dalla quale si innesca tutto il resto; l’utente interagendo con la schermata activity innesca i così detti **intent**, messaggi inviati agli altri oggetti del sistema, ovvero un’oggetto che descrive un’operazione che deve essere svolta. Come risultato un’altra activity potrebbe essere portata in primo piano, e così via. Tutte le activity sono memorizzate nello steck activity.

Il **Content Provider** è un componente di sistema che è in grado di fornirci i file necessario fornendo le risorse come file dati, video, audio o immagini, o estraendone tramite sql da un database; essi non hanno una loro activity ma possono esserne controllate da una.

I **Service** svolgono operazioni anche quando l’attività o l’itent che l’ha innescata è stata tolta dal primo piano, lavorando in background (es. servizi upload o download, riproduzione audio, ecc.).

**Brodcast receiver**, intercettatore di eventi, invia notifiche all’utente.

Questi moduli cooperano e si scambiano dati tra loro.

**Controlli per l’interazione con l’utente**



View

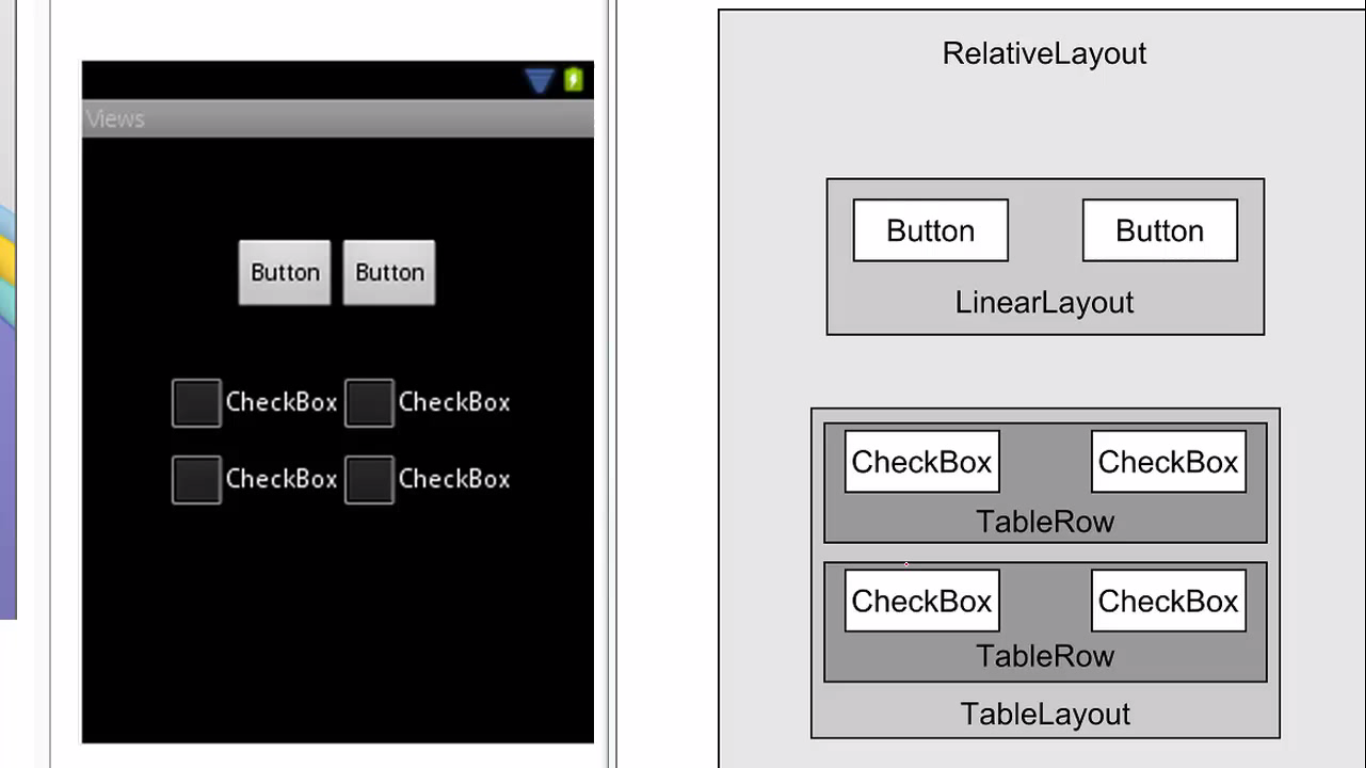
All’interno di un’activity troviamo tante **View**, ad esempio un bottone da premere (chiamato **Widget**) è una view a cui è stata aggiunta la possibilità di iterazione con l’utente. La view definisce i confini dell’area, il contenuto e i parametri del layout. Altre view invece sono statiche e non prevedono l’iterazione con l’utente (come le **Progress Bar**).

Una **view group** è una classe derivata dalla View e aggiunge una schematizzazione alle view che contiene.

