



ANDROID

LEZIONE I - 17/12/2015

<https://github.com/giuseppegrosso/android.git>

- ▶ Android è un sistema operativo specializzato per dispositivi mobili
- ▶ Supporta un gran numero di device e fattori di forma variabili.
- ▶ Basato su kernel Linux
- ▶ Android è un progetto open source anche se con qualche “limitazione”
- ▶ E' dotato di una propria virtual machine che consente la scrittura di codice applicativo in linguaggio Java
- ▶ E' il sistema operativo più diffuso al mondo
- ▶ Ecosistema estremamente frammentato

- ▶ Sviluppato a partire dal 2003
- ▶ Acquistato da Google nel 2005 per 50M \$
- ▶ Android è stato annunciato pubblicamente nel 2007.
- ▶ Android è sviluppato dalla OHA, Open Handset Alliance, un raggruppamento di imprese che ha Google come capofila e l'obiettivo di sviluppare standard e tecnologie aperte per i device mobili.
- ▶ Prima beta pubblica nel 2008.
- ▶ Lanciato commercialmente con la versione 1.1 nel 2009



Cupcake
Android 1.5



Donut
Android 1.6



Eclair
Android 2.0/2.1



Froyo
Android 2.2.x



Gingerbread
Android 2.3.x



Honeycomb
Android 3.x



Ice Cream Sandwich
Android 4.0.x



Jelly Bean
Android 4.1/4.2/4.3



KitKat
Android 4.4.x



Lollipop
Android 5.0/5.1



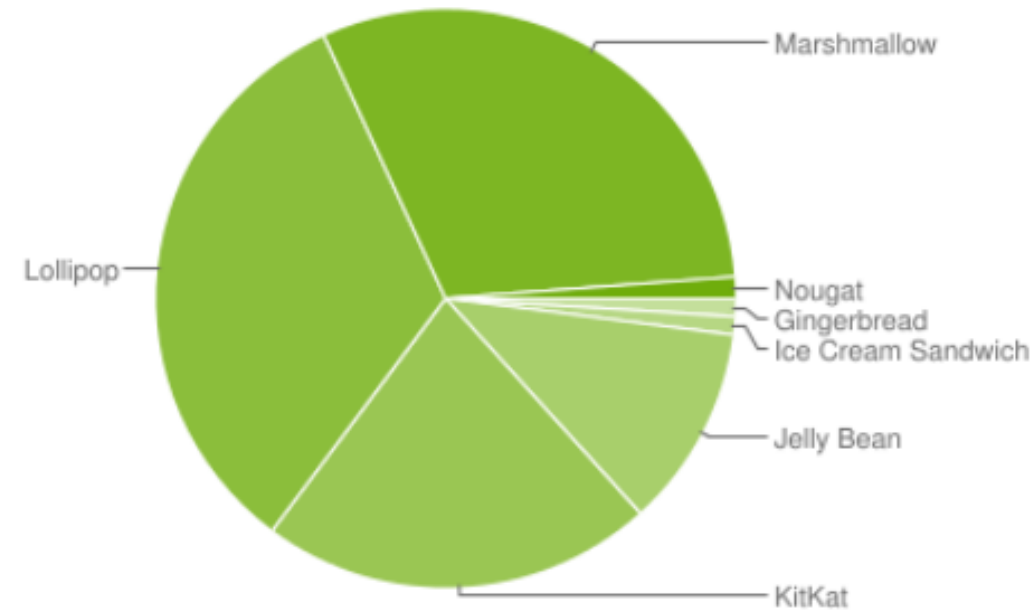
Marshmallow
Android 6.0.x



Nougat
Android 7.0/7.1

For information about how to target your application to devices based on platform version, read [Supporting Different Platform Versions](#).

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.0%
4.1.x	Jelly Bean	16	4.0%
4.2.x		17	5.7%
4.3		18	1.6%
4.4	KitKat	19	21.9%
5.0	Lollipop	21	9.8%
5.1		22	23.1%
6.0	Marshmallow	23	30.7%
7.0	Nougat	24	0.9%
7.1		25	0.3%

