

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

Android SDK, Tools e Android Studio IDE

Angelo Croatti
a.croatti@unibo.it

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

Outline

1 Android SDK

- ADB
- DDMS
- Logging

2 Android IDE

- Android Studio
- AVD



Outline

1 Android SDK

- ADB
- DDMS
- Logging

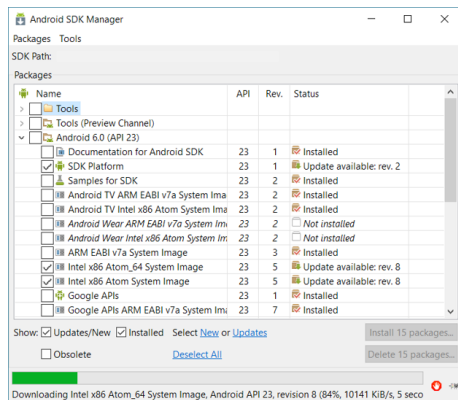
2 Android IDE

- Android Studio
- AVD



Android SDK

- Contiene l'Android Framework, le Native Libraries e tutti i tool a supporto dello sviluppo di Android App
- Mediante l'**Android SDK Manager** è possibile aggiornare l'SDK, scaricare componenti aggiuntivi o extra
 - ▶ La gestione dei componenti è basata sull'API Level



Android Device Bridge (adb)

- Strumento dell'SDK indispensabile per lo sviluppo efficace di applicazioni Android
 - ▶ Consente di comunicare con l'istanza dell'emulatore o con un device fisico connesso all'ambiente di sviluppo (IDE)
- Composto da tre componenti principali:
 - ▶ Un **client**, che è eseguito nella macchina su cui si sviluppa l'applicazione, con il quale si può interagire.
 - ▶ Un **server**, che è eseguito come processo in background sulla macchina su cui si sviluppa l'applicazione, che interagisce con il device (reale o emulato).
 - ▶ Un **demone**, eseguito come processo in background sul device (reale o emulato).
- Può essere utilizzato stand-alone oppure tramite l'IDE

» <https://developer.android.com/studio/command-line/adb>

Android DDMS

- Fornisce strumenti per il monitoraggio e l'interazione con le applicazioni in esecuzione su un device Android (reale o emulato) collegato all'IDE.
 - ▶ Consente di ottenere la lista di device connessi e delle relative applicazioni in esecuzione
- Indispensabile per il debugging
- Per ciascuna applicazione consente di:
 - ▶ Ottenere tutte le informazioni sull'ambiente di esecuzione (la relativa VM)
 - ▶ Ottenere informazioni sui thread attivi (state dumping)
 - ▶ Esplorare il file system (la porzione accessibile)
 - ▶ Catturare screenshots e ottenere informazioni dinamiche sui componenti dell'interfaccia utente
 - ▶ Analizzare i flussi di logging (tool **LogCat**).

Logging in Android I

- Il tool **LogCat** consente di analizzare i flussi di logging di un qualunque device collegato all'IDE (reale o simulato) e quelli di una specifica applicazione in esecuzione.
- Nel codice sorgente è possibile istruire il sistema affinché aggiunga messaggi al flusso di logging.
- Si può far riferimento alla classe **android.util.Log**, in particolare ai seguenti metodi statici:
 - ▶ **Log.v(String tag, String message)** – Livello VERBOSE
 - ▶ **Log.d(String tag, String message)** – Livello DEBUG
 - ▶ **Log.i(String tag, String message)** – Livello INFO
 - ▶ **Log.w(String tag, String message)** – Livello WARNING
 - ▶ **Log.e(String tag, String message)** – Livello ERROR

Logging in Android II

Esempio - Aggiunta di un messaggio al flusso di Logging

```
Log.d(C.LOG_TAG, "My message content");  
  
/* ... */  
  
public class C {  
    public static final String LOG_TAG = "MyAppTag";  
}
```

- L'indicazione di un TAG può essere utile per filtrare il flusso di logging e rintracciare i soli messaggi d'interesse.
- Nota — Qualora in Android si tenti di accedere al Java StdOut – es. attraverso `System.out.println(String msg)` – il messaggio viene rediretto sul flusso di logging, senza tag, con livello verbose.