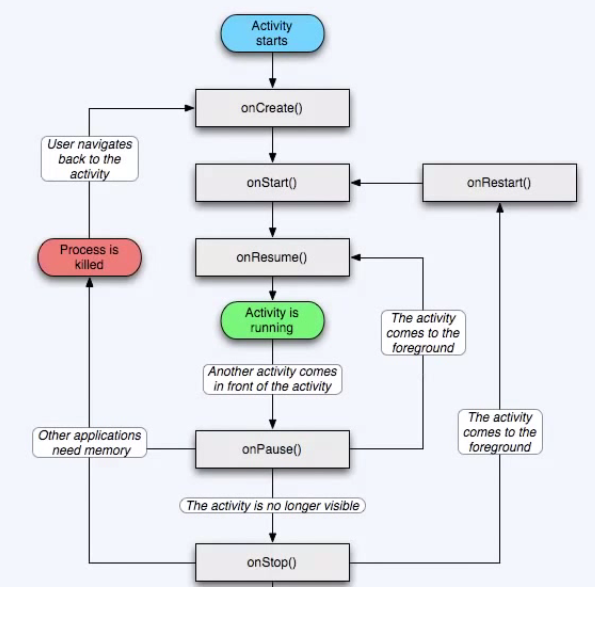
Layout

Contenitori di controlli, definiscono la gerarchia interna di una view e come ciascuna view dovrà essere riposizionata sul display a seconda del tipo di device su cui l’app sta girando.

*Es*. La schermata a sinistra è organizzata in layout come a destra, con un *RealitiveLayout* che permette di disporre tra di loro in modo relativo, ovvero al ridimensionamento la distanza tra gli oggetti rimane la stessa; un *LinearLayout,* che dispone gli oggetti contenuti in linea orizzontale linearmente e decide lo spazio usando dei pesi, per i bottoni; e un *TableLayout* per le CheckBox.



**Cartelle e file di un progetto**

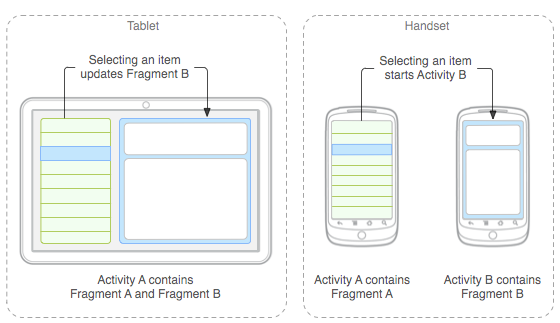
Una classe Activity ha lo schema riportato a sinistra; Quando creo la mia activity sto estendendo la classe Actitvity principale e produrre un override dei metodi della classe madre.

Il codice sorgente lo troviamo nella cartella **src**, mentre le risorse invece nella cartella **res**. La cartella **gen** invece contiene tanti file generati automaticamente e costanti che servono al compilatore.

La libreria *appcompat\_7v* invece è stata introdotta col nuovo aggiornamento per risolvere il problema della troppa frammentazione dei device android.

Nel percorse **res/values/strings.xml** invece troviamo le etichette delle stringhe utilizzate nelle activity.

Fragments

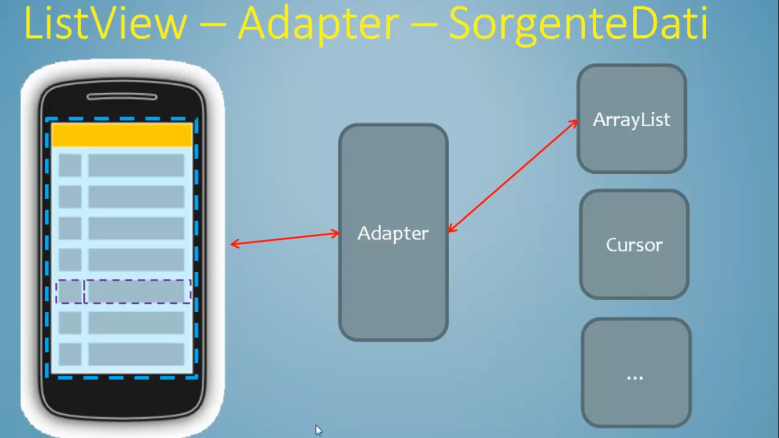
Dalla versione 3.0 di android sono stati introdotti i **fragments**, ovvero invece di sovraccaricare il sistema con l’aggiornamento di un’intera superficie di un activity, il fragment rappresenta una specie di unità sub-acivity, quindi i pezzettini di un’activity che individuano delle zone indipendenti dalle altre in forma rettangolare che si possono aggiornare indipendentemente dalle altre. Quindi potrei avere un’activity suddivisa in 4 frammenti.

Oltre a un guadagno di prestazioni, un frammento ben progettato può rappresentare un’unità riutilizzabile per altre activity.

*NOTA*: Dalla nuova versione dell’SDK di Maggio 2014, sono state reintrodotte le activity come si era soliti usare, quindi i fragment rimangono solo per un discorso di adattabilità a diverse soluzioni-

**Componenti**

ListView

Ha un suo layout specifico, ogni voce è gestita dall’**adattatore** che prende i dati da una sorgente dati (ArrayList, Cursor, Database SQLite, ecc.) con cio sa dialogarci, e gli riversa in una ListView.