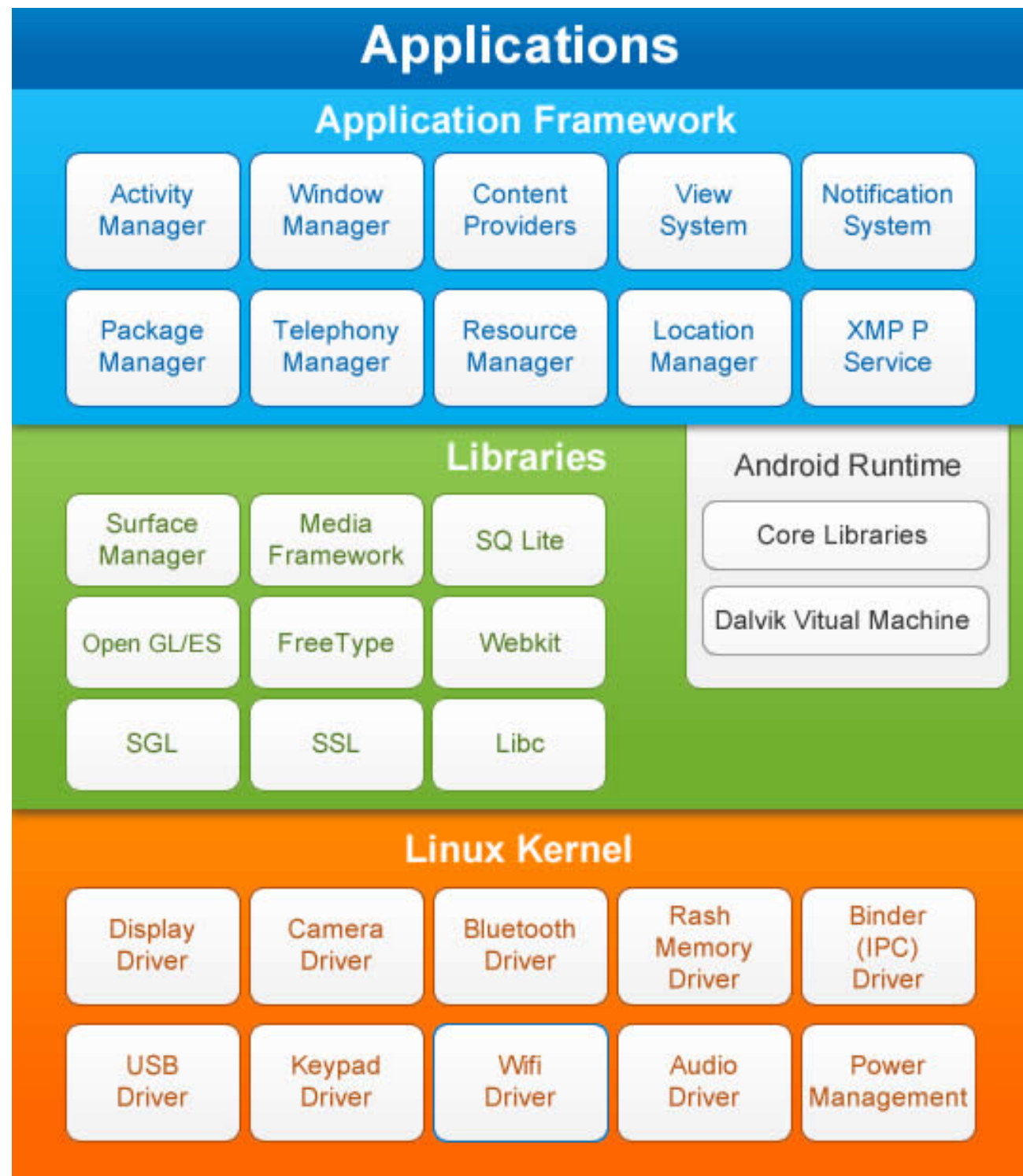


A febbraio 2017 ci sono circa 13 versioni di Android circolanti su un parco di circa 11000 device

FRAMMENTAZIONE

Data collected during a 7-day period ending on February 6, 2017.

Any versions with less than 0.1% distribution are not shown.



App (sia bundled che personali)

API che espongono agli sviluppatori le funzionalità di OS e librerie

Dalvik sostituita da ART da 4.4 in poi

Approccio AOT vs JIT

Attualmente kernel 3.10, inizialmente 2.6.x



SVILUPPARE APP ANDROID

PREREQUISITI e SETUP AMBIENTE


- ▶ Lo sviluppo del codice applicativo avviene in Java per cui è necessaria una conoscenza almeno basilare della sintassi Java (allineata a JDK 6, supportata JDK 7)
- ▶ Fondamentale un'esperienza nella programmazione ad oggetti e dei design pattern più comuni (Singleton, Builder, Observer, Composite, Proxy, Facade)
- ▶ I file che descrivono le risorse, le configurazioni e soprattutto i layout sono scritti in XML per cui è necessario avere familiarità con XML o in generale con linguaggi di markup nested
- ▶ Opzionali competenze C nel caso di sviluppo di componenti con NDK (Native Development Kit) o RenderScript (sintassi C-99 like). Uso molto limitato.

- ▶ Workstation Windows, Mac OS X, o Linux. In questo corso assumiamo una workstation Mac OS X 10.10 o successiva
- ▶ JDK 6 o 7 e configurazione variabili d'ambiente
- ▶ Android Studio o Eclipse + ADT. In questo corso utilizzeremo Android Studio
- ▶ Android SDK, Android Tools e Support Libraries
- ▶ Setup variabile ambiente \$PATH per l'utilizzo di *adb* da *shell*
- ▶ Client GIT (es. Sourcetree per il clone degli esempi di codice)
- ▶ Software basilare image processing (almeno resize e conversioni di formato).

► Android Studio è sviluppato da Google ed è basato su IntelliJ IDEA

 Sync con Gradle. Gestione e risoluzione dipendenze

▼ Gradle Scripts

 **build.gradle** (Project: Lesson1)

 **build.gradle** (Module: app)

 **gradle-wrapper.properties** (Gradle Version)

 **proguard-rules.pro** (ProGuard Rules for app)

 **gradle.properties** (Project Properties)

 **settings.gradle** (Project Settings)

 **local.properties** (SDK Location)

Ogni nuovo progetto è automaticamente configurato per utilizzare Gradle per la gestione dei task e la risoluzione delle dipendenze.

Il progetto (nella sua versione base) è rappresentato da un generico modulo Gradle. Sono possibili configurazioni più complesse.

I file di configurazione sono raggruppati logicamente sotto “Gradle Scripts”.



Sync con Gradle. Gestione e risoluzione dipendenze

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 22
    buildToolsVersion "23.0.1"

    defaultConfig {
        applicationId "it.elbuild.lesson1"
        minSdkVersion 19
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
}
```

- Android Plugin
- Override AndroidManifest.xml
- Supporto per molteplici build types, utile per differenziare ambienti di test, prod e QA.
- Integrazione Proguard (minificazione e obfuscamento codice APK release)



Sync con Gradle. Gestione e risoluzione dipendenze

