

Android

Letture consigliate

- **Android Developer, sito web di riferimento dei realizzatori di Android**
 - <http://developer.android.com/index.html>
- **Massimo Carli, Android 6, guida per lo sviluppatore, Apogeo**
 - Disponibili, dagli stessi autore ed editore, anche altri libri relativi alle versioni precedenti di Android



Cos'è Android?

- Android è un insieme di componenti software, comprendente un sistema operativo, un middleware e un insieme di applicazioni basilari
 - **Android è usualmente utilizzato in smartphones e tablets ma potrebbe essere esteso a supportare qualsiasi dispositivo, ipoteticamente anche un PC**

Breve storia di Android

- 2003: Andy Rubin fonda la Android Inc.
- 2005: Android viene acquisito da Google
- 2007: Viene fondata la Open Handset Alliance, consorzio comprendente 84 membri tra cui produttori di hardware, di software e compagnie di telecomunicazione (oltre a Google stessa), con lo scopo di realizzare congiuntamente tutto il necessario per la diffusione del sistema.
- 2007: viene rilasciato con licenza Apache l'Android Open Source Project

<http://source.android.com/>



Breve storia di Android

- 2007: Android Beta
- 2008: Android 1.0 (eseguibile su un unico dispositivo, HTC G1)
- 2009: Android 1.5 Cupcake (prima vera versione supportata da dispositivi ad ampia diffusione commerciale)
- 2009: Android 1.6 Donut
- 2009: Android 2.0 Éclair
- 2010: Android 2.2 Froyo
- 2010: Android 2.3 Gingerbread
- 2011: Android 3 Honeycomb
- 2011: Android 4 Ice Cream Sandwich



<http://blog.o2.co.uk/home/2011/05/android-versions-whats-the-difference.html>

Breve storia di Android

2012: Android 4.1/4.2/4.3 Jelly Bean



2013/14: Android 4.4 Kit Kat

2014: Android 5 Lollipop
Introduzione della nuova macchina virtuale ART



2015: Android 6 Marshmallow



2016: Android 7 Nougat

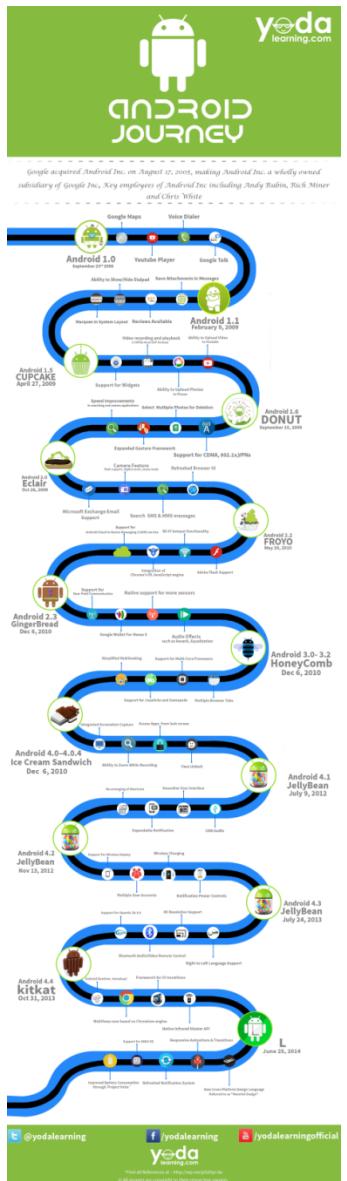
<http://www.cnet.com/news/history-of-android/>

Android 7 Nougat



- **Supporto per l'apertura di finestre multiple sullo schermo**
 - Ma solo l'ultima con cui si è interagito è davvero attivo, mentre le altre sono in pausa
 - C'è però il supporto per il drag & drop
- **Miglioramento delle performance del compilatore**
- **Modalità Doze (stand by intelligente) per risparmiare batteria**
- **Project Svelte, per minimizzare l'utilizzo di memoria delle app in background**
- **Riduzione delle comunicazioni quando sta per finire la disponibilità dal contratto dati**
- **E molto altro:**

<https://developer.android.com/about/versions/nougat/android-7.0.html>



Android Architecture (2011)

Java



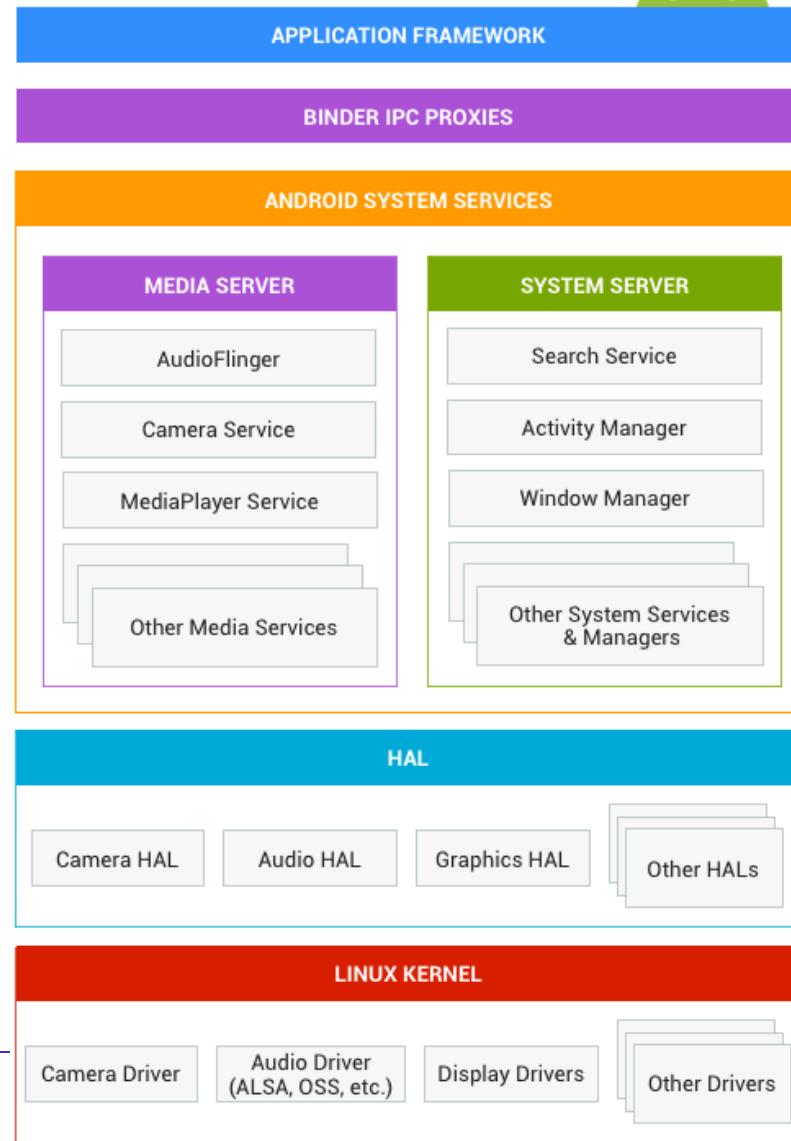
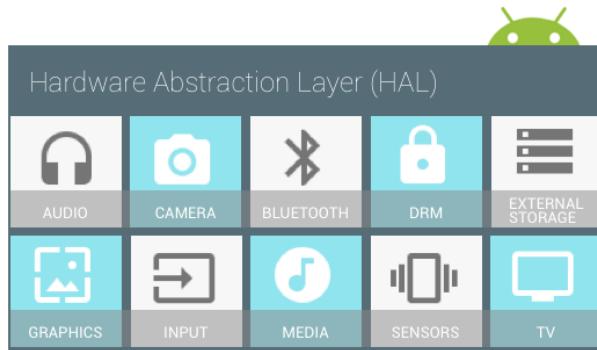
C/C++

Kernel

Android Architecture (2015)

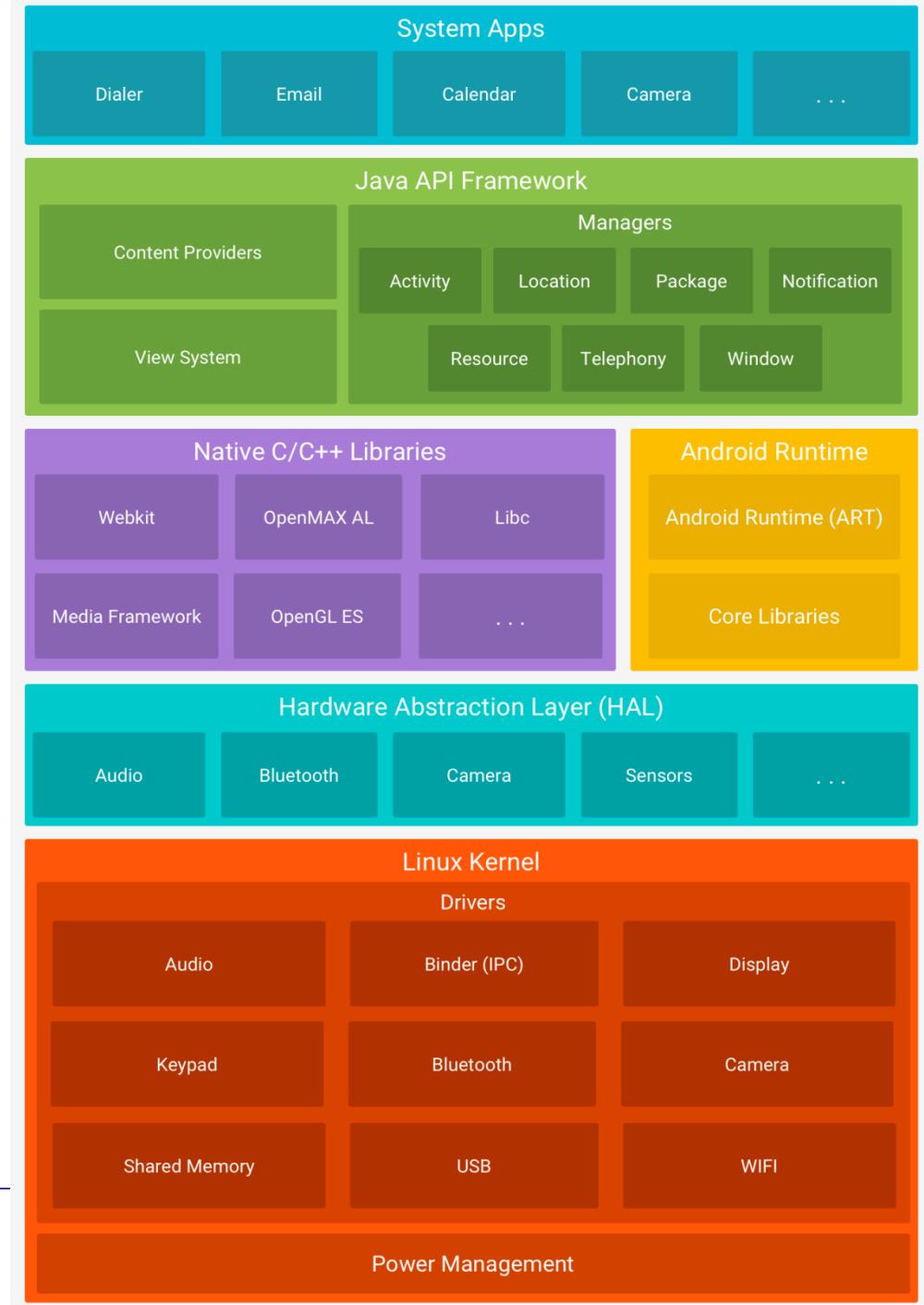


- <https://source.android.com/devices/>



Android Architecture (2016)

- <https://developer.android.com/guide/platform/index.html>



Sistema operativo e librerie

- Il sistema operativo sottostante è una distribuzione, opportunamente ridotta di Linux (kernel 2.6)
- Un certo numero di librerie di base sono state inserite, per supportare alcune features fondamentali:

Handset layouts

Storage

Connectivity

Messaging

Multiple language support

Web browser

Java support

Media support

Streaming media support

Additional hardware support

Multi-touch

Bluetooth

Video calling

Multitasking

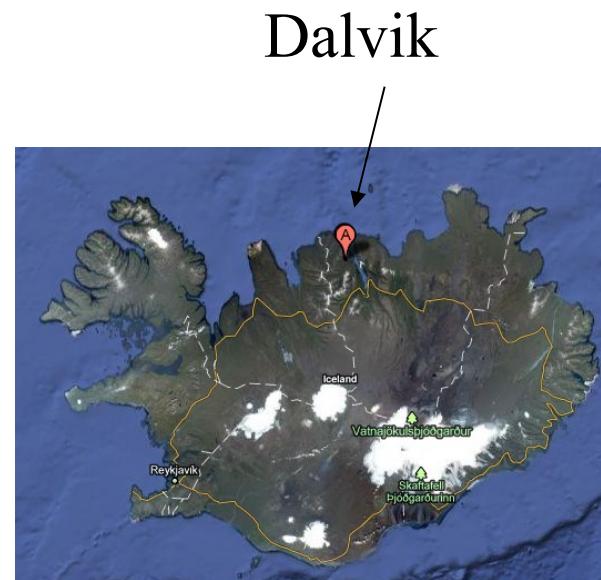
Voice based features

Tethering

Screen capture

Dalvik Virtual Machine

- All'interno di ogni dispositivo Android è presenta una virtual machine, denominata **Dalvik**
 - Dalvik è una macchina virtuale open source in grado di eseguire bytecode, in maniera simile alla Java Virtual Machine della Sun
 - Il bytecode è denominato dex (Dalvik executable)
 - Dalvik è ottimizzata per macchine dalla ridotta memoria
 - Gestisce i thread, con alcune limitazioni
 - Non gestisce le eccezioni
- Ogni applicazione su Android è vista come un diverso user, con un proprio processo, una propria zona dati e una propria istanza di Dalvik virtual machine.
- In futuro, Dalvik sarà integrata/sostituita da ART



Il nome Dalvik deriva dal villaggio di pescatori Dalvíkurbyggð di cui la famiglia di Bornstein, dipendente Google e autore della VM è originaria.

