

ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
Dipartimento di Informatica – Scienza e Ingegneria (DISI)
C.d.S. in Ingegneria e Scienze Informatiche, Campus di Cesena

Programmazione in Android

Introduzione ed Elementi Fondamentali

Angelo Croatti
a.croatti@unibo.it

Sistemi Embedded e Internet of Things
A.A. 2019 – 2020

Outline

1 Introduzione

2 Android OS

3 Applicazioni Android

- Building Blocks
- Il file Manifest
- Intent



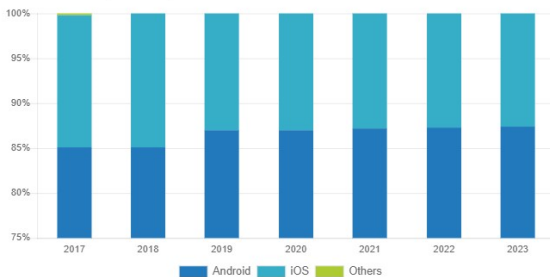
Google Android



- Sistema operativo per dispositivi mobili basato su Linux Kernel
 - ▶ Attualmente sviluppato da Google Inc.
 - ▶ Distribuito sotto licenza *open-source* (licenza Apache 2.0)
- **Sistema Operativo Embedded**, progettato per essere eseguito principalmente su dispositivi con interfaccia touch-screen
 - ▶ Ne esistono diverse varianti, per altre categorie di dispositivi
 - ▶ es. **Android Wear O.S.**, **Android TV**, **Android for Cars**, **Android Things**, ...

Mobile OS Market Share

Worldwide Smartphone Shipment OS Market Share Forecast



Year	2017	2018	2019	2020	2021	2022	2023
Android	85.1%	85.1%	87.0%	87.0%	87.2%	87.3%	87.4%
iOS	14.7%	14.9%	13.0%	13.0%	12.8%	12.7%	12.6%
Others	0.2%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
TOTAL	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

» <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>

Outline

1 Introduzione

2 Android OS

3 Applicazioni Android

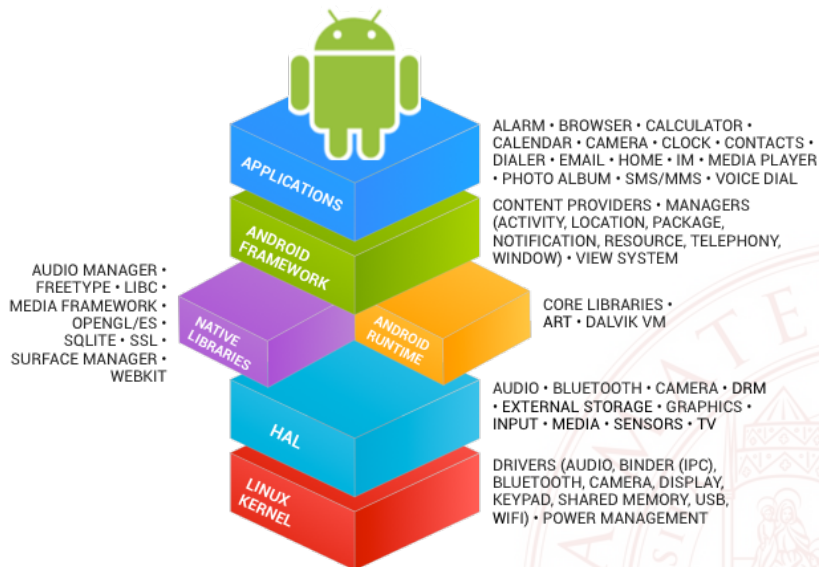
- Building Blocks
- Il file Manifest
- Intent