```
dependencies {  
    compile fileTree(dir: 'libs', include: ['*.jar'])  
    testCompile 'junit:junit:4.12'  
    compile 'com.android.support:appcompat-v7:22.2.1'  
    compile 'com.android.support:design:22.2.1'  
}
```

- ▶ Gestione dipendenze da molteplici fonti.
- ▶ Supporto per dipendenze a compile e runtime, anche separate per sessioni di test
- ▶ Supporto librerie esterne, *JAR*, *jCenter* e repository Maven personali.



Android SDK Manager. Gestione e aggiornamento tool di sviluppo

Appearance & Behavior > System Settings > Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location:

[Edit](#)

SDK Platforms

SDK Tools

SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

Name	API Level	Revision	Status
<input checked="" type="checkbox"/> Android 6.0	23	2	Installed
<input checked="" type="checkbox"/> Android 5.1.1	22	2	Update available
<input checked="" type="checkbox"/> Android 5.0.1	21	2	Update available
<input checked="" type="checkbox"/> Android L Preview	L	3	Partially installed
<input checked="" type="checkbox"/> Android 4.4W.2	20	2	Installed
<input checked="" type="checkbox"/> Android 4.4.2	19	4	Update available
<input checked="" type="checkbox"/> Android 4.3.1	18	3	Installed
<input checked="" type="checkbox"/> Android 4.2.2	17	3	Installed
<input checked="" type="checkbox"/> Android 4.1.2	16	5	Installed
<input checked="" type="checkbox"/> Android 4.0.3	15	5	Installed
<input checked="" type="checkbox"/> Android 4.0	14	4	Installed
<input checked="" type="checkbox"/> Android 3.2	13	1	Partially installed
<input checked="" type="checkbox"/> Android 3.1	12	3	Partially installed
<input checked="" type="checkbox"/> Android 3.0	11	2	Partially installed
<input checked="" type="checkbox"/> Android 2.3.3	10	2	Partially installed
<input checked="" type="checkbox"/> Android 2.3.1	9	2	Partially installed
<input checked="" type="checkbox"/> Android 2.2	8	3	Partially installed