ART Virtual Machine

- **Nuova virtual machine**
 - In Android 4.4 Kit Kat si affianca alla Dalvik
 - A partire da Android 5 Lollipop è l'unica VM
- **Art è basata su tecnologia AOT (ahead-of-time)**
 - Il codice compilato dell'applicazione è direttamente eseguibile
 - Ciò riduce notevolmente i tempi di esecuzione rispetto ad un compilatore JIT (Just-In-Time) come Dalvik
 - Di conseguenza c'è anche un corrispondente risparmio energetico
 - Aumenta, invece, il consumo di spazio sulla memoria del dispositivo
 - Salvo eccezioni, è garantita la compatibilità delle applicazioni (il dex compilato è anche l'input per la ART)

Java Framework

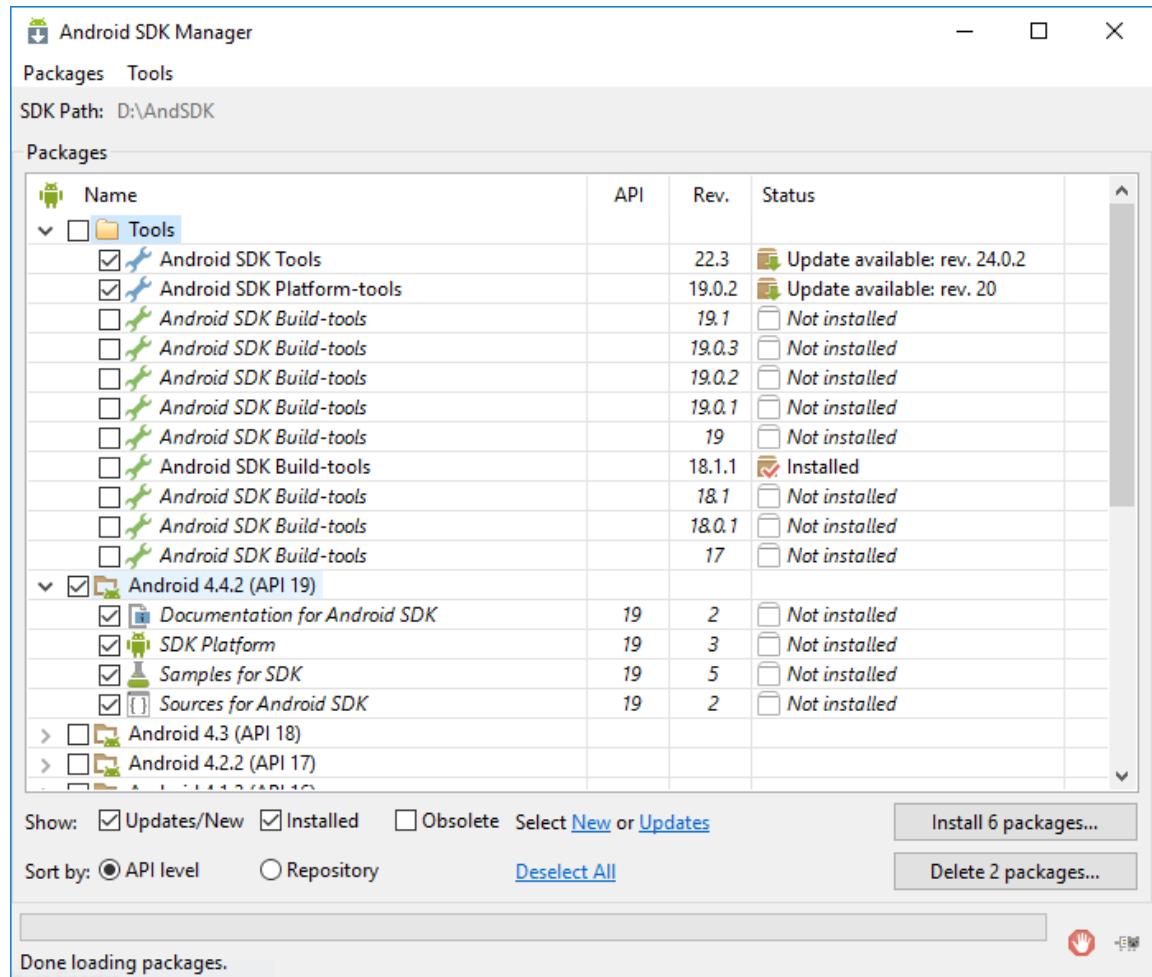
- E' possibile scrivere applicazioni in Java e poi compilarle nel formato dex eseguibile dalla Dalvik Virtual Machine
- La versione di riferimento è J2SE, non J2ME
- **Java non è l'unico possibile linguaggio di programmazione utilizzabile sotto Android, ma è largamente il più comune, nonché quello con maggior supporto software e consigliato dagli stessi sviluppatori di Android**
 - E' possibile realizzare, ad esempio, applicazioni e componenti in C/C++ con il supporto dell'Android Native Development Kit (NDK)
 - <http://developer.android.com/sdk/ndk/overview.html>

Android SDK

- **Android Standard Development Kit è il contenitore di tutti gli strumenti fondamentali per lo sviluppo di applicazioni Android in Java**
- **Scaricabile da:**
 - <http://developer.android.com/sdk/installing.html>
- **Disponibile per tutti i sistemi operativi più diffusi**
- **Non ha necessità di installazione**
- **Comprende:**
 - Strumenti a supporto dello sviluppo
 - Emulatore
 - Documentazione
 - Esempi
 - Strumenti di utilità

Installazione dei componenti aggiuntivi

- Tramite **SDK Manager** è possibile scaricare tutti i componenti aggiuntivi necessari all'esecuzione delle applicazioni in tutte le versioni di Android
- Le applicazioni Android possono essere eseguite su device reali o emulati



Android Virtual Device

- E' possibile generare macchine virtuali Android riproducenti le caratteristiche di una macchina reale tramite AVD Manager
- Nella interfaccia offerta da Android Studio, la creazione di una macchina virtuale può essere effettuata in tre step

Scelta del dispositivo

Category	Name	Size	Resolution	Density
Phone	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
	5,4" FWVGA	5,4"	480x854	mdpi
	5,1" WVGA	5,1"	480x800	mdpi
4,7" WXGA	4,7"	720x1280	xhdpi	

New Hardware Profile Import Hardware Profiles  Clone Device...

Nexus 5

1080px
4,95"
1920px

Size: normal
Ratio: notlong
Density: xxhdpi

Scelta del sistema operativo

Release Name	API Level	ABI	Target
Lollipop	22	armeabi-v7a	Android 5.1
Lollipop	22	x86	Android 5.1
Lollipop	22	x86_64	Android 5.1
Lollipop	21	x86	Android 5.0 (with Google APIs)
L	L	armeabi-v7a	Android 5.0
L	L	x86	Android 5.0
KitKat	19	armeabi-v7a	Android 4.4
Jelly Bean	17	armeabi-v7a	Android 4.2
IceCreamSandwich	15	armeabi-v7a	Android 4.0.3
IceCreamSandwich	15	mips	Android 4.0.3
IceCreamSandwich	15	x86	Android 4.0.3
IceCreamSandwich	15	armeabi-v7a	Android 4.0.3
Gingerbread	10	armeabi	Android 2.3.3
Gingerbread	10	x86	Android 2.3.3
Gingerbread	10	armeabi	Android 2.3.3
Froyo	8	armeabi	Android 2.2
Donut (Deprecated)	4	armeabi	Android 1.6
Donut (Deprecated)	4	armeabi	Android 1.6

Show downloadable system images 

Gingerbread

 API Level **10**
Android **2.3.3**
Android Open Source Project
System Image **armeabi**

Recommendation
Consider using an x86 system image for better emulation performance.
Consider using a system image with Google APIs to enable testing with Google Play Services.

Questions on API level?
See the [API level distribution chart](#)

Altre impostazioni AVD

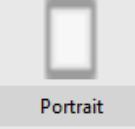
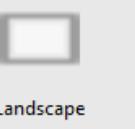
AVD Name

 Nexus 5 4,95" 1080x1920 xxhdpi

 Gingerbread Android 2.3.3 armeabi

Startup size and orientation:

Scale:

Orientation:  
 Portrait Landscape

Emulated Performance:

Use Host GPU (Requires API > 15)
 Store a snapshot for faster startup
You can either use Host GPU or Snapshots

Device Frame: Enable Device Frame

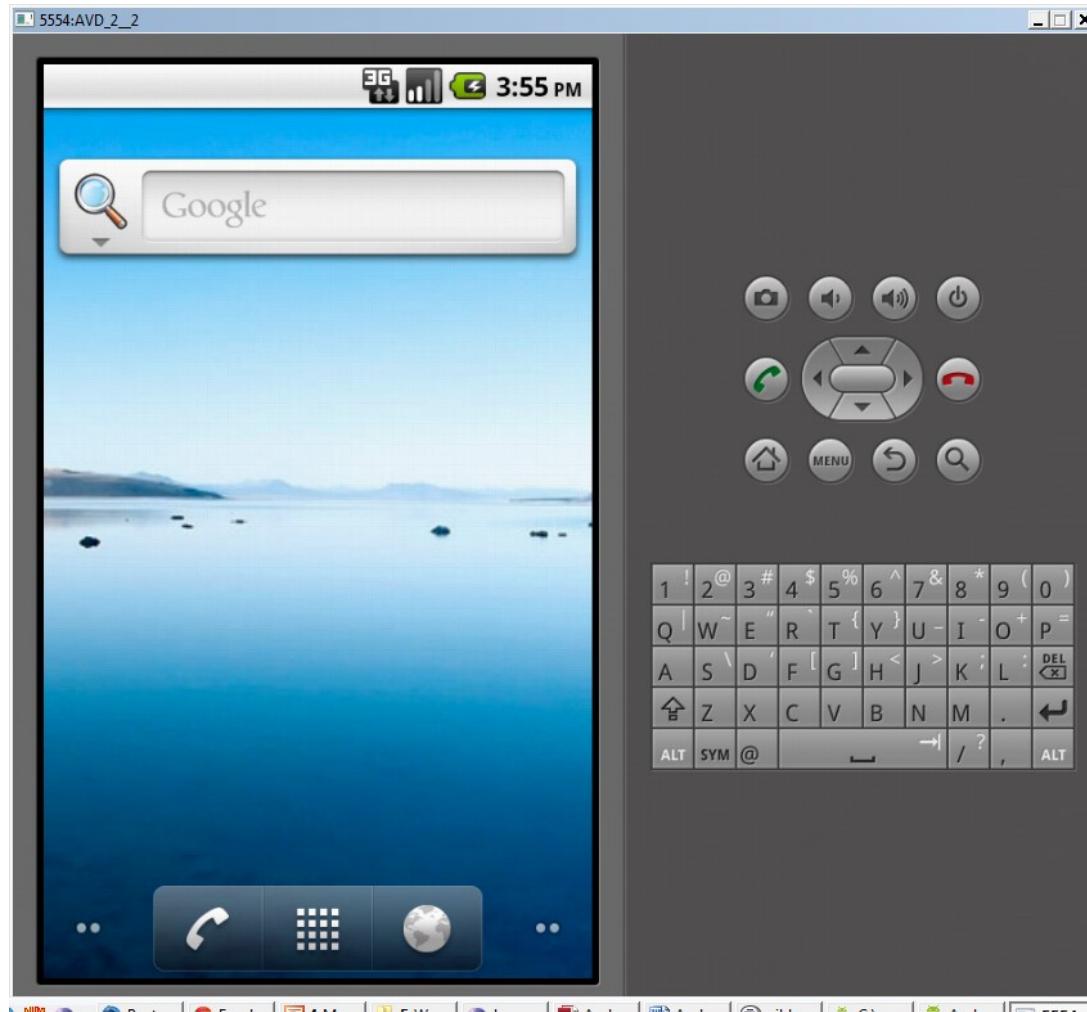
AVD Name

The name of this AVD.

Recommendation

Consider using an x86 system image for better emulation performance.
Consider using a system image with Google APIs to enable testing with Google Play Services.

Emulatore



Altri Emulatori

- **Le prestazioni degli emulatori Android di base sono spesso scadenti**
 - Esistono versioni dell'emulatore Android specializzate per processori ARM e Intel
 - In alcuni PC è necessario abilitare le funzionalità di virtualizzazione dal BIOS
 - Esistono altri emulatori più performanti in specifiche condizioni:
 - GenyMotion (<https://www.genymotion.com>)
 - ManyMo (<https://www.manymo.com/>)
 - BlueStacks (<http://www.bluestacks.com/>)
 - Andy (<http://www.andROID.net/>)

Alcune utility dell'SDK

- **Tutte richiamabili anche direttamente da linea di comando**
- emulator
 - **Avvia un emulatore**
- adb - Android Debug Bridge
 - **Consente la comunicazione con un dispositivo Android, reale o emulato**
 - adb push / adb pull – trasferisce file verso/dal dispositivo
 - adb install – installa un'applicazione
 - adb logcat – mostra il log di debug
 - adb shell – avvia una shell linux sul dispositivo
 - ...
- android
 - **Un file batch dal quale è possibile avviare molte utility per la gestione dei progetti android da linea di comando**
 - android create project \
 --target <target_ID> \
 --name <your_project_name> \
 --path path/to/your/project \
 --activity <your_activity_name> \
 --package <your_package_namespace>
- **... e molte altre**

Android ed Eclipse: ADT

- **ADT (Android Development Toolkit) è un'estensione di eclipse totalmente dedicata ad Android**
 - Scaricabile via Eclipse all'indirizzo <https://dl-ssl.google.com/android/eclipse/>
 - Scaricabile insieme a tutto l'ambiente Eclipse da:
 - Win x86: <http://dl.google.com/android/adt/adt-bundle-windows-x86-20140702.zip>
 - Win x64 : http://dl.google.com/android/adt/adt-bundle-windows-x86_64-20140702.zip
- **Integra nell'ambiente di sviluppo Eclipse tutti gli strumenti necessari allo sviluppo e all'esecuzione di un'applicazione Android**
 - In particolare, rende possibile l'utilizzo di quasi tutte le utility disponibili a linea di comando

Android Studio

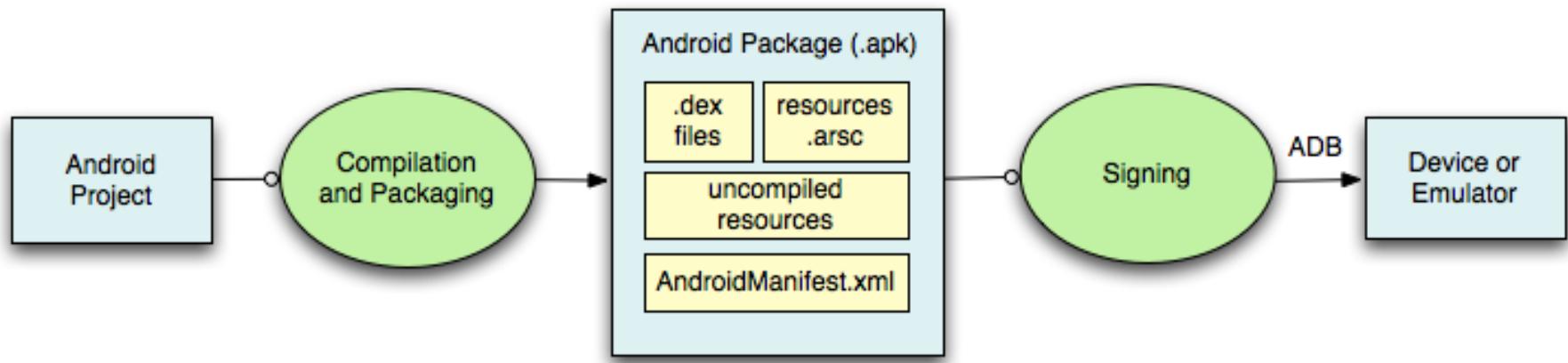
- Più recentemente, Google ha adattato un altro ambiente di sviluppo (IntelliJ Idea) ad Android, costruendo un nuovo ambiente denominato **Android Studio**
 - <http://developer.android.com/sdk/installing/studio.html>