Android OS

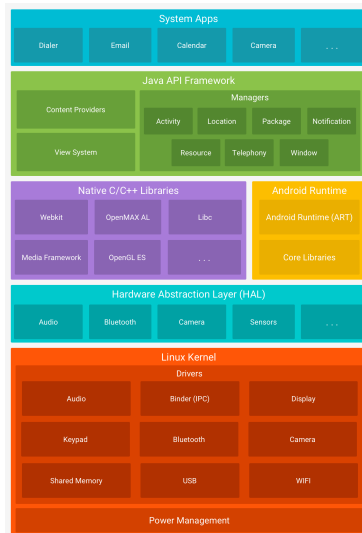
- Si tratta di un sistema operativo (embedded) Linux multi-user
 - ▶ Non è una distribuzione GNU/Linux né un sistema unix-like (tutte le GNU utils di Linux sono sostituite da software Java...)
 - ▶ Ogni applicazione è un *utente* del sistema a cui è associato un ID univoco
 - ▶ Ogni processo del sistema è eseguito mediante una propria virtual machine
- Ciascuna applicazione *vive* in una propria *sandbox* protetta
 - ▶ Ciascuna applicazione è eseguita in un diverso processo di sistema
 - ▶ Il codice di ciascuna applicazione è isolato da quello delle altre
 - ▶ Principio di **Least Privilege**: ogni applicazione ha accesso ai soli componenti esplicitamente richiesti per eseguire i propri compiti

Android OS: Software Stack



Android OS: Software Stack

- **Java API Framework** – Fornisce l'intero feature-set di Android OS, ovvero contiene la definizione dei principali building-blocks.
- **Hardware Abstraction Layer (HAL)** – Fornisce l'interfaccia standard per sfruttare tutte le funzionalità dei dispositivi hardware (sensori, connettività, ...)



Android OS: Versioni e API Level

Codename	Version	API level/NDK release
Android10	10	API level 29
Pie	9	API level 28
Oreo	8.1.0	API level 27
Oreo	8.0.0	API level 26
Nougat	7.1	API level 25
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19
Jelly Bean	4.3.x	API level 18

Android OS: Dalvik Virtual Machine

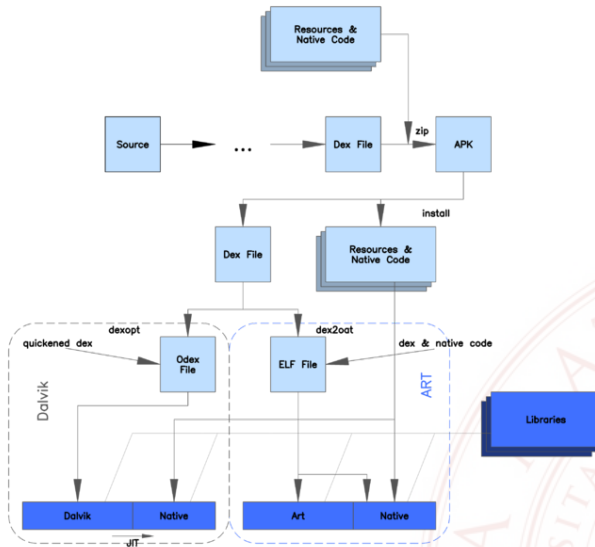
- La **Dalvik VM** è stata la virtual machine di Android fino alla versione 4.x del sistema operativo
 - ▶ Diversamente dalla JVM (che è una stack machine) Dalvik ha un'architettura basata su registri
 - ▶ Richiede meno istruzioni di base, ma più complesse
- Il bytecode accettato da Dalvik è definito in file *.dex che sono il risultato della fase di compilazione del codice Java tramite il compilatore Android (dx).
- Compilazione Just-In-Time (JIT) da Android v. 2.2
 - ▶ Ogni app è compilata solo in parte dallo sviluppatore. A run-time, per ciascuna esecuzione dell'app, sarà l'interprete di Dalvik a compilare definitivamente il codice e ad eseguirlo.