☐ Show Package Details

[Launch Standalone SDK Manager](#)

Preview packages available! [Switch](#) to Preview Channel to see them


Cancel

Apply

OK



Android Virtual Device Manager. Simulatori, lenti ma utili in alcuni casi.



Select Hardware

Choose a device definition

Category

Phone

Tablet

Wear

TV

Search

Name	Size	Resolution	Density
Nexus S	4,0"	480x800	hdpi
Nexus One	3,7"	480x800	hdpi
Nexus 6P	5,7"	1440x2560	560dpi
Nexus 6	5,96"	1440x2560	560dpi
Nexus 5X	5,2"	1080x1920	420dpi
Nexus 5	4,95"	1080x1920	xxhdpi
Nexus 4	4,7"	768x1280	xhdpi
Galaxy Nexus	4,65"	720x1280	xhdpi
Android Wear Square	1,65"	280x280	hdpi
Android Wear Round	1,65"	320x320	hdpi
5.4" FWVGA	5,4"	480x854	mdpi
5.1" WVGA	5,1"	480x800	mdpi
4.7" WXGA	4,7"	720x1280	xhdpi
4.65" 720p (Galaxy Nexus)	4,65"	720x1280	xhdpi

New Hardware Profile

Import Hardware Profiles

Nexus 5

1080px

4,95"

1920px

Size: normal

Ratio: notlong

Density: xxhdpi

Clone Device...

Cancel

Previous


Next

Finish

Release Name	API Level	ABI	Target
Lollipop	22	armeabi-v7a	Android 5.1 (with Google APIs)
Lollipop	22	x86	Android 5.1 (with Google APIs)
Lollipop	22	x86_64	Android 5.1 (with Google APIs)
Lollipop	22	armeabi-v7a	Android 5.1
Lollipop	22	x86	Android 5.1
Lollipop	22	x86_64	Android 5.1
Lollipop	21	armeabi-v7a	Android 5.0 (with Google APIs)
Lollipop	21	x86	Android 5.0 (with Google APIs)
Lollipop	21	x86_64	Android 5.0 (with Google APIs)
L	L	armeabi-v7a	Android 5.0
L	L	x86	Android 5.0
Lollipop	21	armeabi-v7a	Android 5.0
Lollipop	21	x86	Android 5.0
Lollipop	21	x86_64	Android 5.0
KitKat	19	armeabi-v7a	Android 4.4
KitKat	19	x86	Android 4.4
KitKat	19	armeabi-v7a	Android 4.4
Jelly Bean	18	armeabi-v7a	Android 4.3
Jelly Bean	18	armeabi-v7a	Android 4.3
Jelly Bean	17	armeabi-v7a	Android 4.2

☐ Show downloadable system images

Lollipop



API Level

22

Android

5.1

Android Open Source Project

System Image

x86

Recommendation