Principali differenze tra Android Studio e Eclipse ADT

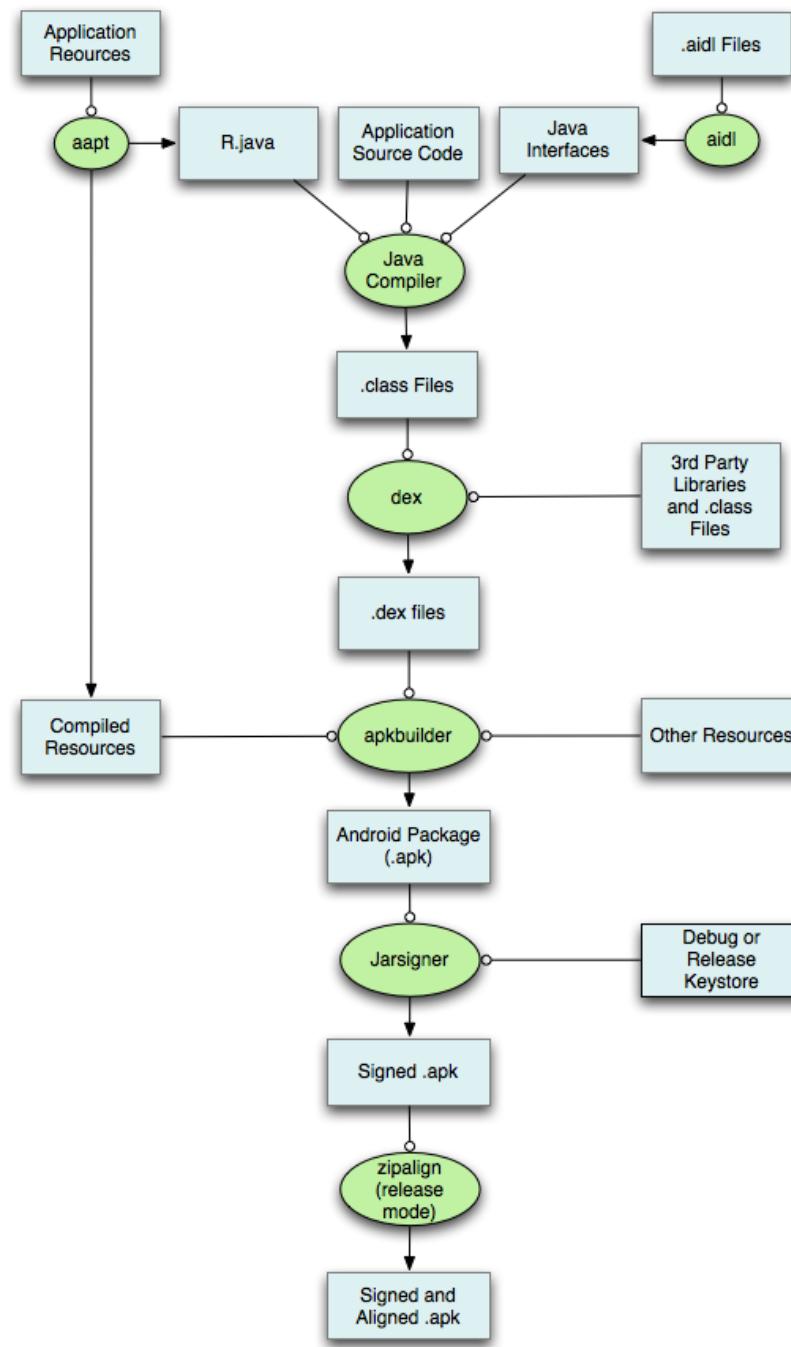
- **Android Studio**
 - Consente di gestire un solo progetto per volta
 - Ha un wizard per la trasformazione di un progetto ADT verso Android Studio
 - Più semplice gestione delle librerie (in ADT spesso era necessario includerle come sottoprogetti)
 - Più semplice gestione delle dipendenze (grazie a gradle utilizzato al posto di ant)
 - Maggiore quantitativo di funzioni utili già presenti nell'IDE (in Eclipse si sarebbe dovuto installarle come plugin di terze parti)

Processo di build (Dalvik)



Build (Dalvik)

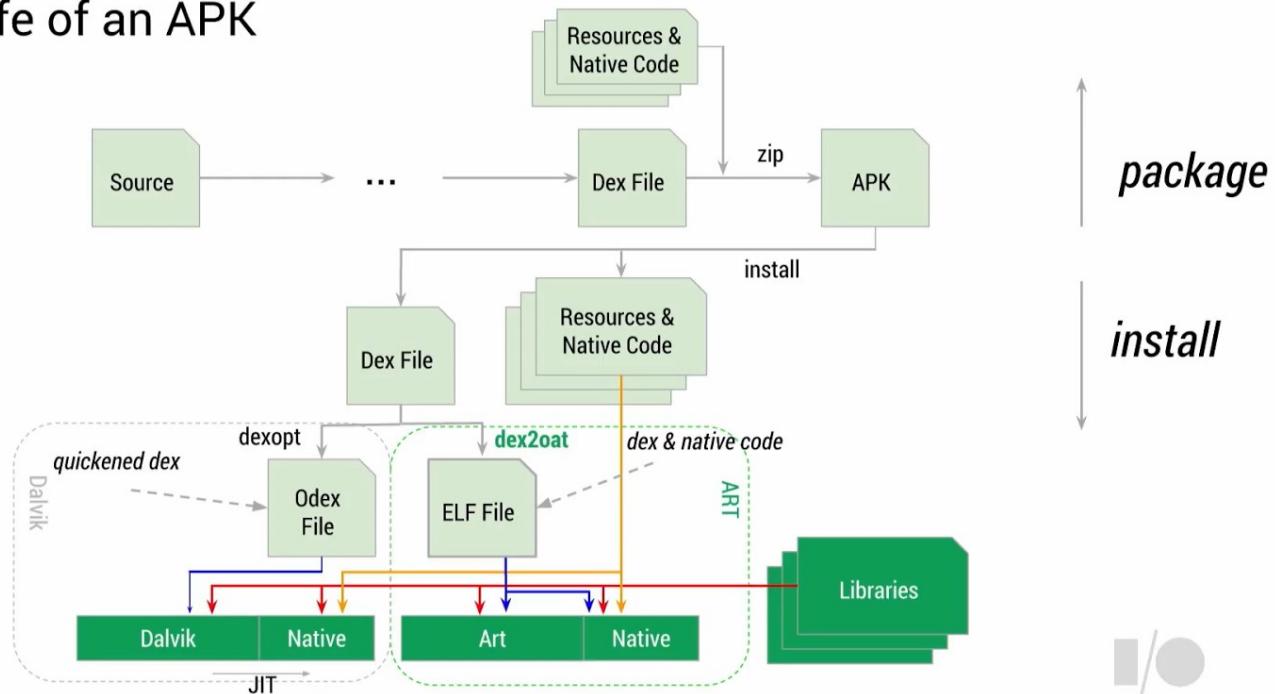
- Aapt: Android Asset Packaging Tool
 - Legge gli xml e genera R.java
- Aidl: Android Interface Definition Language
 - Aidl converte interfacce di servizi .aidl in interfacce Java
- Tutto il codice java è compilato generando bytecode .class
- Dex converte i .class in file dex eseguibili da Dalvik (e include eventuali librerie)
- Apkbuilder comprime e impacchetta i .dex e le risorse (grafiche, etc.) in un unico file .apk
- Jarsigner permette di inserire una firma nel .apk
- Zipalign consente di ottimizzare le risorse di memoria utilizzata dall'applicazione in un dispositivo



Build (Dalvik / ART)

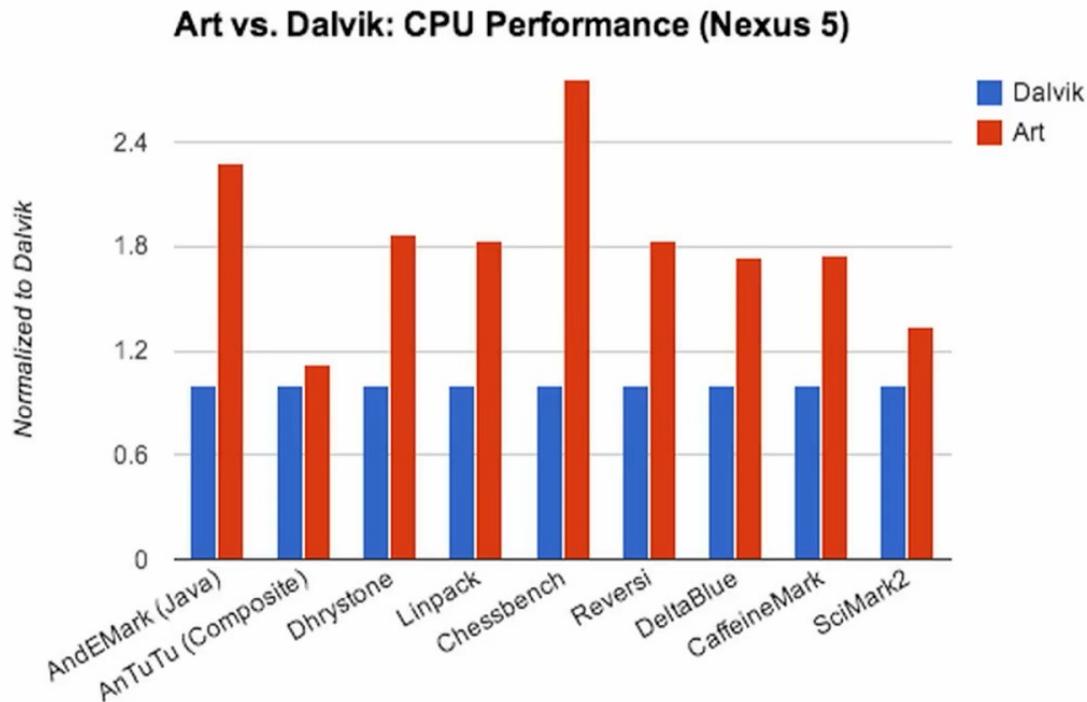
- <http://www.anandtech.com/show/8231/a-closer-look-at-android-runtime-art-in-android-l>

The life of an APK



Performance boosting con ART

Performance Boosting Thing, realized



Signing

- Ogni applicazione Android ha bisogno di una firma privata, che identifichi l'autore, per poter essere pubblicata
 - La firma serve per garantire fiducia nell'applicazione
 - La firma è messa direttamente dall'autore stesso
 - La firma può essere generata da tools integrati nell'ambiente, come Keytool e Jarsigner
- Finchè l'applicazione è in fase di sviluppo e testing, è possibile utilizzare una modalità di firma *debug mode*
 - In debug mode la firma è costante (ma l'applicazione non può essere pubblicata)
 - Keystore name: "debug.keystore"
 - Keystore password: "android"
 - Key alias: "androiddebugkey"
 - Key password: "android"
 - CN: "CN=Android Debug,O=Android,C=US"
 - Per passare in *release mode* è necessaria una firma dell'utente
 - Il plug-in ADT Export Wizard è in grado di supportare il processo di signing
- Se si vogliono rilasciare degli aggiornamenti dell'applicazione, è necessario riutilizzare la stessa firma

Android Manifest

- **Tutte le caratteristiche esterne di una applicazione Android sono strutturate in un file manifest.xml**
- **Manifest.xml è un file pubblico, che può essere letto in chiaro da ogni possibile utente della app**
 - Si tratta di una pratica molto diffusa, nei framework di nuova generazione: dichiarare tutte le costanti di configurazione in file xml statici, che vengono elaborati da qualche metodo del framework, in maniera trasparente al programmatore

Esempio di manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.porfirio.cacciaaltesoro"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-permission
        android:name="android.permission.ACCESS_FINE_LOCATION" />
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".CacciaAlTesoro" <-- Nomi delle activity
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /> <-- E' l'activity di partenza
                <category android:name="android.intent.category.LAUNCHER" /> <-- dell'applicazione
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Icona

A quale “richiesta” (intent) risponde l’applicazione

Accesso a servizi GPS

Nomi delle activity

E’ l’activity di partenza dell’applicazione

Comparirà tra le applicazioni lanciabili

Interfaccia ADT Android Manifest

The screenshot shows the 'Android Manifest' editor window. At the top, there's a toolbar with icons for file operations like New, Open, Save, and Delete. Below the toolbar, the title bar says 'Android Manifest'. Underneath the title bar, there's a section titled 'Manifest General Attributes' with a description: 'Defines general information about the AndroidManifest.xml'. This section contains five input fields: 'Package' (set to 'com.porfirio.caccialtesoro'), 'Version code' (set to '1'), 'Version name' (set to '1.0'), 'Shared user id' (empty), and 'Shared user label' (empty). Each field has a 'Browse...' button to its right. Below these fields is a section titled 'Manifest Extras' with a toolbar containing icons for 'U' (Uses Sdk), 'S' (SDK), 'P' (Permissions), 'U' (Users), 'U' (Units), and 'Az' (Annotations). A list box on the left contains the item '.....U Uses Sdk'. To the right of the list box is a vertical toolbar with four buttons: 'Add...', 'Remove...', 'Up', and 'Down'.

- In Android Studio si edita direttamente il codice XML (ma spesso viene automaticamente generato)

Interfaccia ADT Android Manifest

Android Manifest Application

▼ **Application Toggle**

The [application](#) tag describes application-level components contained in the package, as well as general application attributes.

Define an <application> tag in the AndroidManifest.xml

▼ **Application Attributes**

Defines the attributes specific to the application.

Name	<input type="text"/>	Browse...	Has code	<input type="text"/>
Theme	<input type="text"/>	Browse...	Persistent	<input type="text"/>
Label	<input type="text"/> @string/app_name	Browse...	Enabled	<input type="text"/>
Icon	<input type="text"/> @drawable/icon	Browse...	Debuggable	<input type="text"/>
Description	<input type="text"/>	Browse...	Manage space activity	<input type="text"/> Browse...
Permission	<input type="text"/> android.permission.ACCESS_FINE_LOCATION	<input type="button"/> ▼	Allow clear user data	<input type="text"/>
Process	<input type="text"/>	Browse...	Test only	<input type="text"/>
Task affinity	<input type="text"/>	Browse...	Backup agent	<input type="text"/> Browse...
Allow task reparenting	<input type="text"/>	<input type="button"/> ▼	Allow backup	<input type="text"/>

Application Nodes S P A R M U Az

- ↳ **A** .CacciaAlTesoro (Activity)
 - ↳ **I** Intent Filter
 - ⋮ **A** android.intent.action.MAIN (Action)
 - ⋮ **C** android.intent.category.LAUNCHER (Category)

Add...
Remove...
Up
Down

Android Market: play.google.com

- Strumento standard fornito da Android per la pubblicazione e la pubblicizzazione delle applicazioni
- La registrazione è molto semplice e richiede
 - **Un Account Google**
 - **Una quota di iscrizione di 25\$ (~18€) una tantum**
- Nel caso in cui si voglia vendere applicazioni bisogna fornire poi anche le proprie coordinate bancarie e il proprio codice fiscale

Ricerca di un'applicazione sul market

- **Android Play Store è disponibile**
 - come applicazione Web
 - <https://play.google.com/store>
 - come applicazione client su tutti i dispositivi Android
 - Non disponibile sugli emulatori
 - In passato si chiamava Android Market ed era utilizzato solo per la distribuzione di applicazione. Ora, invece, contiene anche contenuti multimediali (film, musica, libri, giornali)
- **In alternativa, è possibile distribuire applicazioni proprie anche in maniera diretta**
 - Esistono ulteriori market, non approvati da Google

Pubblicazione di una applicazione

- Tramite l'applicazione Web
<https://play.google.com/apps/publish>

The screenshot shows the Google Play Developer Console interface. On the left, there's a sidebar with icons for 'Tutte le applicazioni', 'Servizi di giochi', 'Rapporti finanziari', 'Impostazioni', 'Avvisi', and 'Comunicazioni'. The main area displays a list of published apps:

NOME APPLICAZIONE	PREZZO	INSTALLAZIONI CORRENTI/TOTALE	VOTO MEDIO / N. TOTALE	ARRESTI ANOMALI E ANR	ULTIMO AGGIORNAMENTO	STATO
Apin 1.0	Gratuita	—	—	—	08/giu/2014	Pubblicata
Misteri di Procida 1.3	Gratuita	47 / 539	★ 4,44 / 16	—	12/mar/2014	Pubblicata
Uno Procida Residente 1.12.2	Gratuita	1.672 / 2.553	★ 4,64 / 97	7	06/gen/2014	Pubblicata

For the app 'Uno Procida Residente 1.12.2', the details page is shown:

UNO PROCIDA RESIDENTE
com.porfirio.oranprocida2011 [Visualizza in Google Play Store](#)
PUBBLICATA Annulla la pubblicazione dell'app

APK

PRODUZIONE Versione 25	TEST BETA Configura i test Beta per la tua app	TEST ALPHA Configura i test Alpha per la tua app
--	--	--

CONFIGURAZIONE PRODOTTO [Carica nuovo APK per Produzione](#)