Android OS: ART

- Acronimo di **Android RunTime**, rappresenta il nuovo runtime system di Android.
 - ▶ Da Android 5.0 sostituisce completamente la Dalvik VM.
- Rispetto a Dalvik che è basata su compilazione JIT, ART è basata su tecnologia AOT (ahead-of-time).
 - ▶ L'intera compilazione del codice avviene durante l'installazione dell'app su un device e non durante l'esecuzione della stessa.
 - ▶ Notevoli vantaggi in termini di performance durante l'esecuzione delle applicazioni.
- Garbage Collection (GC) e meccanismi di debugging ottimizzati

» <http://source.android.com/devices/tech/dalvik/index.html>

Android OS: Dalvik vs. ART



Outline

1 Introduzione

2 Android OS

3 Applicazioni Android

- Building Blocks
- Il file Manifest
- Intent



Applicazioni Android: Overview (1/2)

- Linguaggi di Programmazione di riferimento
 - ▶ Kotlin
 - ★ <https://kotlinlang.org/>
 - ★ <https://developer.android.com/kotlin/get-started>
 - ▶ Java (v. 7.0)
 - ★ Alcune funzionalità presenti in Java 8 (e successive versioni) sono supportate solo se si utilizza l'IDE Android Studio (v. 3.0 o successive)
 - ★ <https://developer.android.com/studio/write/java8-support>
- Porzioni di codice possono essere scritte in linguaggio nativo C o C++ tramite JNI
- Compilate tramite l'SDK Android in un file denominato **Android Package** (con estensione **.apk**)
 - ▶ Contiene tutte le risorse necessarie all'applicazione
 - ▶ E' il file mediante il quale è possibile installare l'applicazione sui device con Android OS

Applicazioni Android: Overview (2/2)

- Possono essere progettate per supportare dinamicamente una diversa varietà di dispositivi
 - ▶ Diversi form-factor
 - ▶ Diverse dotazioni in termini di sensori e supporto hardware
 - ▶ ...
- Ciascuna applicazione può essere distribuita previa l'apposizione di un certificato digitale che attesti l'identità dello sviluppatore
 - ▶ **The Android Big Problem** – Negli ultimi 6 anni le applicazioni distribuite sono cresciute di un fattore 5 in termini di dimensione
 - ▶ Dal 2018, **Android App Bundle** consente di distribuire app più leggere (device oriented)

Applicazioni Android: Building Blocks I

- Il comportamento di ciascuna applicazione Android può essere progettato mediante istanze di 4 componenti principali:
 - ▶ **Activity**, **Service**, **Content Provider** e **Broadcast Receiver**
- Ciascun componente possiede un proprio lifecycle predefinito.

1. Activity

- Rappresenta una singola schermata di un'applicazione con associata una propria interfaccia utente.
- L'interfaccia utente di ciascun applicazione è definita mediante uno o più file di risorse (codificati in XML).
- Ciascuna applicazione può definire ed eseguire più activity, una sola delle quali può trovarsi in stato di foreground in un preciso momento.

Applicazioni Android: Building Blocks II

2. Service

- Componente che esegue la propria computazione in background rispetto alle activity.
- Non è associato ad alcuna interfaccia grafica.
- Può prevedere meccanismi per la richiesta di esecuzione di compiti da componenti esterni.

3. Content Provider

- Consente di accedere, gestire e modificare i dati persistenti di una applicazione.
- I dati salvati in un Content Provider possono essere privati per una specifica applicazione o condivisi con le altre.

Applicazioni Android: Building Blocks III

4. Broadcast Receiver

- Componente che permette di catturare e/o rispondere agli annunci (messaggi) propagati all'interno del sistema.
- Un annuncio può riguardare un qualsiasi elemento del sistema e può essere propagato sia dal sistema operativo stesso, sia da una qualunque applicazione. Alcuni esempi:
 - ▶ *Il display del dispositivo è stato spento*
 - ▶ *Un determinato file è stato scaricato ed è ora disponibile*
 - ▶ *È stata identificata una nuova rete WiFi Disponibile*
 - ▶ ...
- Non è associato ad alcuna interfaccia utente ma può attivare una notifica sulla barra di stato.