Newer HAXM Version Available

Consider using a system image with Google APIs to enable testing with Google Play

[Download and install HAXM](#)

Questions on API level?

See the [API level distribution chart](#)

Cancel

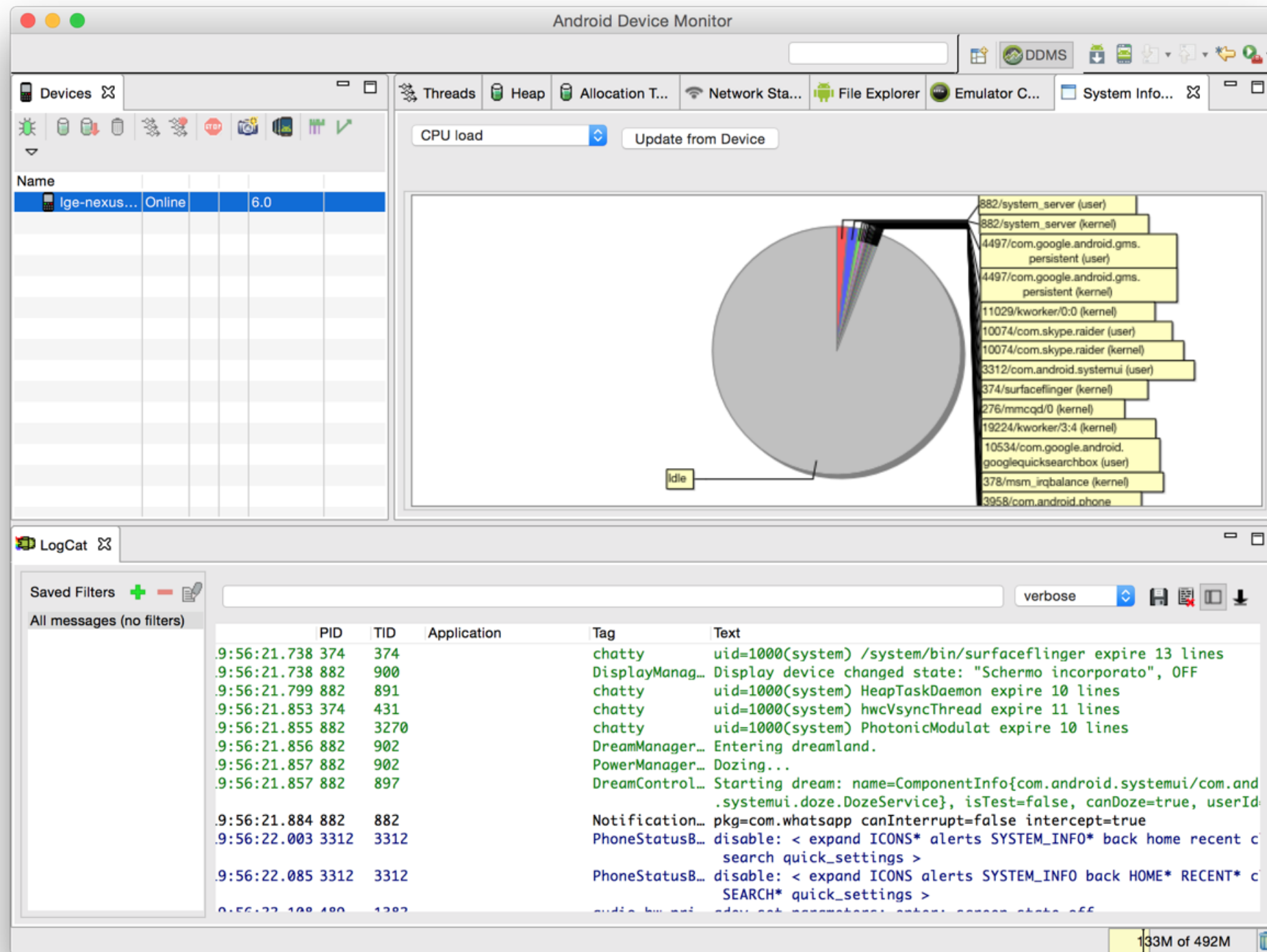
Previous

Next

Finish

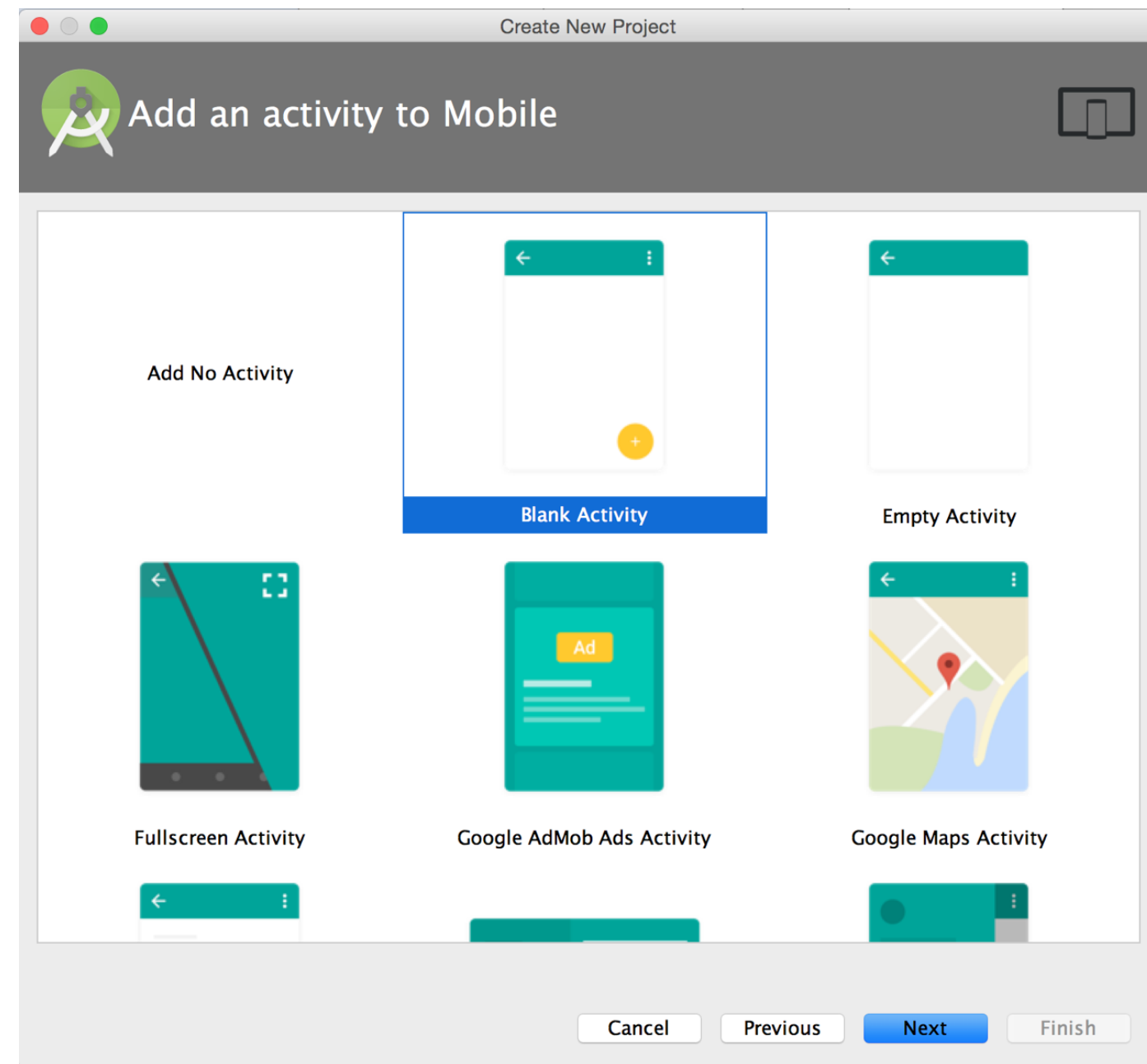
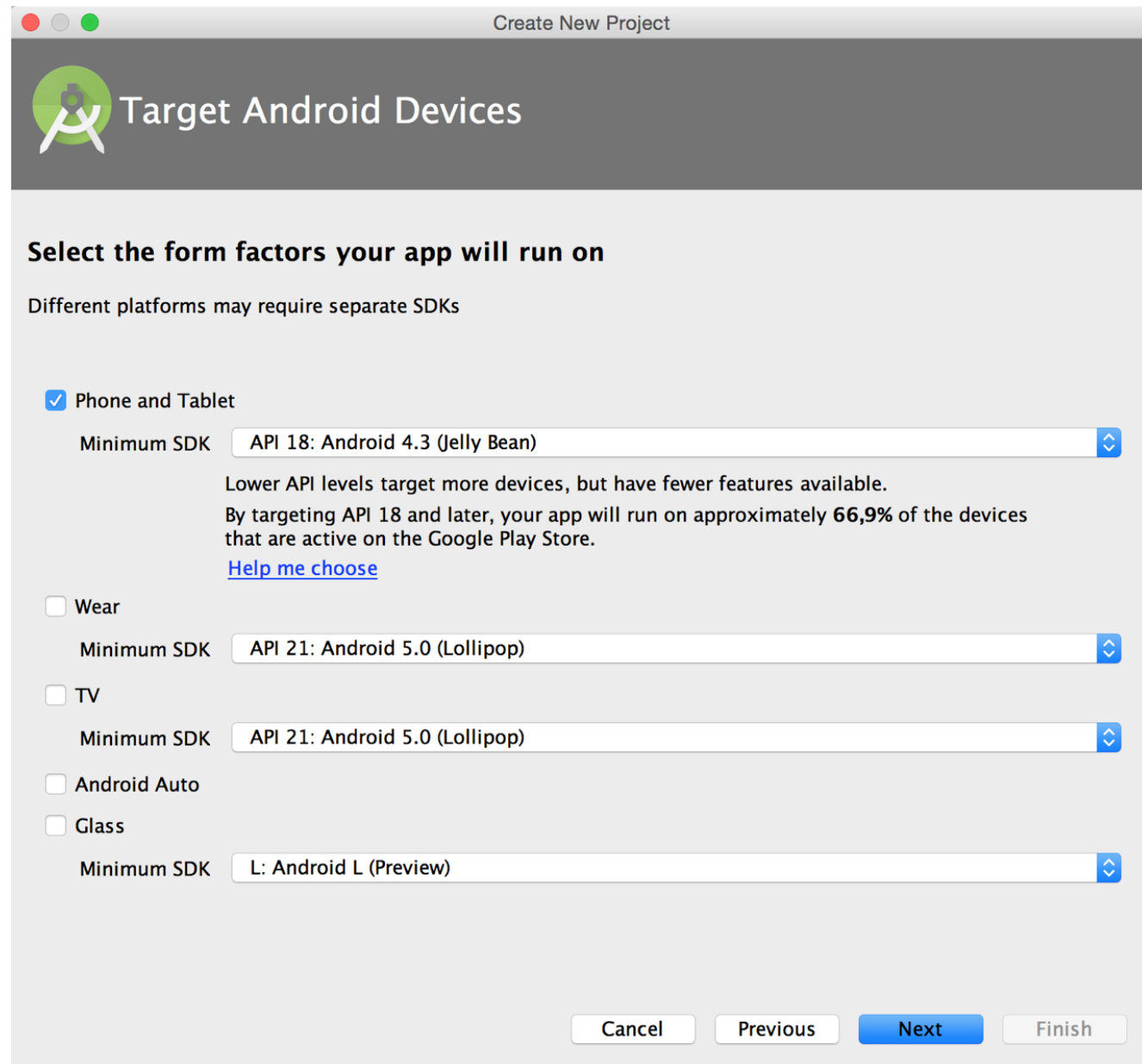


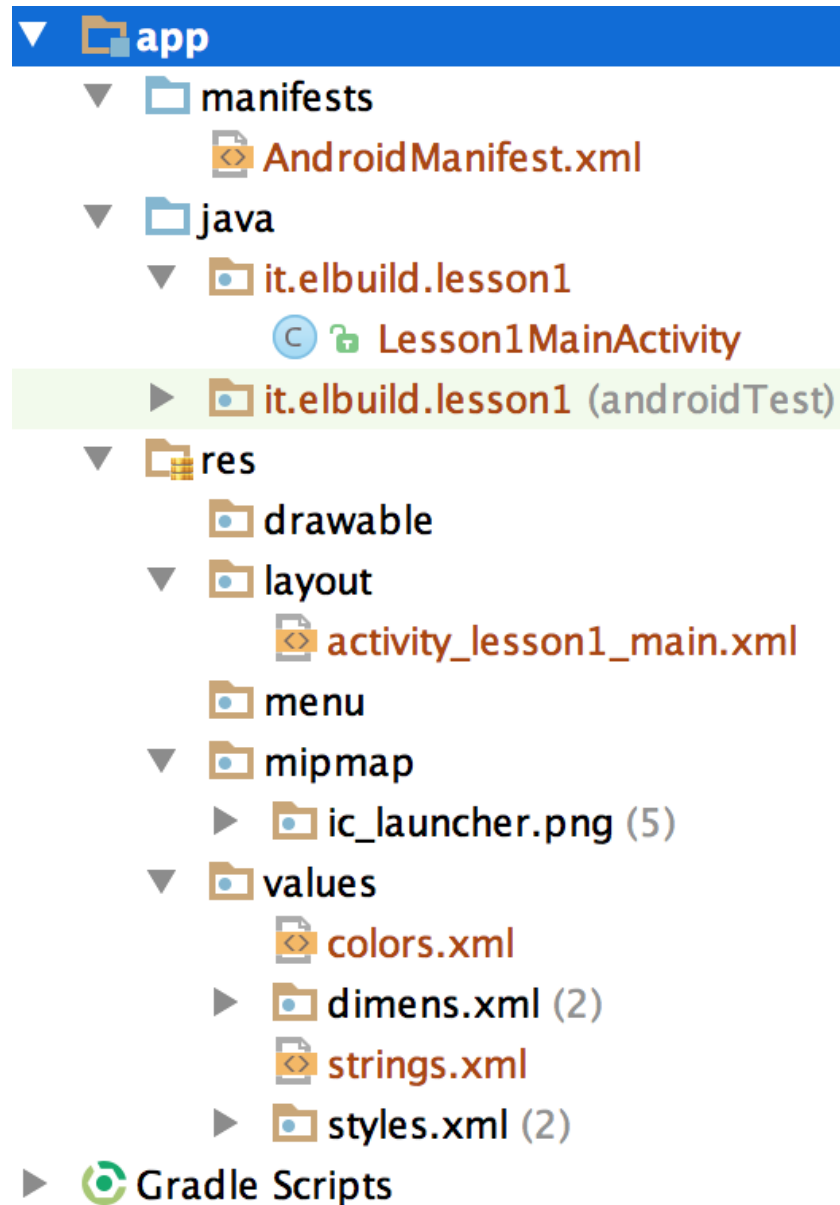
Android Device Monitor. Debug con device fisico





SVILUPPARE APP ANDROID STRUTTURA DI UN PROGETTO





► *manifests* contiene il file AndroidManifest.xml

► *java* contiene il sorgente, organizzato in package

► *res/drawable* contiene risorse grafiche organizzate per adattarsi a schermi e risoluzioni diverse

► *res/layout* e *res/menu* contengono risorse XML per la definizione della UI

► *res/values* contiene stringhe, costanti, definizioni in file XML

► *mipmap* contiene icona app

STRUTTURA PROGETTO - AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest>
```

```
    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />
```

```
</manifest>
```

```
    <application ... />
```

```
<application>
```

```
    <activity>
```

```
        <intent-filter>
            <action />
            <category />
            <data />
        </intent-filter>
        <meta-data />
```

```
    </activity>
```

```
    <service>
```

```
        <intent-filter> . . . <
        <meta-data/>
```

```
    </service>
```

```
    <receiver>
```

```
        <intent-filter> . . . <
        <meta-data />
```

```
    </receiver>
```

```
    <provider>
```

```
        <grant-uri-permission />
        <meta-data />
        <path-permission />
```

```
    </provider>
```

```
    <uses-library />
```

```
</application>
```

```
<?xml version="1.0" encoding="utf-8"?>

<manifest>

    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />

    <application ...

</manifest>
```