APK CORRENTE: pubblicato il giorno 06/gen/2014 04:27:27

Dispositivi supportati
7532 [Visualizzazione elenco](#)

Dispositivi esclusi
0 [Gestisci i dispositivi esclusi](#)

VERSIONE **CARICATO IN DATA** **STATO** **AZIONI**

25 (1.12.2)	06/gen/2014	in Prod	
-------------	-------------	---------	--

Pubblicazione di una applicazione

- **Per pubblicare un'applicazione è necessario**
 - L'apk firmato
 - Almeno 2 screenshots
 - Un'icona ad alta risoluzione e un banner
 - Titolo e descrizione (per ogni lingua che si vuole supportare)
 - Descrizione delle ultime modifiche
 - Tipo e categoria
 - Classificazione dei contenuti
 - Prezzo (se non gratis)
- **Sono automaticamente ricavati dal manifest**
 - Versioni di Android compatibili
 - Dispositivi compatibili
 - Permessi da richiedere all'utente installatore

Modelli di business (cenni)

- **Applicazione gratuita**
 - E' possibile, però, entro certi limiti, lucrare dal servizio svolto dall'applicazione, ad esempio con e-commerce
- **Applicazione con banner pubblicitari**
 - Ogni scaricamento del banner e, in misura maggiore, ogni click su esso fornisce ricavi
 - I ricavi vengono accreditati periodicamente
 - Esistono molti circuiti pubblicitari, tra cui Google Ads
- **Applicazione a pagamento**
 - Google trattiene il 30% del costo, ma accredita velocemente il ricavato
- **Applicazioni con acquisti in-app**
 - E' possibile fare acquisti internamente alla app
 - Ad esempio, in un gioco è possibile acquistare nuove armi, in whatsapp è possibile acquistare un prolungamento della licenza

Crash ed eccezioni

ARRESTI ANOMALI E ANR [Esporta come CSV](#)

Tipo	Mostra nascosti	Ultima segnalazione	Versione di Android	Versione applicazione	Dispositivo
<input checked="" type="checkbox"/> Arresti anomali <input type="checkbox"/> ANR	<input checked="" type="checkbox"/> Sì <input type="checkbox"/> NO	<input type="button" value="Ultimi 90 giorni"/>	<input type="button" value="Tutte le versioni"/>	<input type="button" value="Attualmente in produzione"/>	<input type="button" value="Aggiungi filtro"/>
0 nuovi arresti anomali ? 2 arresti anomali totali					
NOME	<input type="button" value="▼ NUOVO ?"/>	SEGNALAZIONI IN QUESTA SETTIMANA	TOTALE SEGNALAZIONI	ULTIMA SEGNALAZIONE	NASCONDI
java.lang.NullPointerException in com.porfirio.orariprocida2011.OrariProcida2011Activity.leggiMeteo			1	30 set 17:29	<input type="button" value="Nascondi"/>
java.util.NoSuchElementException in java.util.StringTokenizer.nextToken			31	5 ago 23:36	<input type="button" value="Nascondi"/>

Pagina 1 di 1

Pagina 1 di 1

I rapporti sugli arresti anomali mostrati qui vengono conservati per sei mesi. L'esportazione collettiva dei rapporti sugli arresti anomali è disponibile per 18 mesi.

Application Framework

- **Il livello di application framework fornisce un insieme di classi, interfacce e package tramite i quali è possibile sviluppare applicazioni. Ad esempio:**
 - Activity Manager gestisce il ciclo di vita delle activity
 - Content providers consente la condivisione delle informazioni tra diverse applicazioni e servizi
 - Telephony Manager gestisce le azioni legate al telefono
 - Location Manager gestisce le informazioni legate al GPS
 - Notification Manager consente la gestione delle informazioni visualizzate come alert sulla barra di stato del dispositivo
 - ...

Application components (cenni)

Activity

- Il componente fondamentale di una applicazione Android interattiva, poiché rappresenta il contenitore per i widget di una schermata.
- Eredita dalla classe Activity

Service

- I Services sono componenti responsabili di esecuzioni in background, senza possibilità di ricevere input dall'interfaccia utente, eccetto che tramite il Notification Manager
- Un service è implementato come una classe che eredita dalla classe Services della quale poi istanziare un oggetto

Broadcast Receiver

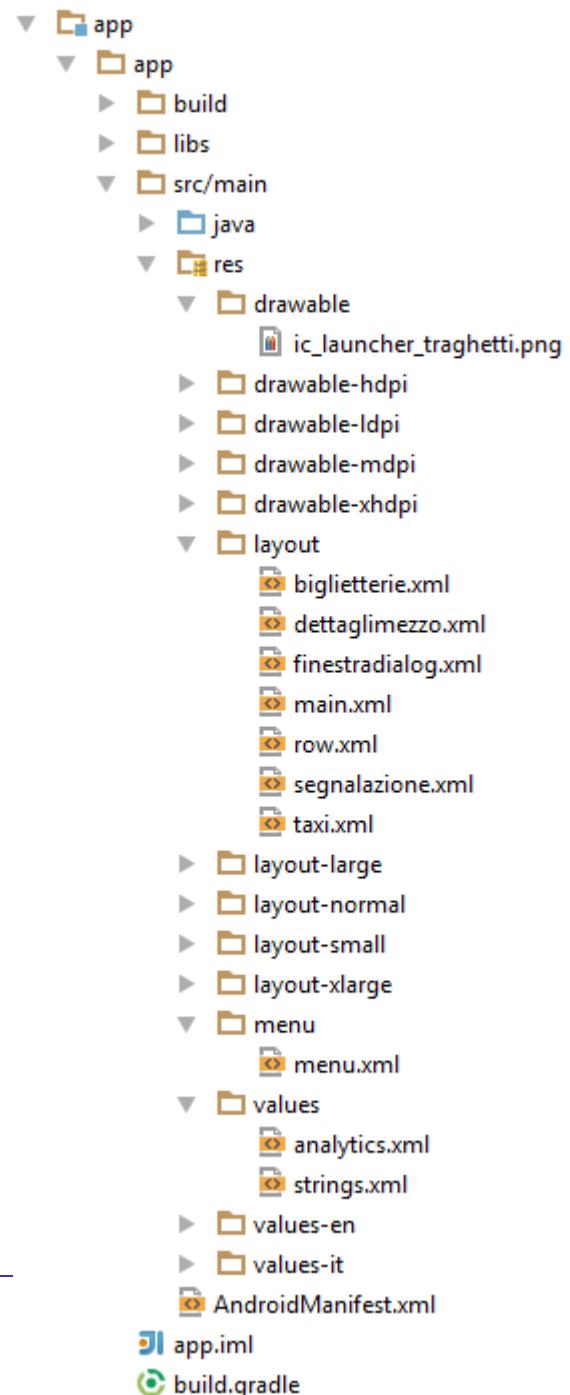
- Un Broadcast Receiver è un ascoltatore di eventi di sistema (Intent), che può partire appena uno di essi arriva, come una ISR
- Ad un Broadcast Receiver è associato un piccolo spezzone di codice che comprende di solito chiamate ad Activity o Services (da eseguire in un tempo limite, oltre il quale il sistema deduce che ci sia stato uno stallo)

Content Provider

- Un Content Provider è un componente che fa da interfaccia verso una sorgente di dati persistente. Un Content Provider consente il disaccoppiamento tra gestione dei dati e resto dell'applicazione
 - In Android sono disponibili librerie per l'interfacciamento con SQLite, un dbms estremamente leggero

Resources

- In Android è ampiamente adottata la tecnica di catalogare tutte le risorse (costanti, descrizioni di interfacce, file grafici, etc.) sotto forma di Resources
- Le risorse sono elaborate automaticamente dal framework di esecuzione tramite l'utility aapt, creando un file R.java che le cataloga e le rende accessibili da codice



File R

- Il framework di esecuzione tiene conto delle caratteristiche dell'ambiente di esecuzione
 - Vengono automaticamente selezionati, ad esempio, il layout e le immagini più consone alle dimensioni e alla risoluzione dello schermo del dispositivo target

```
1+/* AUTO-GENERATED FILE. DO NOT MODIFY. */
2
3 package com.porfirio.orariprocida2011;
4
5
6 public final class R {
7     public static final class array {
8         public static final int strMezzi=0x7f060000;
9         public static final int strPorti=0x7f060001;
10    }
11    public static final class attr {
12    }
13    public static final class color {
14        public static final int blue=0x7f040001;
15        public static final int green=0x7f040002;
16        public static final int lightblue=0x7f040004;
17        public static final int main_back_ground_color=0x7f040003;
18        public static final int red=0x7f040000;
19    }
20    public static final class drawable {
21        public static final int ic_launcher_traghetti_512=0x7f020000;
22        public static final int ic_launcher_traghetti_hdpi=0x7f020001;
23        public static final int ic_launcher_traghetti_ldpi=0x7f020002;
24        public static final int ic_launcher_traghetti_mdpi=0x7f020003;
25        public static final int ic_launcher_traghetti_xhdpi=0x7f020004;
26    }
27    public static final class id {
28        public static final int about=0x7f08002d;
29        public static final int btnBack=0x7f080006;
30        public static final int btnBackTaxi=0x7f08002c;
31    }
32}
33}
34}
```

Internazionalizzazione

- **Un classico utilizzo delle risorse è quello che consente la realizzazione parallela di diverse versioni di un'applicazione per diverse lingue**
- **Tutte le stringhe di testo dell'interfaccia utente devono essere catalogate in una risorsa strings.xml**
- **Se vogliamo realizzare una app in inglese, francese, italiano**
- **Dovrebbe essere creato**
 - uno strings.xml con le stringhe in italiano nella cartella res/values-it
 - uno strings.xml con le stringhe in francese nella cartella res/values-fr
 - uno strings.xml con le stringhe in inglese nella cartella res/values-en (o nella cartella di default res/values)

Internazionalizzazione con Android Studio

Order a translation...

Key	Default Value	Untrap...	English (en)	Italian (it)
Conferma	Confirm	<input type="checkbox"/>	Confirm	Conferma
a	to	<input type="checkbox"/>	to	a
aRischio	Chance of cancellation due to bad weather conditions	<input type="checkbox"/>	Chance of cancellation due to bad weather conditions!!!	A rischio Maltempo!!!
about	Info	<input type="checkbox"/>	Info	Info
aggiornamentoDaWeb	Timetables updated from Web	<input type="checkbox"/>	Timetables updated from Web	Aggiornamento degli orari letto da Web
aliscafi	Hydrofoils	<input type="checkbox"/>	Hydrofoils	Aliscafi
aliscofo	Hydrofoil	<input type="checkbox"/>	Hydrofoil	Aliscofo
altriDettagliRagione	"Other Details : "	<input type="checkbox"/>	"Other Details : "	"Altri dettagli : "
app_name	Uno Procida Residente	<input type="checkbox"/>	Uno Procida Residente	Uno Procida Residente
arrivaAlle	Arrives at	<input type="checkbox"/>	Arrives at	Arriva alle
back	Back	<input type="checkbox"/>	Back	Indietro
backTimetable	Back to timetable	<input type="checkbox"/>	Back to timetable	Torna agli orari
bavaDiVento	Light air	<input type="checkbox"/>	Light air	Bava di vento
brezzaLeggera	Light breeze	<input type="checkbox"/>	Light breeze	Brezza leggera
brezzaTesa	Gentle breeze	<input type="checkbox"/>	Gentle breeze	Brezza tesa
burrasca	Gale	<input type="checkbox"/>	Gale	Burrasca
burrascaForte	Strong gale	<input type="checkbox"/>	Strong gale	Burrasca forte
calma	Calm	<input type="checkbox"/>	Calm	Calma
cheLaCorsaERegolare	that the journey will take place	<input type="checkbox"/>	that the journey will take place	che la corsa e\' regolare
circa	(approximately)	<input type="checkbox"/>	(approximately)	(circa)
condimeteo	Meteorological conditions:	<input type="checkbox"/>	Meteorological conditions:	Condizioni meteo:
confermaOSmentisci	Confirm that the journey will take place or report a problem	<input type="checkbox"/>	Confirm that the journey will take place or report a problem	Conferma la regolarita\' o segnala un problema per questa corsa
connessioneLenta	The connection is too slow: timetable updates have not been loaded	<input type="checkbox"/>	The connection is too slow: timetable updates have not been loaded	La connessione e\' troppo lenta: gli ultimi aggiornamenti degli orari non sono stati caricati
corsaImpossibile	Surely cancelled for bad weather conditions !!!	<input type="checkbox"/>	Surely cancelled for bad weather conditions !!!	Corsa impossibile !!!
corsaQuasi	Probably cancelled for bad weather conditions !!!	<input type="checkbox"/>	Probably cancelled for bad weather conditions !!!	Corsa quasi sicuramente sospesa !!!
costo	Price :	<input type="checkbox"/>	Price :	Costo :
Key: <input type="text"/>				
Default Value: <input type="text"/>				
Translation: <input type="text"/>				

Compatibilità con gli schermi

- **Si può utilizzare una soluzione analoga a quella usata per l'internazionalizzazione**
 - Diverse cartelle contenenti versioni diverse dei file di layout
 - Cartelle etichettate convenzionalmente in base a:
 - Dimensione (small-normal-large-xlarge)
 - Densità
 - Orientamento (land-port)
 - Proporzioni
 - I nomi delle cartelle si ottengono componendo le etichette.
Ad esempio:
 - res/layout-xlarge-land/

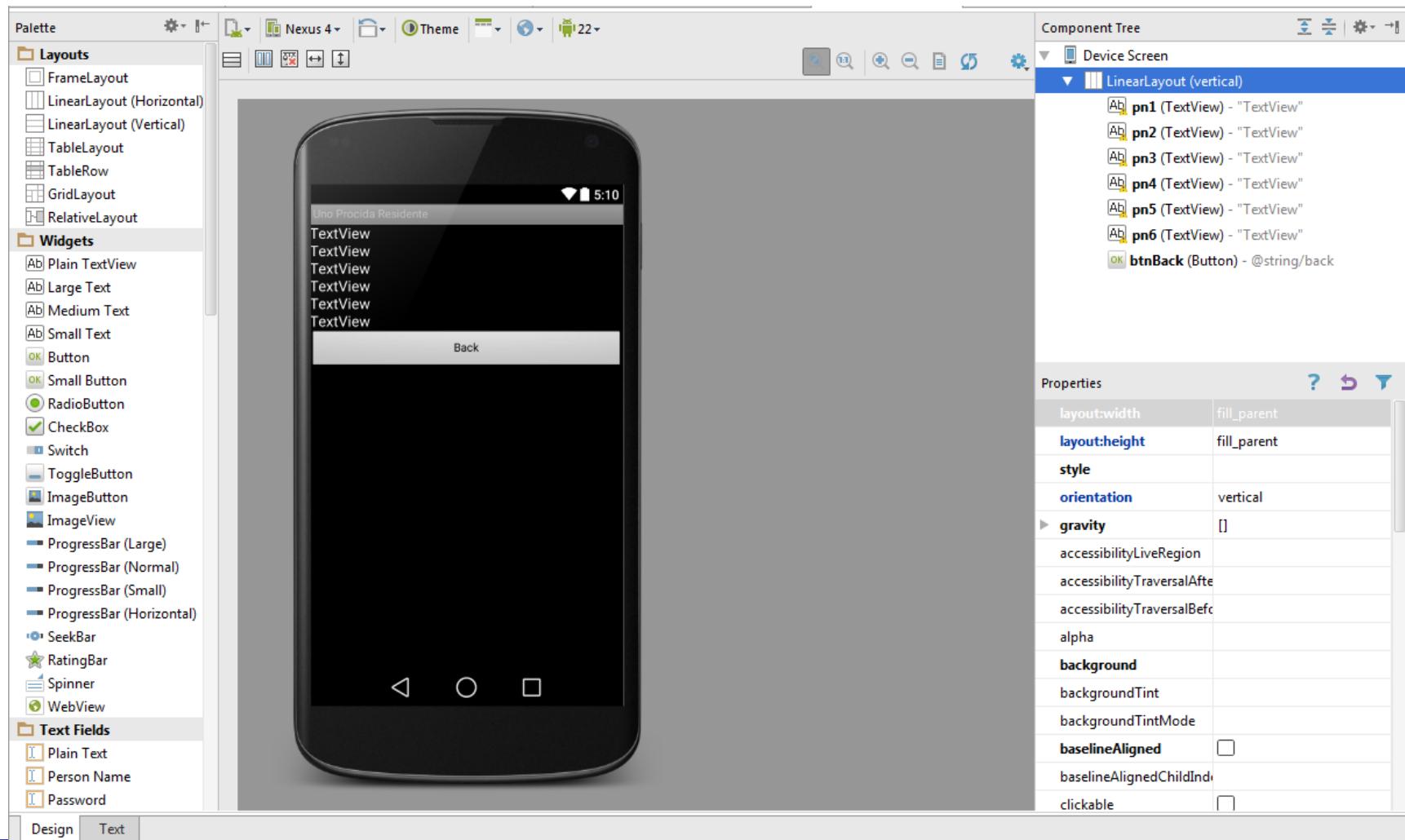
Accesso alle risorse

- **Da codice java, tramite il file R. Sono disponibili funzioni come findViewById(id), getString(id) ...**
 - ImageView imageView = (ImageView) findViewById(R.id.myimageview);
- **Da file xml (ad esempio da un file di layout è possibile accedere ad un valore di una string):**
 - <Button android:text="@string/submit"