Building Blocks - Osservazioni

1. Ogni componente è istanziato nell'ambito di una precisa applicazione ma può essere eseguito da qualunque altro componente di qualunque altra applicazione.
 - ▶ Esempio: *Per scattare una foto da una propria applicazione non è necessario implementare un componente specifico ma è sufficiente richiamare il componente (activity) dell'applicazione dedicata che potrà eseguire il compito associato ed informare l'applicazione chiamante quando la foto risulta disponibile.*
2. Ogni componente è eseguito dal sistema all'interno del processo della relativa app.
 - ▶ Qualora si voglia eseguire un componente definito da un'altra app è necessario che sia il sistema a farlo (**Principio di Isolamento delle App**).

Il file Manifest

- Descrive l'applicazione, mediante sintassi XML.
 - ▶ Deve essere obbligatoriamente presente nella root directory dell'applicazione in un file denominato **AndroidManifest.xml**
 - ▶ Dichiarare l'API Level di riferimento (minVersion e targetVersion)
- Elenca tutti i componenti dell'applicazione che il sistema può eseguire
- Elenca i **permessi** che l'applicazione richiede per la propria esecuzione
- Dichiarare i componenti HW e SW che l'applicazione intende utilizzare
- ...

» <http://developer.android.com/guide/topics/manifest/manifest-intro.html>

Il file Manifest: esempio

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="it.unibo.pslab.FirstExample"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="19" android:targetSdkVersion="24" />

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

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name="it.unibo.pslab.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name="it.unibo.pslab.ConfigurationActivity"
            android:label="@string/app_name_conf" />

        <service
            android:name="it.unibo.pslab.MyService"
            android:exported="true" >
        </service>

    </application>
</manifest>
```

Attivazione dei Componenti

- Ciascun componente può essere attivato inviando un messaggio asincrono al sistema chiamato **Intent**.
 - ▶ Meccanismo valido per attivare istanze di Activity, Service e Broadcast Receiver.
- L'Intent definisce l'azione da eseguire (il componente da attivare, il messaggio da propagare, ...)
 - ▶ Può specificare una serie di informazioni aggiuntive (**flag** e **extra**) destinate al sistema e/o al componente da attivare.
- Esistono due diversi tipi di Intent:
 - ▶ **Intent Espliciti**, creati ed invocati attraverso il nome esplicito del componente da attivare.
 - ▶ **Intent Impliciti**, descrivono un'azione generica da eseguire che può essere intercettata da un componente che sia in grado di eseguirla.

Intent: meccanismo di esecuzione

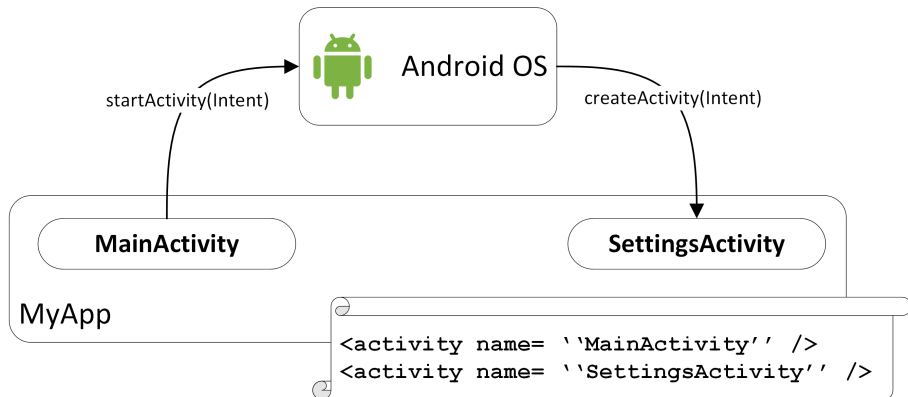
Intent Espliciti

- Dopo la creazione, il sistema avvia immediatamente il componente specificato nell'Intent.

Intent Impliciti

- Il sistema cerca un possibile componente che sia in grado di eseguire l'Intent
 - ▶ Il componente può essere interno all'applicazione oppure scelto tra quelli raggiungibili nell'intero sistema.
- Se viene identificato un unico componente candidato, questo viene eseguito automaticamente.
- Se vengono identificati più componenti candidati, la scelta sul quale eseguire è demandata all'utente.

Intent Espliciti



Intent Espliciti: creazione e utilizzo

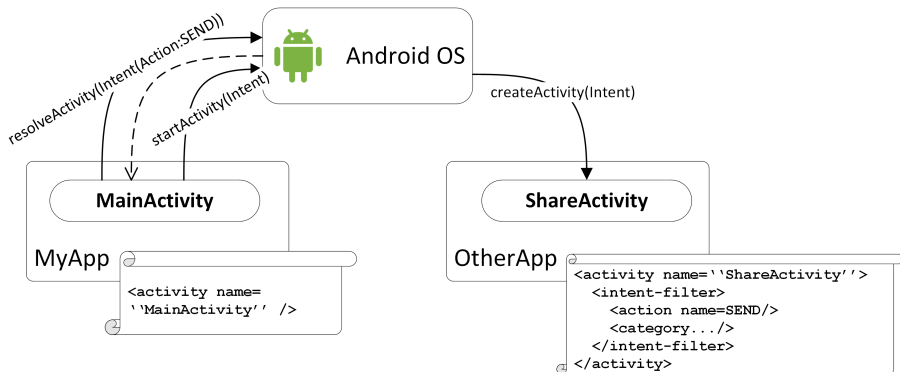
- Deve essere creato ed inizializzato un oggetto della classe `android.content.Intent`
 - ▶ Fornisce diversi costruttori che permettono di configurare opportunamente l'Intent.
 - ▶ Generalmente si utilizza il costruttore che accetta come parametri contesto e indicazione della classe che rappresenta il componente da eseguire: `Intent(Context packageContext, Class<?> cls)`

Esempio