Outline

1 Android SDK

- ADB
- DDMS
- Logging

2 Android IDE

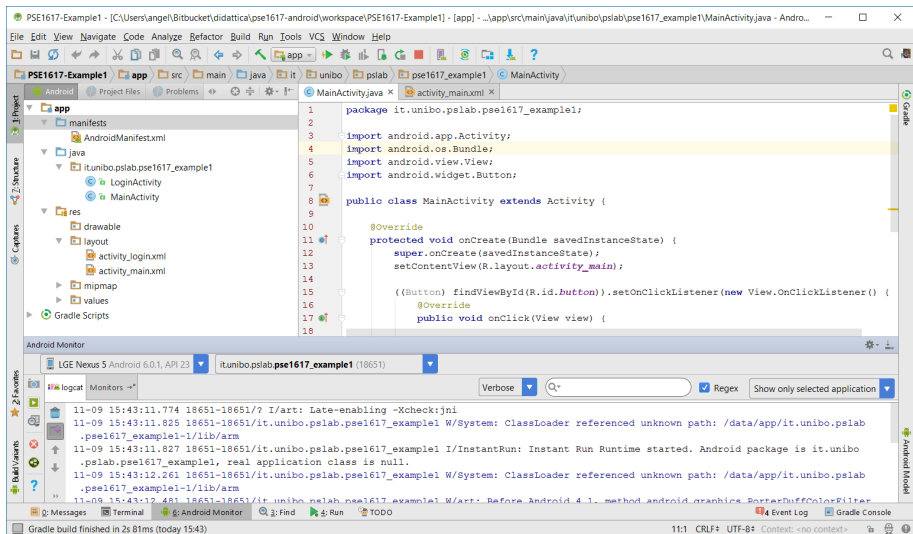
- Android Studio
- AVD



Android Studio I

- IDE ufficiale per lo sviluppo di Android App
 - ▶ Costruito come estensione dell'IDE **IntelliJ IDEA**
 - ▶ Disponibile gratuitamente (licenza Apache 2.0) per tutte le piattaforme (Windows, Mac, Linux)
 - ▶ » <https://developer.android.com/studio/>
- Build System per le applicazioni basato su **Gradle**
 - ▶ <https://gradle.org/>
- Supporta l'**Instant Run**
- Gestione integrata dei repository **GitHub**
- Rapido accesso ai tool della **Google Cloud Platform**

Android Studio II



Android Manifest e Gradle Building

- In Android Studio, alcune dichiarazioni relative all'applicazione che dovrebbero essere inserite nel file Manifest, sono state spostate nel file **build.gradle** per agevolare il processo di building.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="it.unibo.ifts16.ifts16testapp" >

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportRtl="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service
            android:name=".MyService"
            android:exported="true" >
        </service>
    </application>
</manifest>
```

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

android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"

    defaultConfig {
        applicationId "it.unibo.ifts16.ifts16testapp"
        minSdkVersion 19
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'
}
```

Supporto a Java 8 in Android Studio

- Per abilitare il supporto alle feature di Java 8 in Android Studio è necessario impostare nel file `build.gradle` del progetto le `compileOptions` opportune:

```
android {  
    ...  
  
    compileOptions {  
        sourceCompatibility JavaVersion.VERSION_1_8  
        targetCompatibility JavaVersion.VERSION_1_8  
    }  
}
```

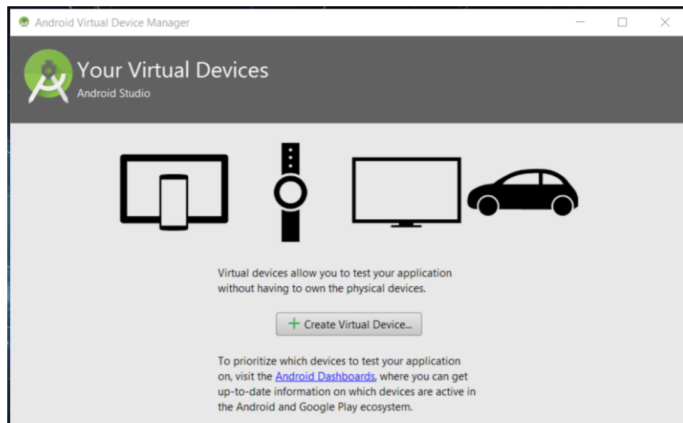
- Dalla UI: File > Project Structure... > (module) > impostare *sourceCompatibility* e *targetCompatibility*

Android Virtual Device (AVD)

- Un AVD costituisce un emulatore per dispositivi (smartphone/tablet) Android-based
 - ▶ Possono essere creati emulatori capaci di simulare la maggior parte delle caratteristiche HW e SW di qualunque device android-based reale
- Sull'emulatore possono essere eseguite (e sottoposte a debug) le applicazioni Android senza utilizzare un device fisico
 - ▶ Non possono essere emulate tutte caratteristiche di un device fisico (es. accesso ai sensori, comunicazione via bluetooth, ...)
- In Android Studio, gli emulatori possono essere creati ed utilizzati mediante lo strumento **AVD Manager**
 - ▶ Il DDMS tratta le applicazioni in esecuzione su un AVD alla stregua di quelle in esecuzione su un device fisico