Layout e resources

- Il layout grafico di un'applicazione Android potrebbe essere gestito completamente tramite codice sorgente, nel quale fossero istanziati dinamicamente gli oggetti dell'interfaccia utente, poi settati e utilizzati
- In alternativa, grazie alle funzionalità di Android Development Toolkit è possibile esprimere in forma dichiarativa il layout grafico di una Activity in XML
- L'estensione ADT di Eclipse consente, poi, di avere una preview grafica dell'interfaccia così progettata

Layout grafico



Layout grafico

- **Ad ogni Activity può essere associato un layout grafico (un file .xml)**
 - Anche altri componenti (ad esempio i Customized Dialogs) possono avere il proprio file di layout
 - Anche singoli widget o sezioni di activity possono avere propri layout
- **I layout xml sono gestiti via codice fondamentalmente con metodi come setContentView**

```
setContentView(R.layout.main);
```

 - Main è il nome del layout (main.xml)
 - Il comando è nel metodo onCreate dell'Activity

Stili e Temi

- **Android ci consente di definire e utilizzare un ulteriore tipo di risorsa che prende il nome di tema o stile**
 - ha uno scopo simile a quello dei CSS (Cascading Style Sheet) per le pagine HTML.
 - Uno stile può essere applicato ai componenti
 - Un tema può essere assegnato a un'activity oppure a un'intera applicazione ed è sostanzialmente un insieme di stili.

Stili e Temi

- **Indicando uno stile, il componente erediterà tutti gli attributi dello stile**
 - Tutto ciò che non è definito nell'ambito degli attributi di un componente è ereditato dal corrispondente stile
- **Uno stile eredita obbligatoriamente a sua volta da un tema genitore**
 - Tutto ciò che non è definito nell'ambito dello stile è ereditato dal corrispondente tema
- **Nelle prime versioni di Android non c'erano stili e temi: tutti gli attributi dovevano essere scritti per ogni componente**
- **Tramite funzionalità di refactoring è possibile estrarre stili dai componenti**

Esempio Style e Theme

- Fissa quattro proprietà (applicabili ad esempio ad un componente) e ne eredita altri da uno stile denominato `TextAppearance.Medium`

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CodeFont" parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

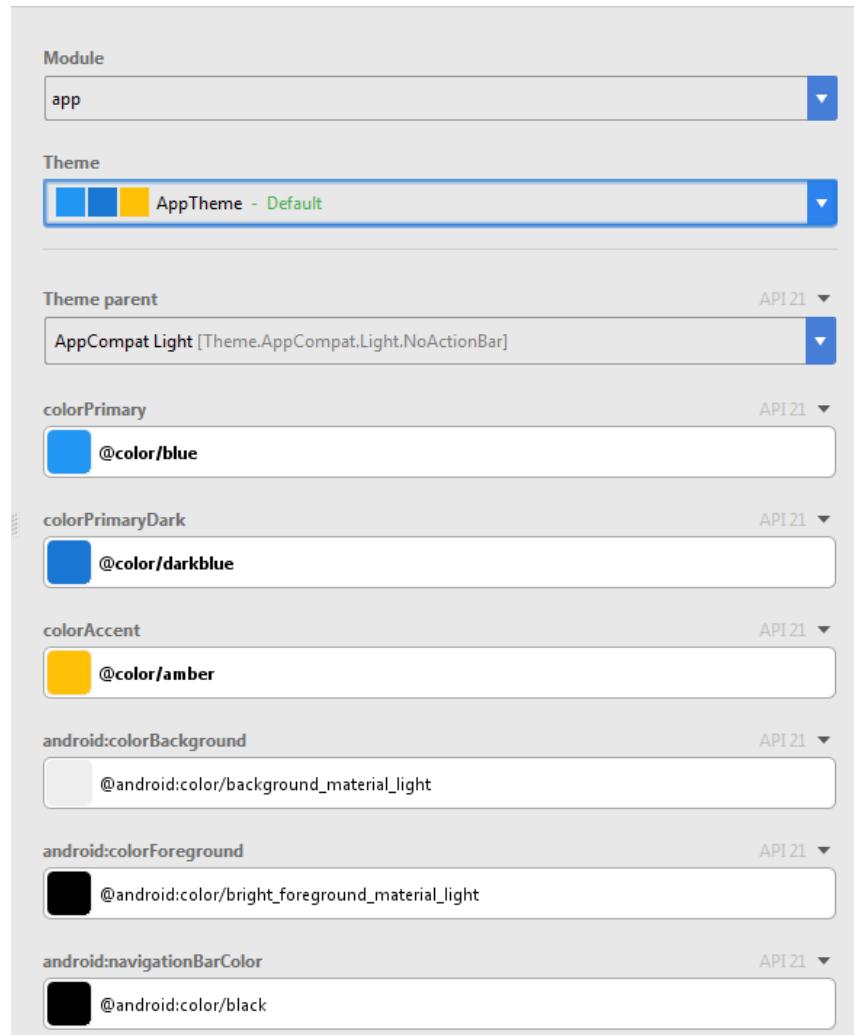
- Per applicare uno style:

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />
```

- Per applicare un theme:

```
<activity android:theme="@style/CustomTheme">
```

Editor di stili e temi in Android Studio

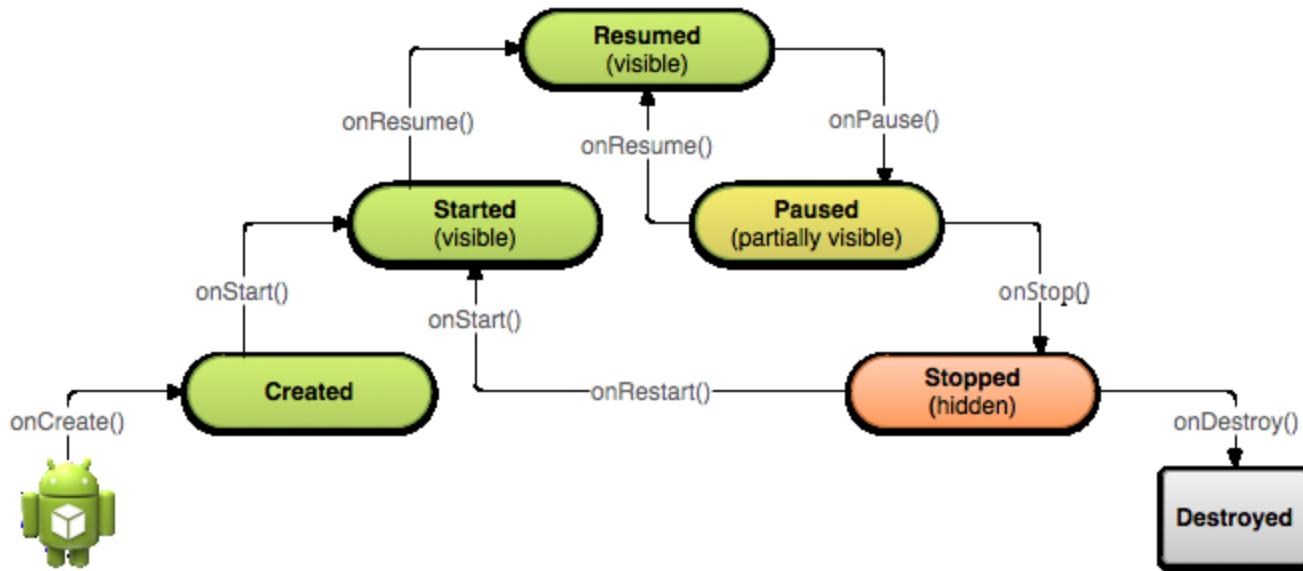


Application components: Activity

Activity

- **Una activity in Android rappresenta una singola schermata di una applicazione interattiva**
 - Da non confondere con il concetto di UML Activity
- **Una applicazione può avere diverse activity**
- **Una sola activity per volta può essere sullo schermo**
 - A differenza dei sistemi operativi per PC, non è prevista la possibilità di avere più finestre aperte contemporaneamente
- **Al passaggio da una Activity ad un'altra, l'activity esistente viene messa in pausa**
- **Una activity è implementata come una classe che eredita dalla classe Activity della quale poi istanziare un oggetto**

Ciclo di vita di una Activity



Le activity non ‘running’ vanno ad accodarsi in uno stack, pronte per essere rimesse in foreground sul video del dispositivo

Activity

- Per istanziare una activity è necessario dichiarare una classe che la estenda
 - public class myActivity extends Activity
- La classe così creata potrà ridefinire (per override) alcuni metodi di Activity, tra cui quelli relativi alla gestione del suo ciclo di vita:
- **onCreate(Bundle savedInstanceState)** - Eseguito al primo avvio dell'activity
- **onDestroy()** - Eseguito alla chiusura e deallocazione dell'activity
- **onPause()** - Eseguito quando l'activity smette di essere in primo piano (foreground), messa in secondo piano da un'altra Activity
- **onResume()** - Eseguito quando l'Activity ritorna in primo piano (foreground)
- **onStop()** - Eseguito quando l'Activity viene sostituita da un'altra (ma è ancora istanziata in memoria)
- **onRestart()** – Eseguito quando l'Activity viene riavviata
- Il ripristino di una Activity dopo una pausa è reso possibile dalla chiamata, automatica, a **onSaveInstanceState()** che utilizza l'oggetto Bundle passato da **onCreate** per mantenere le informazioni necessarie al ripristino

Intent 1/2

- **Una activity Android non può indiscriminatamente chiamare altre activity**
 - Questo meccanismo esiste sia per ragioni di sicurezza, sia per favorire il riuso di componenti, che viene mediato dal sistema
- **Col termine Intent si definisce un oggetto corrispondente ad un messaggio col quale si richiede l'attivazione di una Activity (o anche un servizio o un receiver)**
 - Ad esempio come parametro di **startActivity** o **startActivityForResult**

Intent 2/2

- Il modo più semplice per avviare da programma un'altra Activity è con explicit intent:

```
Intent intent = new Intent(this, MiaActivity.class);  
startActivity(intent);
```

- Se, invece, vogliamo far partire un'activity che svolga un particolare compito senza conoscere staticamente la classe che la implementa (implicit intent):