► *uses-permission* elenca i permessi che la app richiede all'utente per operare correttamente

► *permission*, *permission-tree* servono a stabilire e gestire i permessi di accesso alla app da parte di componenti terze. Use case non frequenti

► *permission-group* serve a raggruppare logicamente le permission quando vengono presentate all'utente

vedi a lato .../>

► *instrumentation* è utilizzato per test automatizzati

► Gli altri tag child del tag *manifest* servono a specificare dipendenze della app verso configurazioni o dotazioni hardware in modo da impedire l'installazione via Play Store a quei device che non sono compliant

<application>

<activity>

</activity>

<service>

</service>

<receiver>

</receiver>

<provider>

</provider>

<uses-library

</application>

► **activity** è un elemento che compare N volte, uno per ogni Activity che compone l'app. Rimanda alla classe Java che implementa la specifica Activity e specifica opzionalmente gli Intent che la lanciano.

► **service** è un elemento che compare N volte, uno per ogni Service che compone l'app. Rimanda alla classe Java che implementa lo specifico Service.

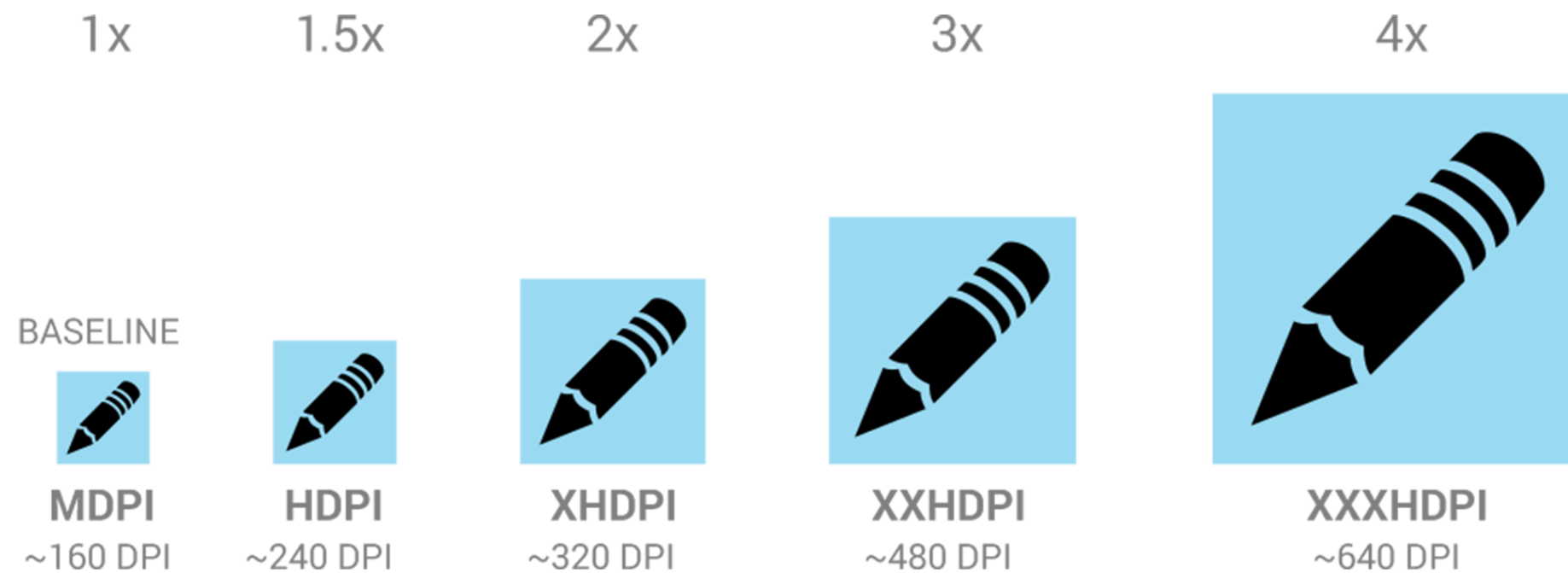
► **receiver** è una sorta di listener globale che può essere registrato per gestire eventi in ingresso alla app, generati dal sistema operativo (sotto forma di Intent). Se ne possono registrare N, sempre specificando la classe Java da invocare.

► **provider** sono utilizzati per accedere a DB SQLite e condividere contenuto con app terze.

Un **drawable** è un concetto generico che rappresenta un qualsiasi elemento grafico che può essere disegnato sulla UI di una app Android. Un drawable può essere costituito da:

- ▶ Un'immagine in formato .png, .jpg o .gif (*BitmapDrawable*)
- ▶ Un file *Nine-Patch* .9.png che rappresenta una generica area con regioni *stretchable* (*NinePatchDrawable* utile per sfondi particolari da adattare ad elementi a dimensione variabile)
- ▶ Una *shape* descritta in un file XML (*ShapeDrawable*)
- ▶ Una combinazione secondo una logica layered o meno di più drawable come definiti nei tre punti sopra.

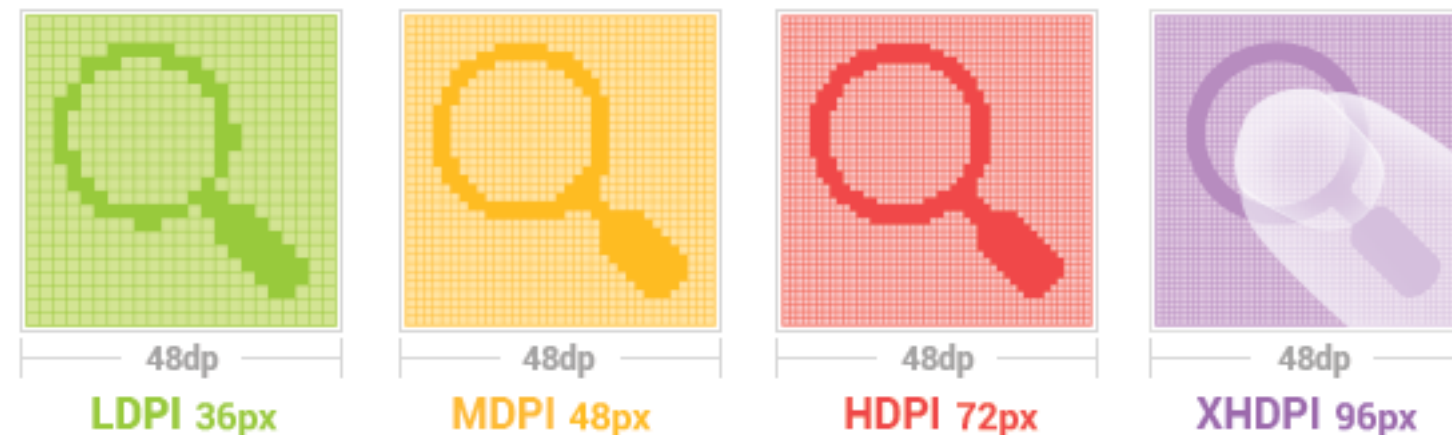
La cartella *res/drawable* contiene i **drawable**. Per supportare molteplici dimensioni di schermo e molteplici densità di *pixel* Android introduce il concetto di *dip* o *Density Independent Pixel*.



La stessa risorsa nei vari formati va inserita in una cartella **drawable-{res}** dove {res} è uno fra *ldpi* (obsoleto), *mdpi*, *hdpi*, *xhdpi*, *xxhdpi*, *xxxhdpi*.

Tipicamente il grafico esporta la risoluzione più alta ed esistono tool che automano l'importazione dei vari drawable nei formati ridotti da parte dello sviluppatore.

Lo sviluppatore pensa e sviluppa solo in termini di **dip** (o **dp**), mai di **pixel**!



E' anche possibile inserire in una cartella *drawable* (senza suffisso) una risorsa xxxhdpi e utilizzare la stessa in ogni layout, ma è una strategia molto subottima (ok per immagini piccole, icone o simili).