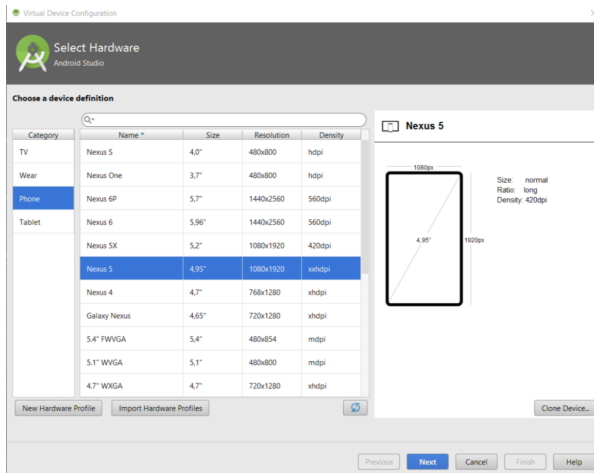
Creazione di un AVD I

1. (in Android Studio) Menù Tools > Android > AVD Manager
2. Click su “Create Virtual Device...”



Creazione di un AVD II

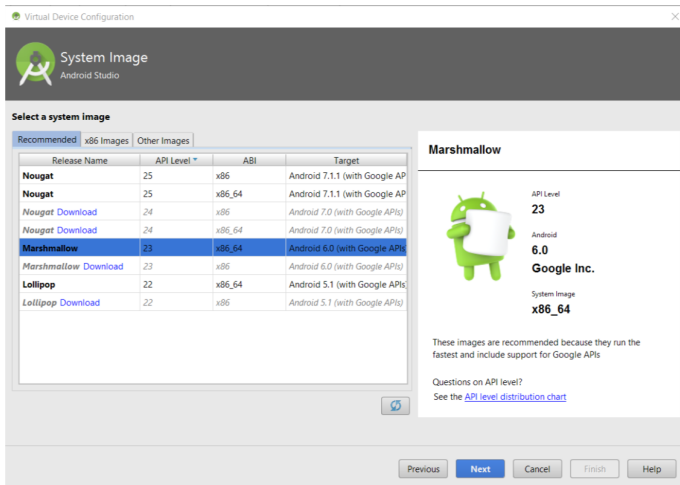
3. Scegliere l'hardware di riferimento da emulare



Creazione di un AVD III

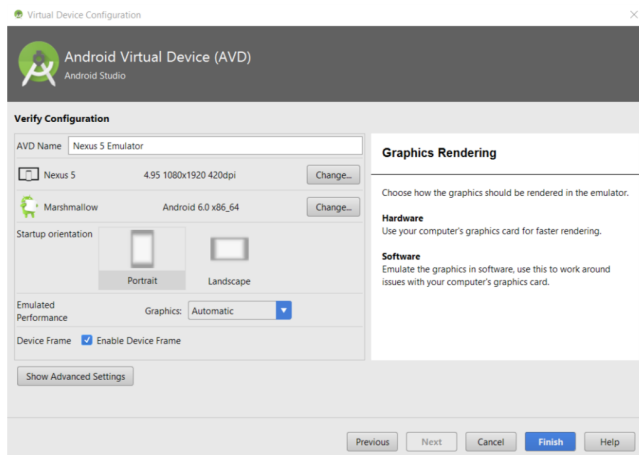
4. Selezionare l'immagine di sistema da associare all'emulatore
 - ▶ Preferire le immagini di sistema proposte nella sezione "Recommended", che fanno riferimento alle immagini di sistema basate sulle Google APIs.
 - ▶ Possono essere aggiornate/scaricate tramite l'Android SDK Manager

Creazione di un AVD IV



Creazione di un AVD V

5. Associare un nome all'emulatore e configurare le impostazioni avanzate ("Show Advanced Settings")



Creazione di un AVD VI

6. Configurazioni Avanzate

- ▶ **Emulated Performance** – Specifica se utilizzare o meno il supporto della GPU (se presente). Si tende a lasciare l'impostazione su "Automatic".
- ▶ **Camera** – Può essere associata all'emulatore una webcam installata nel sistema, da utilizzare come camera per l'emulatore.
- ▶ **Memory and Storage** – Consente di specificare la quantità di RAM da riservare all'emulatore, la quantità di storage interno previsto per il device emulato e, eventualmente, l'uso di una SDCard emulata o agganciata ad un file nel file system guest.
- ▶ **Device Frame** – Consente di personalizzare la skin dell'emulatore
- ▶ **Keyboard** – Abilita l'utilizzo della tastiera hardware collegata al sistema guest all'interno dell'emulatore

Creazione di un AVD VII

Camera Front: Emulated ▾

Back: Emulated ▾

Network Speed: Full ▾

Latency: None ▾

Emulated Performance Graphics: Automatic ▾

☐ Multi-Core CPU 1 ▾ (Experimental)

Memory and Storage RAM: 768 MB ▾

VM heap: 64 MB ▾

Internal Storage: 800 MB ▾

SD card: ☒ Studio-managed 100 MB ▾

☐ External file ...