```
Intent intent = new Intent(Intent.ACTION_SEND);  
startActivity(intent);
```

- In questo caso si chiede di avviare una activity che abbia settato il filtro ACTION_SEND
- Per passare dati da una Activity ad un'altra si può utilizzare il metodo putExtra di Intent (passaggio per valore)**

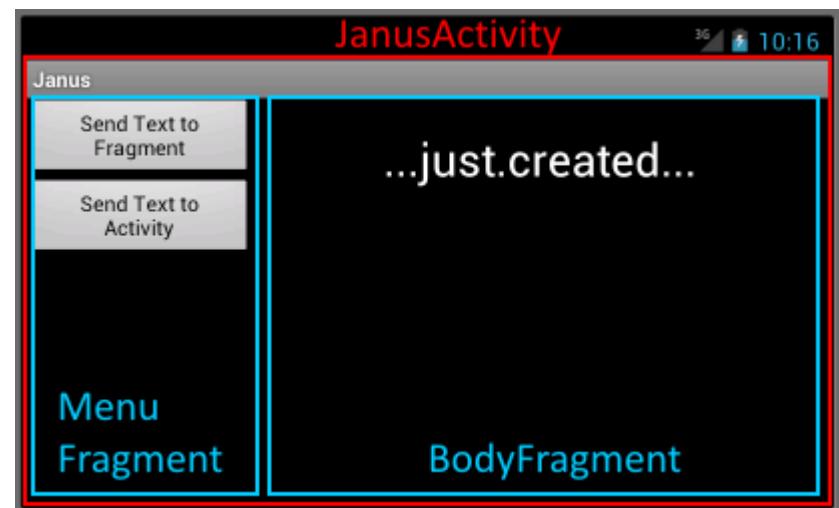
Cenno ai Fragment

- **Android non è un ambiente a finestre**
 - Si tratta di una scelta progettuale motivata dalla necessità di poter essere installato con processori poco performanti, scarsa memoria e schermi video molto piccoli
- **A partire dalle versione 3.0, la diffusione dei tablet ha portato i progettisti a proporre una soluzione parziale al problema: i Fragment**
- **I Fragment somigliano molto ai Frame che potevano comporre una pagina Web (frameset)**

Cenno ai Fragment

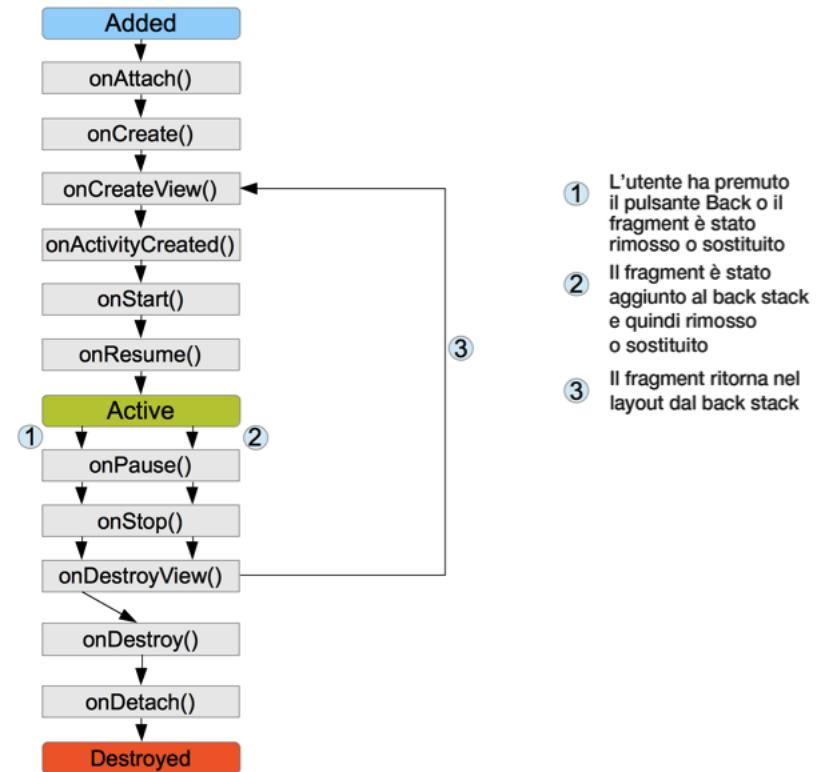
- A differenza di due Activity, due o più fragment condividono lo stesso spazio di memoria comune, quindi è possibile per un fragment leggere e modificare gli elementi degli altri fragment della stessa activity

- Ogni fragment ha il suo layout e il suo ciclo di vita



Ciclo di vita di un fragment

- Il ciclo di vita di un Fragment si posiziona all'interno del ciclo di vita di una Activity, avendo ulteriori metodi suoi propri



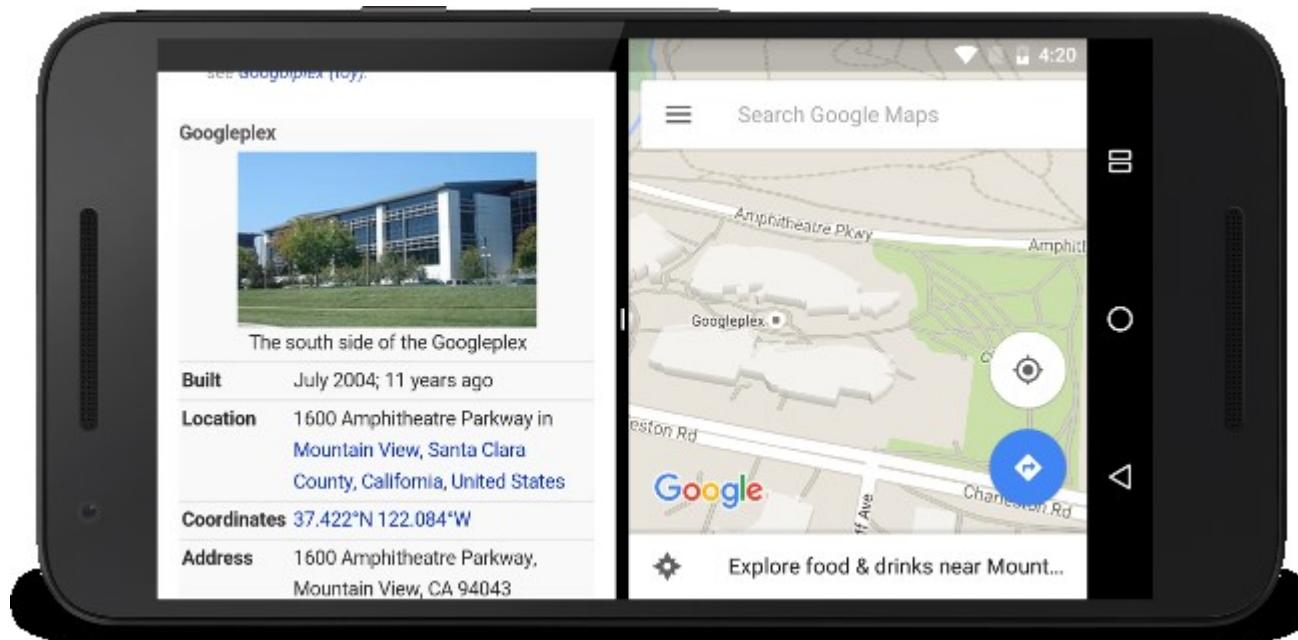
Cenno al Multi-Window

- **A partire da Android 7.0, si cerca di supportare anche finestre multiple sullo schermo**
 - Ma una sola delle finestre (quella col focus) risulta running!
 - Il ciclo di vita delle activity rimane quindi lo stesso
 - Ad ogni resize della window attiva corrisponde una coppia di pause/restart (così come quando si passa da portrait a landscape, e viceversa)

<https://developer.android.com/guide/topics/ui/multi-window.html>

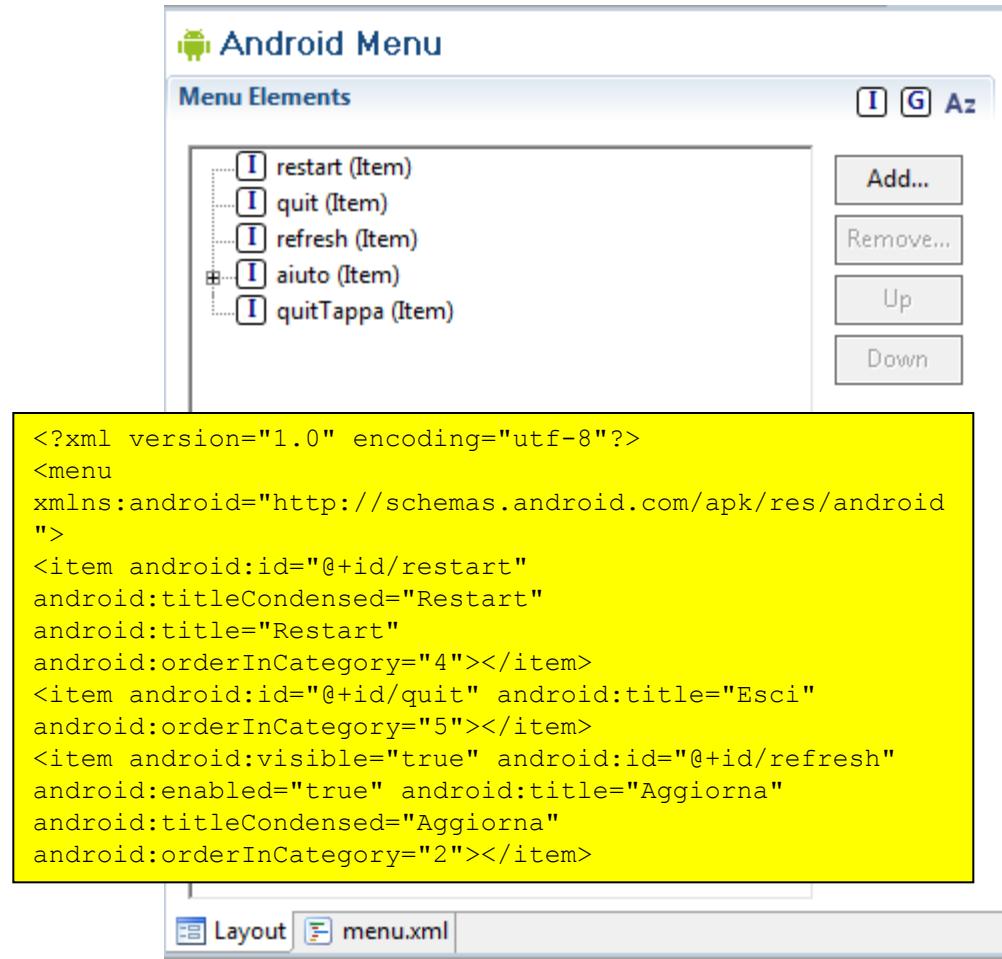
Cenno al Multi-Window

- Il programmatore deve esplicitamente dichiarare se l'applicazione può agire in modalità multi-window, se e come può essere ridimensionata e se sia supportato il drag & drop



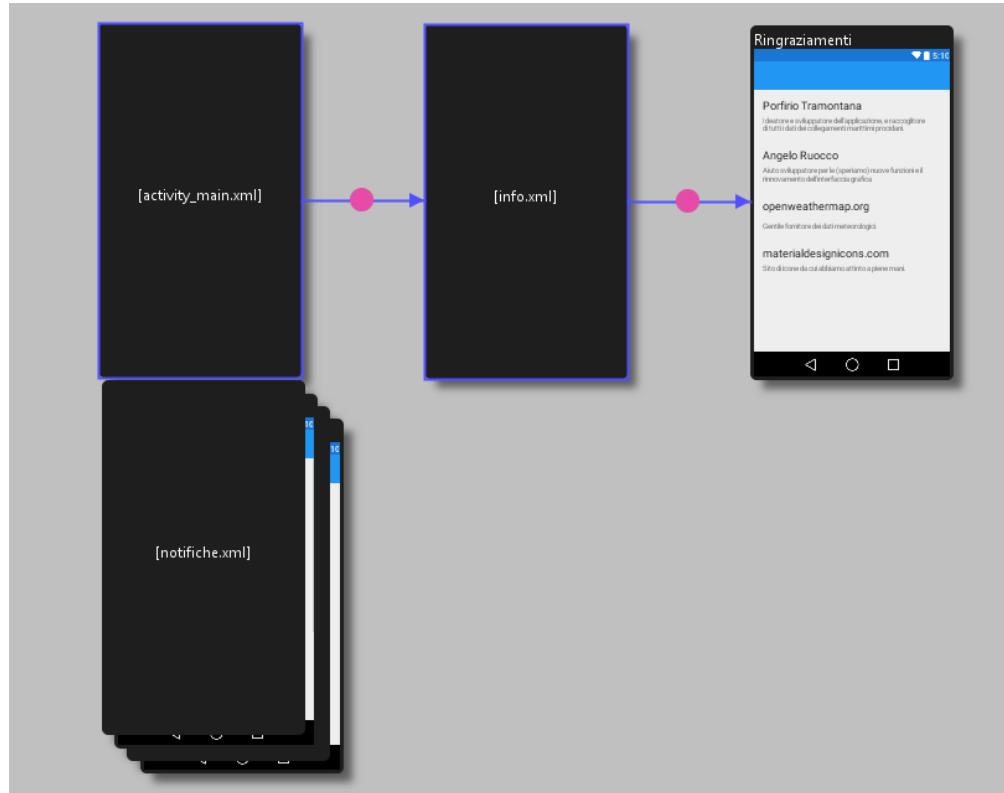
Menu

- **Anche i menu possono essere scritti in XML o con l'ausilio dello strumento visuale**
- **In alternativa, potrebbero essere istanziati dinamicamente nel codice**
- **L'editor di menu è disponibile solo in ADT: in Android Studio va editato direttamente l'XML**
- **I menu stanno diventando obsoleti, con le nuove versioni di Android, sostituiti da pulsanti esplicativi in fragment laterali, come implementato di default dai layout consigliati**



Navigation Editor in Android Studio

- Attivabile da Tools/Android/Navigation Editor



Event Delegation

- La gestione dei menu in Android è realizzata con il pattern Event Delegation
- Dal momento che una sola Activity per volta può essere visualizzata e una Activity può avere un solo menu, il codice relativo all'handling dei menu è contenuto nell'Activity
- Se fosse esistita una classe menu, sarebbe stato possibile assegnarle dei metodi evento da ridefinire. Viceversa, esiste:
 - un unico metodo onCreateOptionsMenu(Menu menu) dell'activity che ne crea il menu
 - Un unico metodo onOptionsItemSelected(MenuItem item) che associa a qualsiasi opzione del menu il suo codice di handling
 - Questi due metodi hanno funzione di Delegati per la creazione/scelta di un qualsiasi elemento del menu

Event Delegation: esempi

```
//Genera un menu leggendo la composizione nel file R, automaticamente  
// generato a partire dalle informazioni in menu.xml  
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.menu, menu);  
    return true;  
}  
  
// Parte dalla selezione di un item dal menu dell'activity. Tramite il parametro item viene passato  
// l'identificatore della voce di menu scelta  
public boolean onOptionsItemSelected(MenuItem item) {  
    // Handle item selection  
    switch (item.getItemId()) {  
        case R.id.restart:  
            ...  
            return true;  
        ...  
    default:  
        return super.onOptionsItemSelected(item);  
    }  
}
```

Dialog

- **La gestione delle finestre di dialogo rappresentano un altro esempio di event delegation**
 - Un Dialog è una interfaccia *modale* che va in primo piano rispetto all'activity cui appartiene, e vi rimane fino a che l'utente non ha specificato una risposta
 - Ad esempio, il messaggio di conferma che si fornisce ad un utente che sta cercando di uscire dal programma
- **In Android non esiste la possibilità di dichiarare i dialog via xml, per cui devono essere generati dinamicamente**
 - Il layout di un dialog, personalizzato, però, può essere realizzato allo stesso modo dell'interfaccia di un activity
- **I dialog non sono activity, ma suoi attributi, cui possono essere agganciati dei gestori degli eventi, la cui delega per l'esecuzione spetta ancora all'Activity**

Dialog: esempio 1/2

Tutto il codice seguente si pone nella classe che eredita da Activity

```
// dichiarazione della costante identificativa e dell'oggetto AlertDialog
static final int QUIT_DIALOG_ID = 1;
AlertDialog quitDialog;

// costruzione dell'AlertDialog tramite la Factory AlertDialog.Builder
AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setMessage("Sei sicuro di voler abbandonare?")
        .setCancelable(false)
        .setPositiveButton("Si", new DialogInterface.OnClickListener() {
            // codice associato al pulsante Si
            public void onClick(DialogInterface dialog, int id) {
                CacciaAlTesoro.this.finish();
            }
        })
        .setNegativeButton("No", new DialogInterface.OnClickListener() {
            // codice associato al pulsante Si
            public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
            }
        });
quitDialog = builder.create(); //istanziazione dell'oggetto quiDialog
```

Dialog: esempio 2/2

```
// Override del metodo onCreateDialog di Activity, responsabile della  
// visualizzazione in foreground del Dialog  
protected Dialog onCreateDialog(int id) {  
    switch (id) {  
case QUIT_DIALOG_ID:  
    return quitDialog;  
...  
  
// Chiamata del metodo showDialog, che scatena la visualizzazione del  
// Dialog. In questo caso il dialog è legato alla pressione di una voce dal menu  
public boolean onOptionsItemSelected(MenuItem item) {  
    // Handle item selection  
    switch (item.getItemId()) {  
    case R.id.quit:  
        showDialog(QUIT_DIALOG_ID);  
        return true;
```

Toast

- **Il Toast è il modo più semplice e immediato per mostrare un messaggio sul video del device per un tempo limitato**
 - `Toast.makeText(getApplicationContext(), "Hello toast!", Toast.LENGTH_SHORT).show();`
- **E' possibile:**
 - settare il tempo di visualizzazione («breve» o «lungo»)
 - Settare il messaggio da visualizzare
 - Settare la posizione in cui visualizzarlo
 - Specificare il layout del Toast

Gestione della persistenza

- **Ogni applicazione Android gira in un suo processo, con un proprio spazio dati e una propria porzione di file system**
- **Quali opportunità sono messe a disposizione per la gestione di dati persistenti?**
 - Persistenza nel tempo tra istanze della stessa app:
 - File
 - Shared Preferences
 - Database SQLite
 - Persistenza tra applicazioni diverse
 - Content Provider
 - Accesso a risorse esterne (ad esempio Web)

SharedPreferences

- **Una tecnica, basata sui file, per la memorizzazione di dati basilari**
 - Create per salvare «preferenze» intese come informazioni di configurazione, che dovessero rimanere persistenti tra diversi utilizzi della stessa app sullo stesso dispositivo
 - Basate su file memorizzati nella porzione di file system dedicata ad una singola app nell'ambito di un dispositivo
 - Dati organizzati come valori scalari
-

SharedPreferences

- **Esempio di lettura**

```
private SharedPreferences settings;  
settings = getSharedPreferences("RecordFile", 0);  
min = settings.getInt("record", 1000);
```

- **Esempio di scrittura (salva record)**