SVILUPPARE APP ANDROID I PRIMI PROGETTI



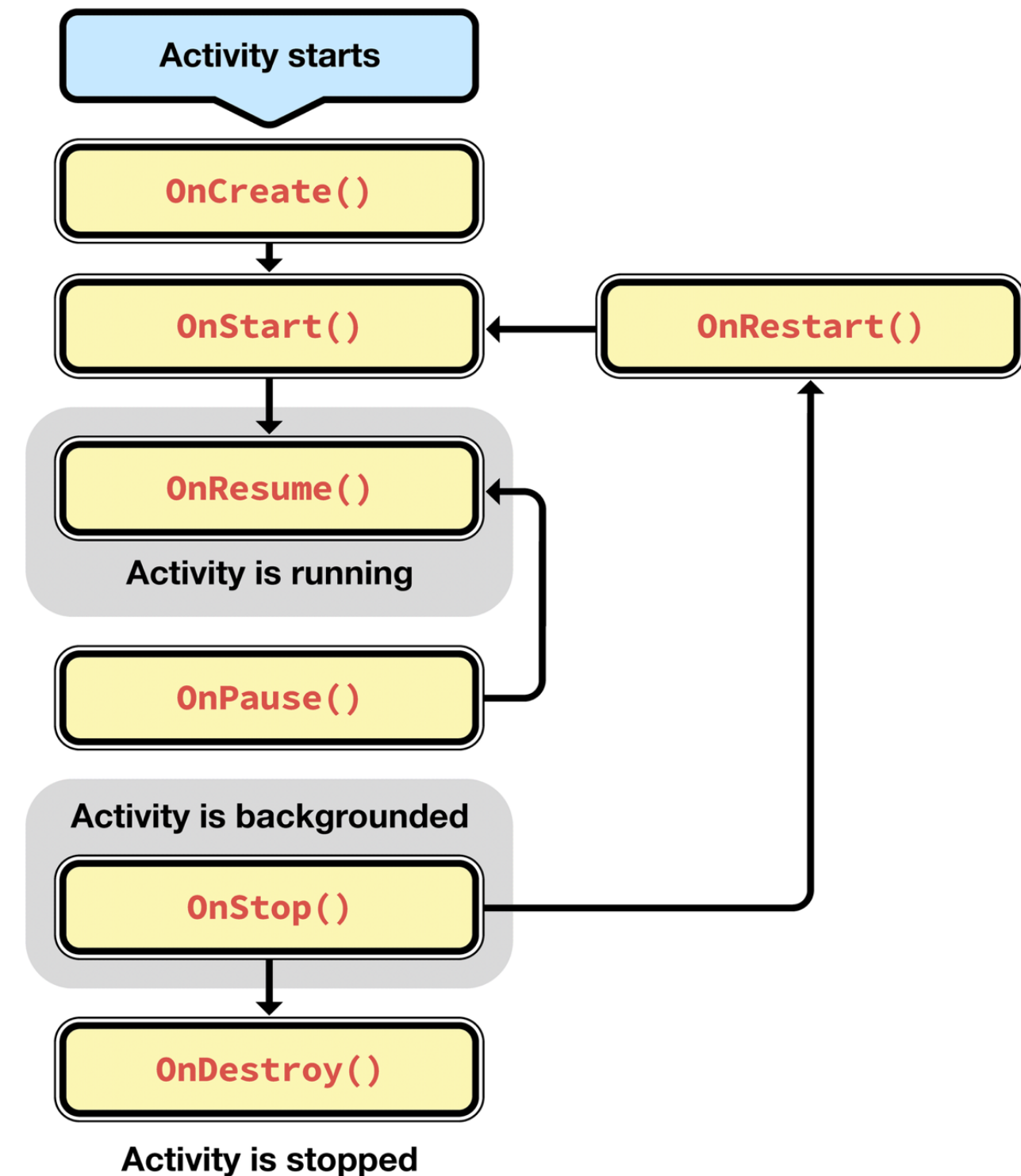
ActivityLifeCycle

<https://github.com/giusepppegrosso/android.git>

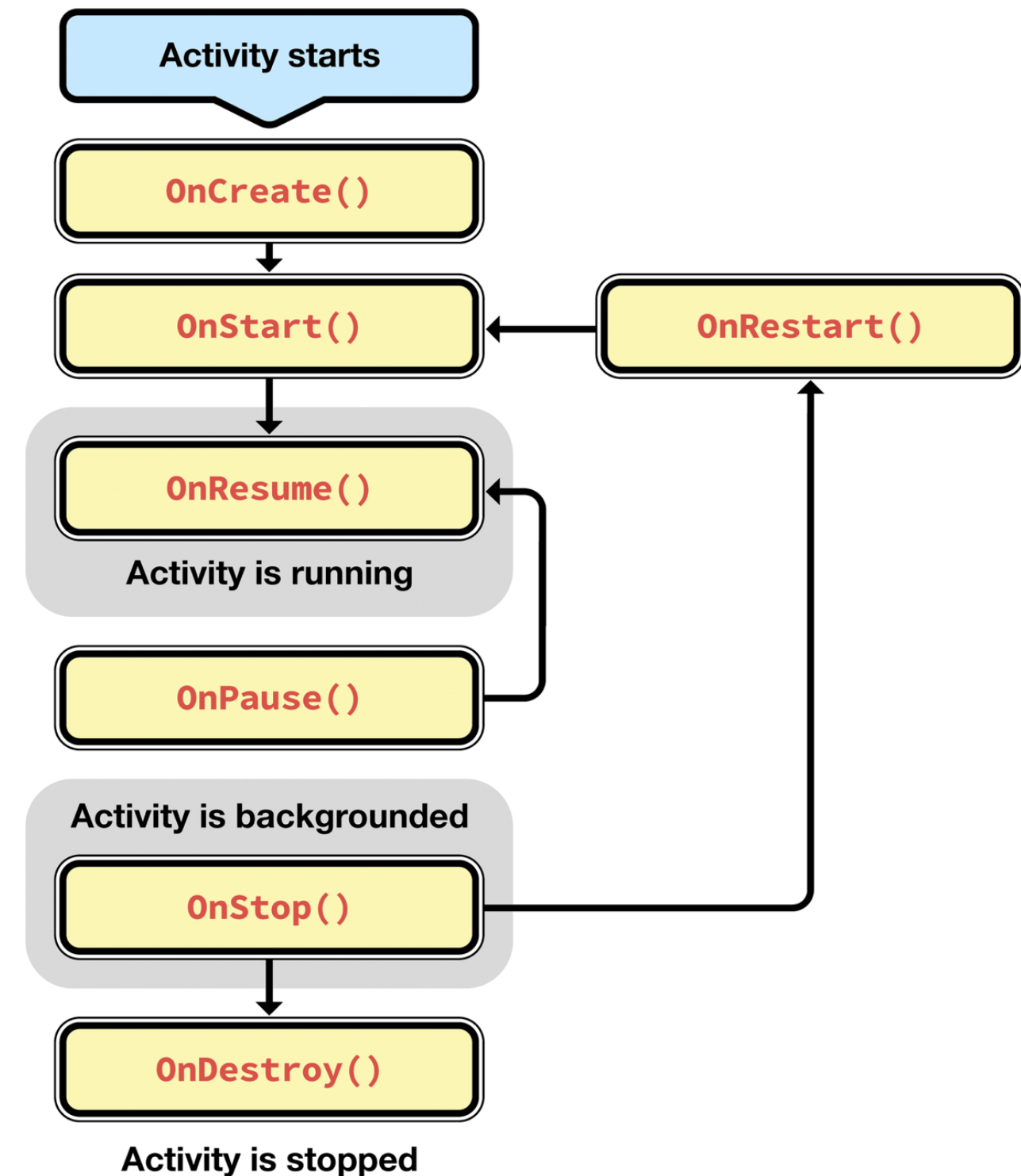
Lo scopo di questa app è prendere familiarità con il concetto di *life cycle* di una *Activity Android*.