```
Intent i = new Intent(getApplicationContext(), MyActivity.class);  
i.putExtra("USERNAME", "mario.rossi");  
i.putExtra("PASSWORD", "abc123");
```

- L'uso dell'Intent dipende dal componente che si vuole attivare.
 - ▶ Es. Richiamare il metodo `startActivity(i)` per avviare un'activity.

Intent Impliciti



Intent Impliciti: creazione e utilizzo I

- Analogo al caso precedente, senza la necessità di specificare la classe che rappresenta il componente da attivare
 - ▶ Deve essere specificata l'azione associata all'intent (m. `setAction()`).
 - ▶ Deve essere verificata la presenza di almeno un componente (nel sistema) in grado di risolvere l'intent prima di attivarlo (es. metodo `resolveActivity()`).

Esempio

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");

if(sendIntent.resolveActivity(getPackageManager()) != null){
    startActivity(sendIntent);
}
```





Intent Impliciti: creazione e utilizzo II

- Per poter risolvere un'intent, è necessario che (almeno) ad un componente sia data la possibilità di intercettare un particolare tipo di intent.
 - ▶ Deve essere specificato un **Intent Filter** nella dichiarazione del componente nel File Manifest.

Esempio

```
<activity android:name="ShareActivity">  
  
  <intent-filter>  
    <action android:name="android.intent.action.SEND"/>  
    <category android:name="android.intent.category.DEFAULT"/>  
    <data android:mimeType="text/plain"/>  
  </intent-filter>  
  
</activity>
```

Riferimenti - Risorse Online

-  **Android Developers - Guide**
» <https://developer.android.com/guide/>
-  **Android Developers - API Reference**
» <https://developer.android.com/reference/>
-  **Android Developers - Samples**
» <https://developer.android.com/samples/>
-  **Android Developers - Design & Quality**
» <https://developer.android.com/design/>



Riferimenti - Libri

-  Zigurd Mednieks, Laird Dornin, G. Blake Meike, Masumi Nakamura
Programming Android
O'Reilly, 2011
-  Chris Haseman, Kevin Grant
Beginning Android Programming: Develop and Design
Peachpit Press, 2013
-  Ronan Schwarz, Phil Dutson, James Steele, Nelson To
The Android Developer's Cookbook : Building Applications with the Android SDK
Addison-Wesley, 2013
-  Theresa Neil
Mobile Design Pattern Gallery: UI Patterns for Smartphone App
O'Reilly, Second Edition, 2014