```
SharedPreferences.Editor editor = settings.edit();  
editor.putInt("record", min);  
editor.commit();
```

Accesso ai file

- **Le funzioni di libreria per l'accesso ai file utilizzabili in Android sono essenzialmente quelle utilizzate in Java**
 - Ad esempio tramite le classi FileInputStream e FileOutputStream
- **Ogni applicazione ha una sua porzione di file system**
 - Tramite DDMS possiamo esplorare il file system e trovare i file nella cartella data/data
- **In aggiunta, tutte le applicazioni possono scrivere su di una memoria esterna (ad esempio SD card)**
 - Per sapere l'indirizzo nel file system della SD card:
 - Environment.getExternalStorageDirectory()

Database SQLite

- Si possono creare veri e propri database all'interno di una app Android utilizzando il DBMS SQLite
 - Scelto per le sue dimensioni molto contenute, meno di 500 kB
- Le funzioni per accedere al database sono nei package
 - android.database;
 - android.database.sqlite.
- I database SQLite sono memorizzati in file con estensione .db in una sottocartella database della cartella del file system della app
- SQLite mette a disposizione la maggior parte dei metodi messi a disposizione da un generico DBMS SQL compatibile
 - Ad esempio, è possibile salvare insiemi di query o istruzioni DDL di SQL in un file con estensione .sql ed eseguirli con apposite istruzioni della libreria android.database.sqlite, come execSQL
 - Supportati anche i cursori (Cursor)
 - Supportata anche la gestione delle transazioni

Content Provider

- **Un Content Provider è un componente che fa da interfaccia verso una sorgente di dati persistente.**
- **Un Content Provider consente il disaccoppiamento tra gestione dei dati e resto dell'applicazione**
- **Un Content Provider rende possibile la realizzazione di fonti dati persistenti condivise tra più applicazioni**
 - Ad esempio la rubrica dei contatti
- **Un Content Provider realizza una separazione concettuale tra fornitore dei dati (Content Provider) e livello di accesso ai dati**
 - Spesso i dati cui accede un Content Provider sono database SQLite

Content Provider

- **Un Content Provider è un componente di una applicazione dichiarabile in modo analogo a Activity, Service e Broadcast Receiver**
- Attraverso la classe **ContentResolver**, una qualsiasi app può accedere ad un **content provider**, identificato da una **URI**
- **Nell'implementare un content provider bisogna:**
 - Descrivere l'elenco di servizi (o dati) forniti (in maniera analoga a quanto si farebbe con servizi REST)
 - Implementare il livello di accesso ai dati in grado di fornire tali servizi

Ulteriori approfondimenti

Thread in Android

- **I Thread in Android seguono le stesse regole di base dei Thread in Java**
 - Ad esempio, un Thread può essere creato estendendo la classe Thread oppure definendo un'implementazione della classe Runnable e istanziando un oggetto della classe Thread che la utilizzi
- **Le Activity sono eseguite nell'ambito di un Thread di base denominato UIThread**

Thread e Task

- Un thread è una successione di operazioni che agiscono su informazioni che possono essere condivise con altri thread. Un thread potrebbe essere eseguito per un tempo indefinito (ad esempio la connessione con una socket)
 - L'interfaccia Runnable è utilizzata per i Thread
- Un task è invece una successione di operazioni che prima o poi si completa e che produce di solito un risultato (ad esempio un download)
 - i task vengono solitamente eseguiti all'interno di particolari thread.
 - L'interfaccia Callable è utilizzata per i task.

AsyncTask

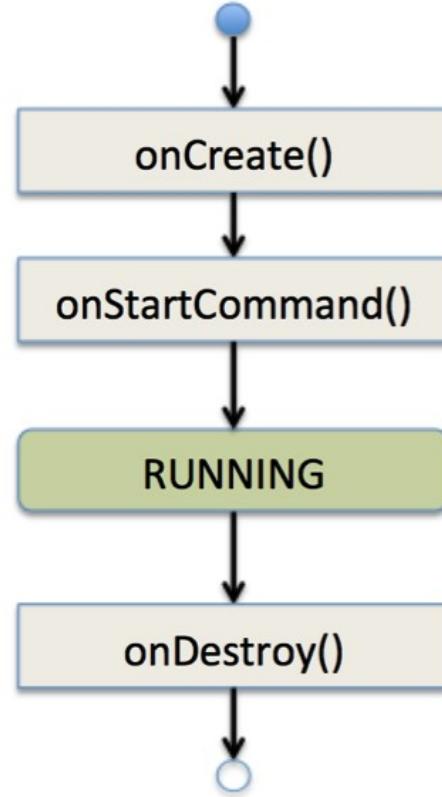
- **I task possono essere eseguiti in Android nell'ambito della classe AsyncTask**
 - Metodi principali di cui fare l'override:
 - onPreExecute
 - onPostExecute
 - onCancelled
 - onProgressUpdate
 - doInBackground
 - E' il metodo principale e viene eseguito in un thread diverso (anonimo) da quello in cui viene istanziato l'oggetto (a differenza degli altri metodi)
- **Il task non può essere messo in pausa ma solo distrutto (cancelled)**
- **Il ciclo di vita dell'AsyncTask è comunque legato sempre a quello dell'Activity che lo ha avviato**

Services

- **I Services sono componenti responsabili di esecuzioni in background, senza possibilità di interagire con l'interfaccia utente**
 - eccetto che tramite il Notification Manager, che consente di scrivere sulla barra delle notifiche, emettere suoni o vibrazioni
- **Più service possono essere eseguiti in concorrenza tra loro e con una Activity**
- **Un service è implementato come una classe che eredita dalla classe Services della quale poi istanziare un oggetto**
 - Di solito si effettua l'override di due metodi fondamentali: startService e stopService
 - Un service può avviare un Intent (e tramite esso un'Activity)
 - Un service può essere avviato tramite un Intent

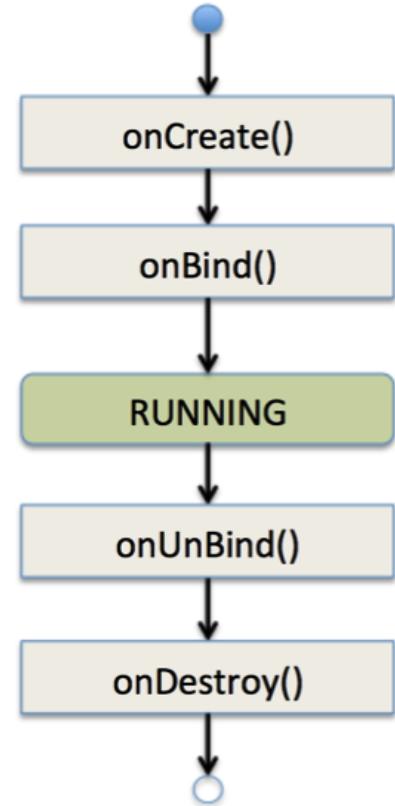
Servizi «Started»

- **Un servizio può essere avviato in modalità «Started» da una classe («client») che lo istanzia, richiamando il metodo `startService()` e può controllarlo**
 - Può anche fermarlo con `StopService`
 - Il servizio stesso può anche autofermarsi con `stopSelf`



Servizi «Bounded»

- In aggiunta, un service può essere collegato ad una classe client tramite il metodo bindService (noto l'Intent e un riferimento al servizio avviato)
 - Per disconnettersi da un servizio si può richiamare unbindService
- Un servizio viene distrutto solo se il suo iniziatore lo stoppa e non c'è più alcun client ad esso collegato
- Un service è inizialmente pubblico ma è possibile limitarne la visibilità attraverso un opportuno attributo in fase di dichiarazione nel file **AndroidManifest.xml**.



Notification Service

- **Tramite le notifiche, una qualunque Activity o Service può interagire con la Notification Area (in alto nel display), aprendo la quale si accede al Notification Drawer**
- **Al momento (è stato oggetto di parecchie revisioni, al variare delle versioni di Android) è possibile gestire notifiche tramite la classe NotificationCompat.Builder**
- **E' possibile associare ad ogni notifica testo, icona ed altre caratteristiche grafiche e anche un Pending Intent tramite il quale specificare cosa viene fatto selezionando, ad esempio, la notifica**

Esempio di Notification

- **Esempio: invio di una notifica con testo, icona e titolo**

```
mNotificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE) ;

Notification notification = new
NotificationCompat.Builder(this)
.setSmallIcon(R.drawable.ic_launcher)
.setContentText(contentText)
.setContentTitle(contentTitle) .build() ;

mNotificationManager.notify(VERY_SIMPLE_NOTIFICATION,
notification) ;
```

Broadcast Receiver

- **Un Broadcast Receiver è un ascoltatore di eventi di sistema (Intent), che può partire appena uno di essi arriva**
 - In analogia con una ISR
- **Ad un Broadcast Receiver è associato un piccolo spezzone di codice che comprende di solito chiamate ad Activity o Services**
 - da eseguire in un tempo limite, oltre il quale il sistema deduce che ci sia stato uno stallo
- **Sono dichiarati anch'essi nel Manifest**
 - Un Broadcast Receiver può anche essere attivato da codice col metodo Context.sendBroadcast()

Broadcast Receiver

- Per rispondere ad una chiamata in broadcast un componente deve registrarsi come *receiver* nel manifest (oppure tramite chiamata del metodo *registerReceiver*)
- Alla chiamata in Broadcast (con *sendBroadcast*) tutti i ricevitori registrati vengono avviati
- Se si vuole tener conto di una priorità nelle chiamate bisogna utilizzare il metodo *sendOrderedBroadcast*
 - Verranno considerate le priorità definite tramite l'attributo *priority* nel manifest

Content Provider

- **Un Content Provider è un componente che fa da interfaccia verso una sorgente di dati persistente.**
- **Un Content Provider consente il disaccoppiamento tra gestione dei dati e resto dell'applicazione**
 - In Android sono disponibili librerie per l'interfacciamento con SQLite, un dbms estremamente leggero basato su SQL
 - Un db SQLite serve unicamente per salvare dati persistente sulla memoria del dispositivo
 - In alternativa, possiamo utilizzare file di testo per implementare database XML
 - L'utilizzo di un Content Provider NON è necessario ma facilita astrazione, riuso e testing

Accesso a risorse Web

- **Una applicazione Android può accedere in numerosi modi a risorse remote**
- **Solo a titolo di esempio:**
 - Con il widget WebView è possibile aprire all'interno di una Activity l'equivalente di un browser
 - Con il widget MapView è possibile accedere dall'interno di una Activity alla mappe di Google
 - È possibile eseguire richieste http, ad esempio verso servizi Web

Esempio: riuso di un servizio meteo

- **Caso 1) Il servizio meteo è disponibile tramite richieste http**
 - In risposta viene restituito un XML
 - Bisogna scrivere un analizzatore in grado di leggere dati da un XML
 - SAX, DOM, ...

Codice SAX (1/2)

```
try {
    url = new URL("http://www.google.com/ig/api?weather=Procida");
/* Get a SAXParser from the SAXParserFactory. */
    SAXParserFactory spf = SAXParserFactory.newInstance();
    SAXParser sp = spf.newSAXParser();
/* Get the XMLReader of the SAXParser we created. */
    XMLReader xr = sp.getXMLReader();
/* Create a new ContentHandler and apply it to the XML-Reader*/
    MeteoXMLHandler meteoXMLHandler = new MeteoXMLHandler(this);
    xr.setContentHandler(meteoXMLHandler);
/* Parse the xml-data from our URL. */
    xr.parse(new InputSource(url.openStream()));
/* Parsing has finished. */
} catch (MalformedURLException e) {e.printStackTrace();
} catch (ParserConfigurationException e) {e.printStackTrace();
} catch (SAXException e) {e.printStackTrace();
} catch (IOException e) {e.printStackTrace();
}
```

Codice SAX (2/2)

```
package com.porfirio.orariprocida2011;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
...
public class MeteoXMLHandler extends DefaultHandler{
    StringBuffer buff = null;      boolean buffering = false;
@Override
    public void startDocument() throws SAXException {    }
@Override
    public void endDocument() throws SAXException {    }
@Override
    public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException
    {
        if (localName.equals("wind_condition")) {
            /** Get attribute value */
            String attr = attributes.getValue("data");
            setMeteo(attr);
        }
    }
@Override
    public void endElement(String uri, String localName, String qName) throws SAXException {    }
...
}
```

- **Caso 2) Il servizio meteo è disponibile sotto forma di json raggiungibile via http**
 - Bisogna scrivere un analizzatore di file json

Esempio di risposta json

```
{  
  "response":{  
    "version":"0.1",  
    "features":{  
      "conditions":1  
    }  
  },  
  "current_observation":{  
    "image":{  
      "url":"http://icons-ak.wxug.com/graphics/wu2/logo_130x80.png",  
      "title":"Weather Underground",  
    },  
    "display_location":{  
      "full":"San Francisco, CA",  
      "city":"San Francisco",  
      "state":"CA",  
    },  
  },  
  ...  
}
```

Codice per json

```
import org.json.JSONException;
import org.json.JSONObject;
import org.json.simple.*;
private void leggiMeteo() {
URL url;
JSONObject jsonObject=null;
try {
jsonObject =
    readJsonFromUrl("http://api.wunderground.com/api/7a2bedc35ab44ecb/geolookup/conditions/q/IA/Procida.json");
if (!(jsonObject==null)){
    meteo.setWindKmh((Double) jsonObject.getJSONObject("current_observation").get("wind_kph"));
    Integer windDir=(Integer) jsonObject.getJSONObject("current_observation").get("wind_degrees");
    meteo.setWindDirectionString((String) jsonObject.getJSONObject("current_observation").get("wind_dir"));
    meteo.setWindBeaufort((Double) jsonObject.getJSONObject("current_observation").get("wind_kph"));
} catch (JSONException e) {Log.d("ORARI", "dati meteo non caricati da web");}
}
```

Servizi di Sistema

- **Il sistema Android fornisce classi e servizi per accedere in lettura e/o comandare un po' tutti i sensori a disposizione:**
 - Power Service
 - Vibrator Service
 - Alarm Service
 - Sensor Service
 - Audio Service
 - Telephony Service
 - Wi-Fi Service
 - ...
-

Sensor Service

- **Tramite la classe SensorManager è possibile accedere alla maggior parte dei sensori:**
 - `SensorManager sm = (SensorManager) getSystemService(Context.SENSOR_SERVICE)`
- **Per conoscere l'insieme dei sensori disponibili:**
 - `getSensorList (type)`
 - Type può essere:
 - `TYPE_ACCELEROMETER`
 - `TYPE_GYROSCOPE`
 - `TYPE_LIGHT`
 - `TYPE_MAGNETIC_FIELD`
 - `TYPE_ORIENTATION`
 - `TYPE_PRESSURE`
 - `TYPE_PROXIMITY`
 - `TYPE_TEMPERATURE`
 - ...

Esempio: Sensore di luminosità

- **Aggiungiamo un altro TextView sul layout**
- **In onCreate aggiungiamo:**

```
txtSensore= (TextView)findViewById(R.id.textView2);
sensorService = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
sensor = sensorService.getDefaultSensor(Sensor.TYPE_LIGHT);
if (sensor != null) {
    sensorService.registerListener(mySensorEventListener, sensor,
        SensorManager.SENSOR_DELAY_NORMAL);
}
```

- **In questo modo abbiamo creato un campo di testo e un servizio di ascolto del sensore luminosità, legato alla classe mySensorEventListener**
-

Classe mySensorEventListener