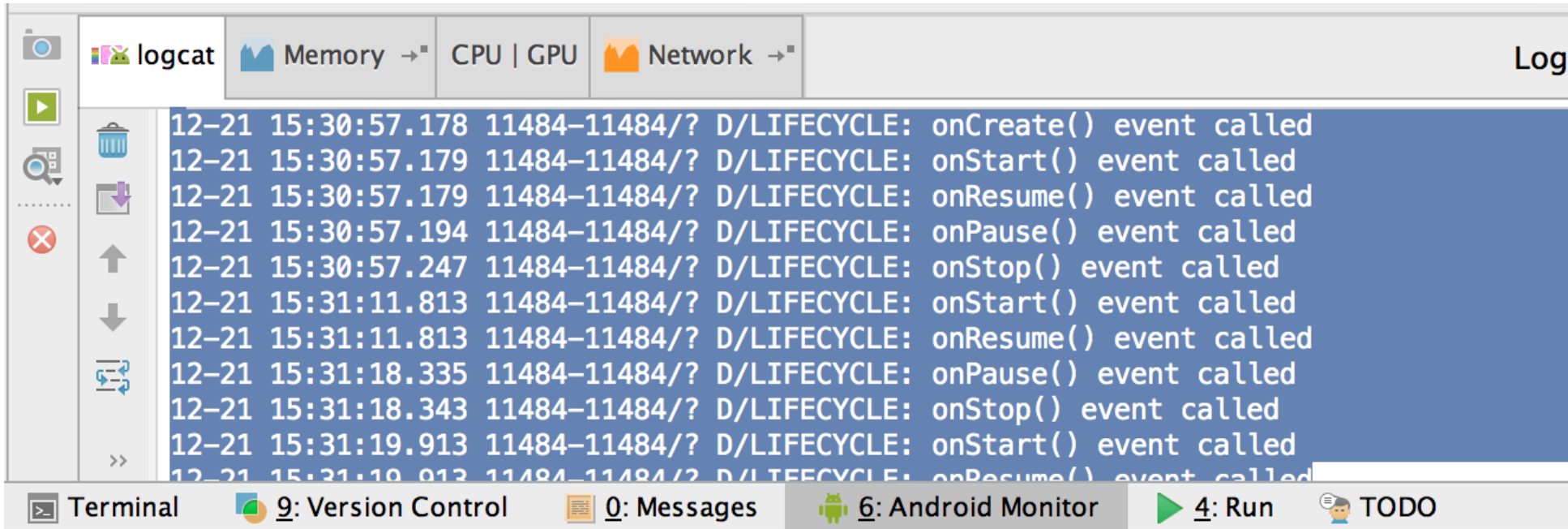
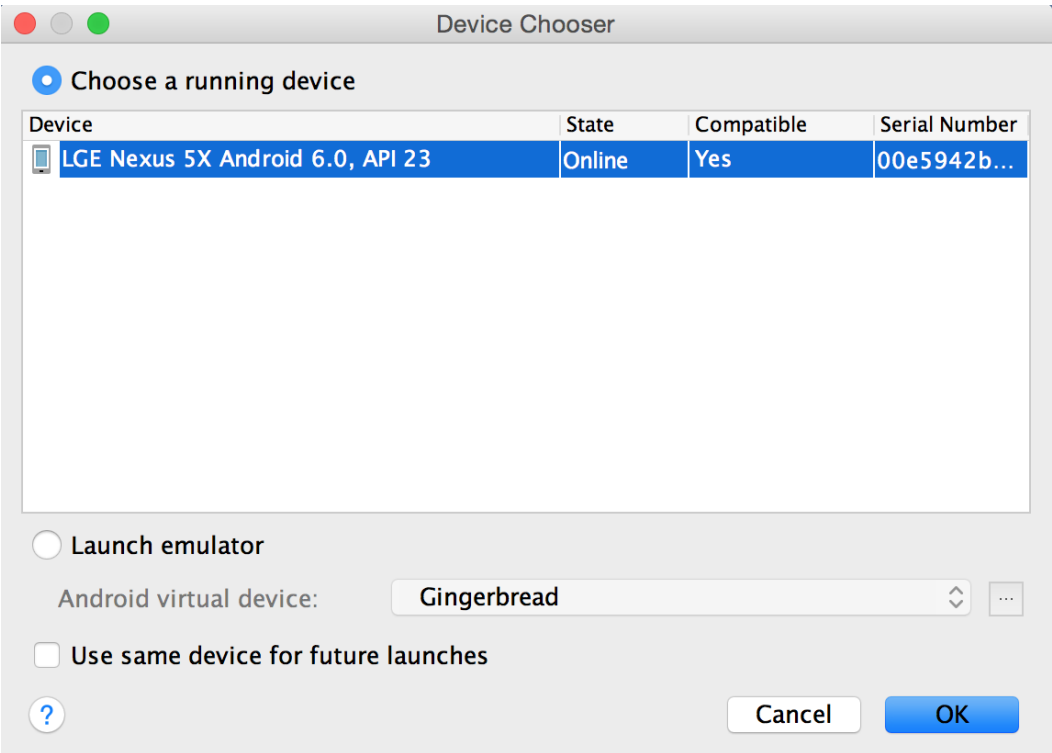
```
public class MainActivity extends Activity {  
  
    private static final String TAG = "LIFECYCLE";  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Log.d(TAG, "onCreate() event called");  
    }  
  
    .... other methods here  
  
}
```

Esaminiamo output con LOGCAT nelle diverse situazioni.



- **onCreate**: l'activity viene creata. Il programmatore deve assegnare le configurazioni di base e definire quale sarà il layout dell'interfaccia;
- **onStart**: l'activity diventa visibile. È il momento in cui si possono attivare funzionalità e servizi che devono offrire informazioni all'utente
- **onResume**: l'activity diventa la destinataria di tutti gli input dell'utente.
- **onPause** (l'inverso di onResume) notifica la cessata interazione dell'utente con l'activity;
- **onStop** (contraltare di onStart) segna la fine della visibilità dell'activity;
- **onDestroy** (contrapposto a onCreate) segna la distruzione dell'activity.







<https://github.com/giuseppegrosso/android.git>