# 3. ANDROID

Android è un sistema software basato sul kernel linux.

Gli **utenti** **del sistema android sono le** **applicazioni** che hanno accesso alla loro porzione di memoria, separata da quella delle altre applicazioni.

Quando un’applicazione è lanciata su un device, viene creato un nuovo **processo**. L’idea del processo in android è diversa da altri:

* **Application manager decide**
  + Quale processo può continuare a vivere e per quanto tempo
  + Quale processo eliminare e ricreare
* **L’utente non ha decisioni sui processi**
  + Quando un utente chiude un’applicazione, il processo potrebbe continuare a vivere
  + Se ho problemi con le risorse, l’application manager potrebbe decidere di uccidere un’applicazione aperta e far ripartire quell’app dopo chiedendole di ripartire da dove era rimasta.

**L’applicazione nasce come una conseguenza di una unix fork** system call (non scriviamo noi un main), inizia da una fork di un processo esistente. Quindi, la maggior parte dell’inizializzazione è già fatta.

Un programma Android è un **callback, reactive style**: Non ho il controllo su cosa succede ma su come reagire

* Quando Android mi dice che l’applicazione è creata → faccio questo
* Quando Android mi dice che l’applicazione sta per essere killata → faccio questo

*L’applicazione deve svolgere le sue operazioni in modo safe perché potrebbe essere killata in un qualsiasi momento.*

**Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteKotlin coroutines**: gestire l’esecuzione concorrente di task su kotlin.

Android elimina molti dei interProcessComunication forniti da linux e li ha irmpiziazzati con i propri perché in questo modo garantisce la sicurezza delle applicazioni.

Network driver: molto potente gestione del wifiDirect e altro.

Librerie: per gestione media, etc… vengono linkate all’applicazione

**Kernel gestisce:**

* Permessi e sicurezza
* Memoria a basso livello
* Processi e thread
* Network layer → fornendo accesso all’utente se vuole settare vpn, etc.
* Display, tastiera, camera, memoria flash, file audio

**L’applicazione runna su una** **ART VirtualMachine**, basata sui registri. Quando lancio un’app sul cellullare, viene creato un processo contenente una dalvik/ART virtual machine **forkando un altro processo**.

L’applicazione gira **sul suo spazio designato** e non può andare da altre parti. Se l’applicazione sta usando dei device (sensore gps), deve dichiarare in maniera formale che necessita di accedere a quell’HW →feature use. Ogni applicazione necessita l’esplicita permissione dell’utente per accedere ai suoi dati privati.

Android è formato da un insieme di API:

* Classi che possono includere funzionalità

**Struttura dell’applicazione:**

* Dati e codice per supportare l’end user per fare dei task
* L’insieme delle informazioni sono storate in un pezzo di flash disk in uno spazio privato dell’applicazione
* L’applicazione runna nel contesto utente che è stato creato durante l’installazione
* Non hanno un entry point (main) ma 1+ componenti attivati dal sistema operativo

*I componenti di un’applicazione sono distribuiti in un singolo pacchetto sw (APK) e descritti nel manifest file*

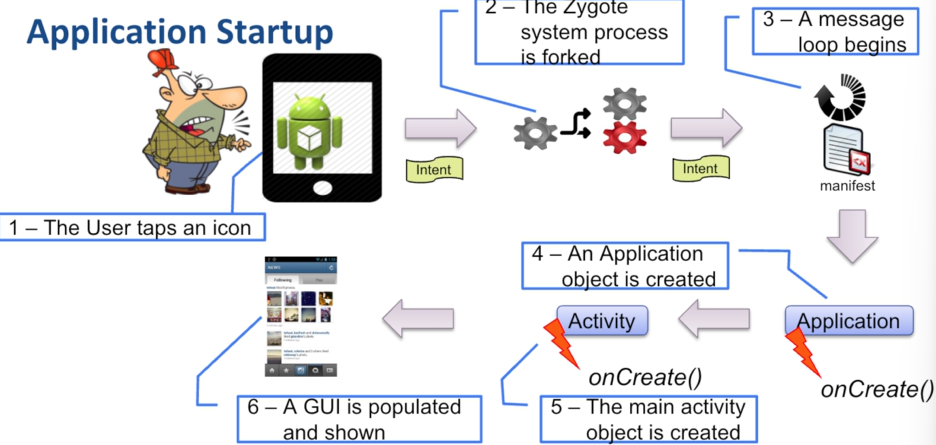
L’applicazione consiste in **4 componenti** creati sotto il totale controllo del S.O.:

* **1+ Activity**: componente sw per iterazione con utente che:
  + Immagine che contiene testo, schermata, diagramma

    Descrizione generata automaticamenteHa una GUI
  + Può fare task dentro un’app
  + Ciclo di vita:
    - Creata
    - Rappresentazione visiva
    - Visibile →started phase
    - L’utente può usarla
    - Se arriva una phone call l’applicazione è ancora visibile ma non interactable. Se accetto la chiamata, l’applicazione verrà stoppata. (potrebbe anche essere killata momentaneamente)
  + Estende la classe android.app.activity
* **Services**: per operazioni in background (non necessita user interaction):
  + Play music in background
  + Scaricare dati in background
  + Svolgere dei task predefiniti quando la cpu è già attiva e non quando è in deepSleep
  + Sottoclasse di android.app.Service
* **Content provider**: componente speciale che gestisce l’insieme delle informazioni necessarie ad un’applicazione e le fornisce in una maniera programmatica → comunicazione tra applicazione (shared with)
  + Es: la mia applicazione stora immagini che possono essere fornite anche ad altre applicazione
    - Allora il content providere gestisce queste immagini in una maniera sqlLite in modo tale che altre applicazioni possano ottenerle con una sqlLite command
  + Es: la rubrica fornisce i miei contatti a WhatsApp
  + Sottoclasse di android.content.ContentProvider
* **Broadcast receiver** : componente che attente messaggi → triggerati da eventi
  + *Es: batteria bassa*
  + *Es: no connessione*
  + *Es: messaggio broadcast da una specifica applicazione*

**ANDROID APP APPLICATION**: classe obbligatoria, se creo un processo, è la **prima classe che istanzio**. Questa istanza rimane allocata più di ogni altro componente, è **l’ultima ad essere killata**. Volendo si possono creare delle sottoclassi di A.A.A. per usarla a mio proposito, esempio storare informazioni che durino almeno quanto il processo corrente. Il manifest file dell’applicazione indica il nome dei componenti e A.A.A..

### Processo

1. Utente clicca icona
   1. S.O. fornisce una descrizione di cosa è accaduto in un messaggio → **intent**
   2. Intent is packaged e spedito al processo forkato
2. Processo forkato legge l’intent e inizia il loading dell’application che contiene dei file, tra cui il manifest file
   1. Nel manifest file sono presenti una serie di componenti ed è specificata la Application Class → una sua istanza sarà creata
3. Message loop
4. Invocato onCreate() method per Application
   1. Crea la user interface la mostra
5. Invocato onStart() e onResume() method
6. GUI popolata e msotrata all’utente