Device Frame ☒ Enable Device Frame

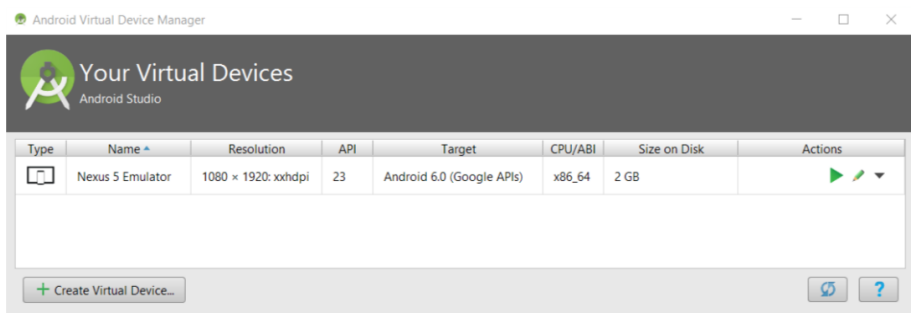
Custom skin definition nexus_5 ▾ ...

[How do I create a custom hardware skin?](#)

Keyboard ☒ Enable keyboard input

Avvio dell'AVD I

- Al termine della creazione dell'emulatore, è possibile avviare l'AVD attraverso il bottone di Launch
 - ▶ Il completo avvio dell'emulatore può richiedere fino a qualche minuto (i tempi dipendono dalle prestazioni dell'hardware guest e dalla possibilità o meno di utilizzare il supporto della GPU)



Avvio dell'AVD II

– NOTA –

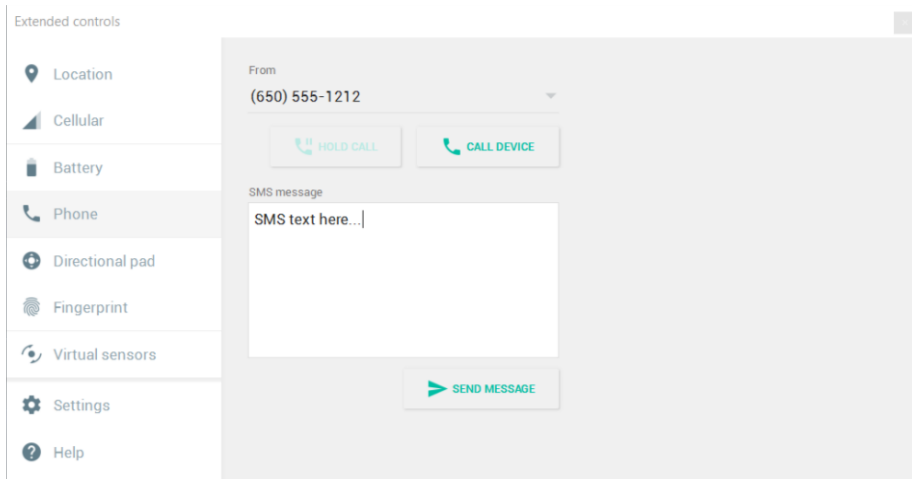
*La finestra
dell'Android Virtual
Device Manager non
deve essere chiusa
prima del completo
avvio dell'emulatore!*



AVD Extended Controls I

- L'emulatore, consente di simulare i valori di alcuni sensori, della posizione geografica (GPS), dello stato della batteria e della rete.
- Consente inoltre di simulare la ricezione di chiamate e l'invio di SMS al device.
- Per accedere a tali controllo, una volta avviato l'emulatore, si clicchi sul tasto "More", l'ultimo tra quelli proposti dalla barra laterale dell'emulatore.





AVD Extended Controls II



Utilizzo di Dispositivi Fisici durante lo sviluppo

- Ad Android Studio può essere collegato via USB un device fisico su cui eseguire le applicazioni in fase di sviluppo.
 1. Abilitare le opzioni per lo sviluppatore sul device
 2. In “Opzioni Sviluppatore”, abilitare il Debug via USB
 3. Accettare la fingerprint della sorgente (PC) alla prima esecuzione dell'applicazione sul device fisico

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