```
private SensorEventListener mySensorEventListener = new SensorEventListener()
{
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        float luce = event.values[0];
        txtSensore.setText(Double.toString(luce));
    }
};
```

- Quando cambia il valore del sensore lo leggiamo (è in `event.values[0]`) e lo visualizziamo in `txtSensore`

Gestione del GPS

- E' possibile conoscere i dati sulla posizione tramite la classe Location
- Nel manifest bisogna dare il permesso all'applicazione di accedere a dati sulla posizione
 - Tale permesso sarà notificato all'utente che installerà l'app, il quale potrebbe anche decidere di non installarla, se teme per la privacy dei propri dati
 - Si può distinguere tra tre tipologie di dati posizionali:
 - FINE LOCATION, a massima precisione (ad esempio GPS)
 - COARSE LOCATION, con precisione minore (ad esempio dati sulla posizione della cella oppure dati provenienti da ripetitori wi-fi)
 - MOCK LOCATION, che forniscono dati finti, utili a scopo di testing
- Tramite getSystemService si può ottenere un oggetto LocationManager
 - L'oggetto LocationManager può essere interrogato per ottenere informazioni sincrone sulla posizione

Gestione del GPS

- Una Activity può implementare l'interfaccia LocationListener. In tal caso definirà metodi come:
 - startListening (nel quale di solito si registra come ascoltatore di specifiche informazioni Location)
 - `private void startListening()
myManager.requestLocationUpdates(LocationManager.GP
S_PROVIDER, 0, 0, this);`
 - stopListening
 - onLocationChanged (che viene chiamato ogni volta che si osserva una variazione della location, e quindi consente un aggiornamento asincrono delle informazioni sulla posizione)
 - `public void onLocationChanged(Location location) { ... }`
 - Sull'oggetto Location restituito si possono chiamare metodi come `getLongitude` e `getLatitude`

Play Services

- Per accedere alle informazioni su GPS, mappe e altro, nelle versioni più recenti è necessario includere le librerie Play Services in maniera separata
 - Nelle prime versioni esistevano versioni Android con e senza Google libraries
 - Nello sviluppare un progetto le librerie vengono incluse dal progetto e aggiunte come un extra progetto che viene incluso come libreria (con Eclipse dalle proprietà del progetto)
 - Su di un emulatore, vengono installate anche le librerie
 - Su di un dispositivo potrebbe essere necessario disinstallare le librerie esistenti, se diverse
 - Nei Play Services è dichiarata una più completa versione del LocationListener visto prima

Mappe e Localizzazione

- **Ci sono diversi modi per poter accedere a informazioni geografiche:**
 - Accesso alla posizione fornita dal GPS o da altro fornitore di posizione
 - Accesso ai dati delle Google Maps

Accesso alle Google Maps

- Per utilizzare le Google Maps in un'applicazione è necessario prima abilitarne l'utilizzo lato server
 - Dall'indirizzo <https://code.google.com/apis/console> è possibile abilitare le Google Maps (o qualsiasi altra API) per un proprio progetto
 - L'utilizzo delle API è soggetto ad alcune limitazioni decise da Google relative a numero e tipologia di utilizzi
 - Il sistema fornisce una API Key avendo ricevuto in input il codice SHA-1 che identifica l'applicazione richiedente
 - In modalità di debugging, un codice può essere trovato eseguendo:
 - keytool -list -v -keystore debug.keystore -alias androiddebugkey -storepass android -keypass android

Accesso alle Google Maps

- **La API Key va scritta nel manifest**
 - <meta-data android:name="com.google.android.maps.v2.API_KEY" android:value="API_KEY"/>
- **Inoltre è necessario abilitare obbligatoriamente altri permessi:**
 - <uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>

Accesso alle Google Maps

- Una vista delle mappe di Google è inseribile all'interno di una estensione di fragment, detta **MapFragment**
- E' possibile visualizzare diversi tipi di mappa
 - Normal – Hybrid – Satellite – Terrain
- La mappa può essere «centrata» sulla posizione corrente semplicemente prendendola dall'oggetto Location
- Esistono ulteriori metodi per zoomare, scrollare, ruotare la mappa, inserire in essa dei marker, disegnare figure sulla mappa, gestire gli eventi utente sulla mappa
- E' possibile ascoltare per eventi relativi all'avvicinamento dell'utente ad una zona della mappa (Geofence)

Android e Sicurezza

- **Android è un sistema relativamente sicuro poiché molte caratteristiche di sicurezza sono state nativamente impiantate già all'interno del sistema operativo e dell'ambiente di esecuzione**
 - Android è open source: chiunque può manomettere il sistema superando queste impostazioni di sicurezza e rendendolo insicuro.
 - Ci riferiamo qui alle caratteristiche dei sistemi Android originali, con applicazioni realizzati secondo le linee guida Google (senza le quali non è possibile la pubblicazione su Android Market)

Sicurezza in Android

- **Android Application Sandbox**
 - Ogni applicazione viene eseguita in un ambiente isolato, dotato di proprio spazio dati, codice, e porzione di file system locale
- **Alcune tecniche a livello di sistema operativo prevengono errori di gestione della memoria**
- **Le caratteristiche fondamentali del sistema non possono essere modificate da utenti con permessi normali né da applicazioni non di sistema**

Principali problemi di sicurezza

- File memorizzati nella memoria interna di una applicazione ma con le modalità **MODE_WORLD_WRITEABLE** o **MODE_WORLD_READABLE** potrebbero essere modificati da altri processi
- File creati su memorie esterne (ad esempio SD Card) sono visibili a chiunque
- Content Provider possono essere resi utilizzabili a più applicazioni (con l'opzione `android:exported=true`): in tal caso non sono disponibili opzioni per controllare i diritti d'accesso applicazione per applicazione

Principali problemi di sicurezza

- **Tramite il sistema dei permessi una applicazione dichiara tutte le tipologie di feature che utilizzerà: richiedere troppi permessi semplifica lo sviluppo ma rende l'utente meno fiducioso**
 - In particolare, per quanto riguarda i permessi di accesso ai dati personali dell'utente salvati sul dispositivo
- **L'accesso a Internet (ad esempio tramite le WebView) pone in essere tutti i classici rischi di sicurezza che si hanno utilizzando ad esempio un browser Web su di un pc (ad esempio phishing, adware, ecc.)**
- **Una applicazione che legge SMS ed esegue operazioni in base ad esso dovrebbe fare attenzione a validarne i contenuti, poiché non ci può essere sempre fiducia nella loro provenienza**

Principali problemi di sicurezza

- **La Input validation è necessaria in Android come in ogni altro sistema con interfaccia utente**
 - Android fornisce alcuni meccanismi per prevenire, ad esempio, buffer overflows
 - E' il programmatore a dover fare attenzione ad altre tipologie di vulnerabilità, come ad esempio cross site scripting su javascript (ad esempio webView) e SQL injection (ad esempio Content Provider)

Principali problemi di sicurezza

- **Spesso le applicazioni possono collegarsi tramite Intent in maniera dinamica ad altre applicazioni che si propongono di riceverli.**
 - Attenzione alle applicazioni maliziose: una volta installate potrebbero in maniera semi automatica (una sola abilitazione dell’utente è necessaria) qualificarsi per poter interagire con altre applicazioni
 - E’ possibile specificare nell’Intent che solo un componente della stessa applicazione possa rispondere
- **Per interazioni mutue tra applicazioni (ad esempio scambio di messaggi RPC) è consigliato l’utilizzo dei meccanismi Binder o Messenger che prevedono una mutua autenticazione tra i componenti dialoganti**

Principali problemi di sicurezza

- **In teoria le app possono caricare codice a tempo di esecuzione da fonti diverse all'APK**
 - Questa pratica è rischiosa, poiché il codice caricato dinamicamente ha gli stessi diritti d'accesso di quello statico: un'infiltrazione potrebbe causare l'esecuzione di codice arbitrario su applicazioni accettate come sicure
- **Le caratteristiche di sicurezza della Dalvik virtual machine sono essenzialmente simili a quelle di una JVM**
- **Le caratteristiche di sicurezza del kernel linux sono quelle classiche**

Tutorial

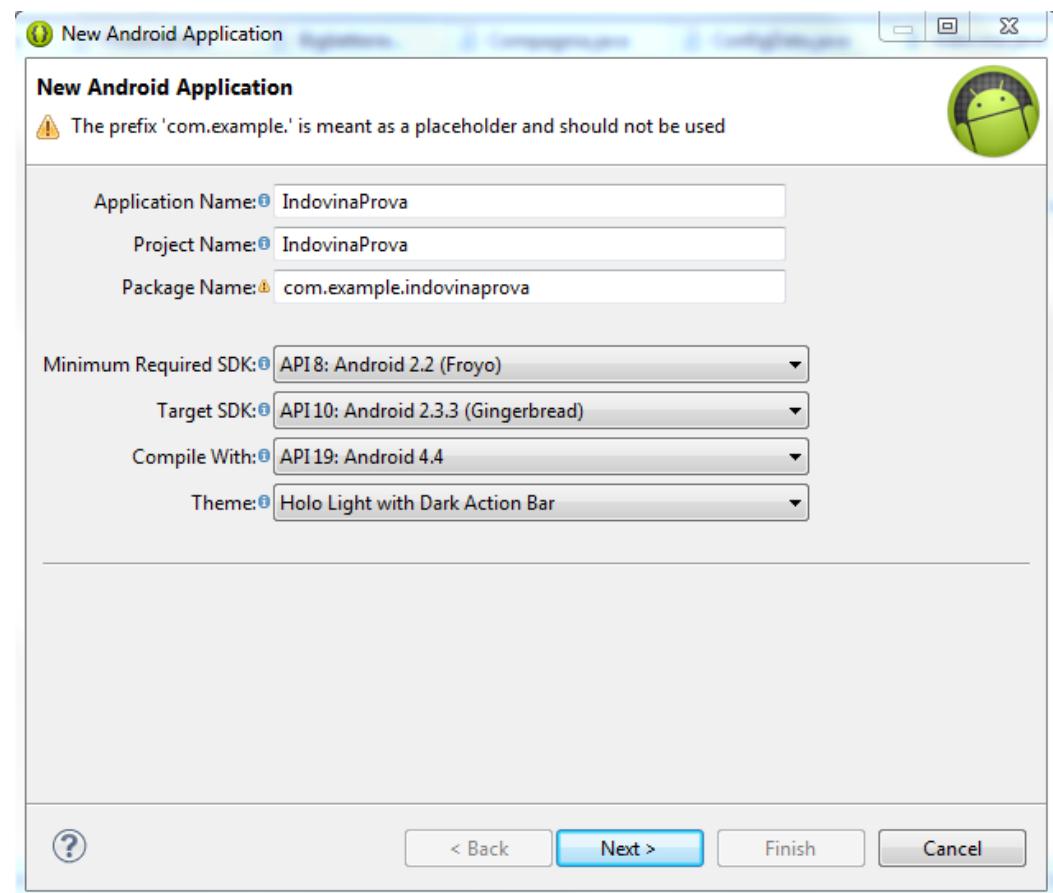
Tutorial: IndovinaOraDelitto (versione ADT)

- **Si tratta di un progetto molto più piccolo**
 - Piccolo gioco nel quale bisogna indovinare “l’orario del delitto” (un orario scelto a caso nelle 24 ore)
 - Ad ogni tentativo, il sistema risponde dicendo se l’orario esatto è precedente o successivo
 - Un’unica interfaccia → un’unica Activity
 - Gestione delle Shared Preferences (per memorizzare il punteggio migliore)
 - Riuso di un componente (widget) per la richiesta dell’orario
 - Visualizzazione di un’immagine di sfondo e di un’icona



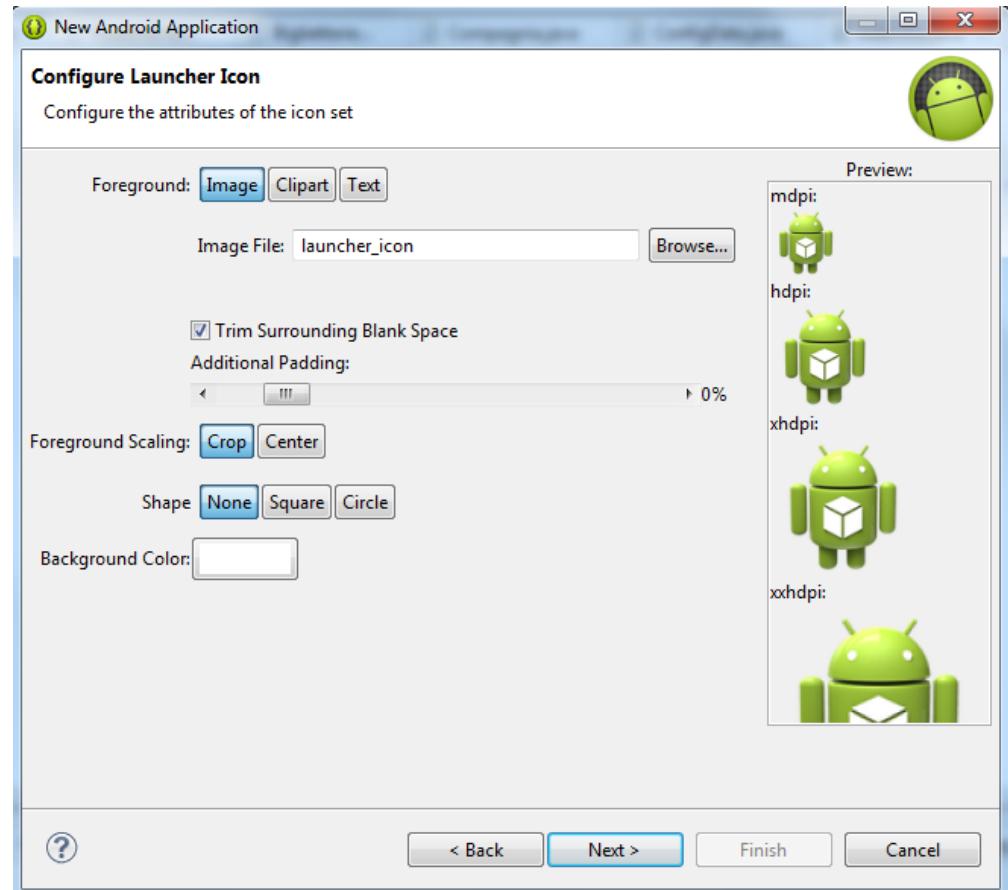
Crea una nuova Android Application

- **Oltre a nome e package bisogna impostare:**
 - Versione minima della SDK
 - Più è bassa più la app sarà compatibile per dispositivi vecchi
 - Versione target della SDK
 - Quella sulla quale testeremo la app
 - Compilatore
 - Scegliamo di solito il più recente



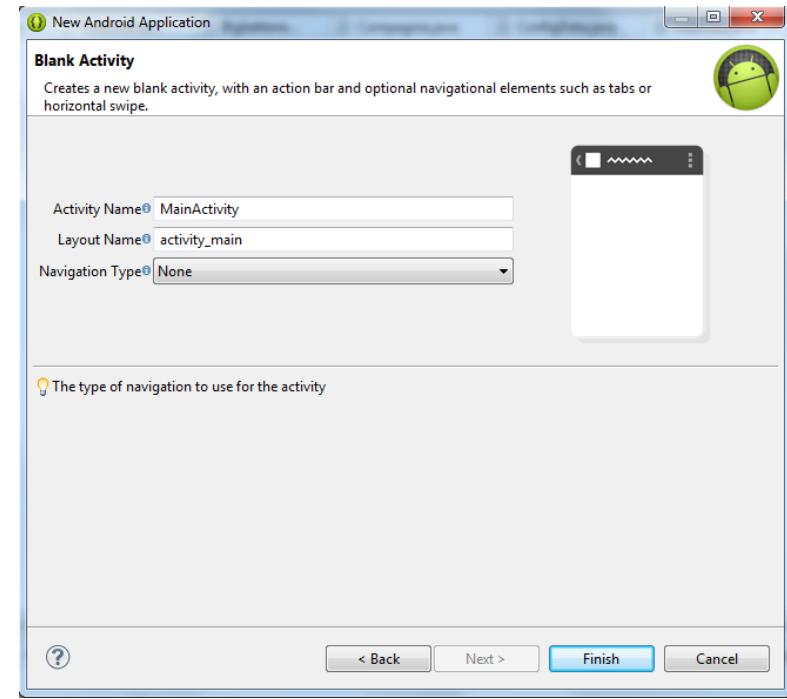
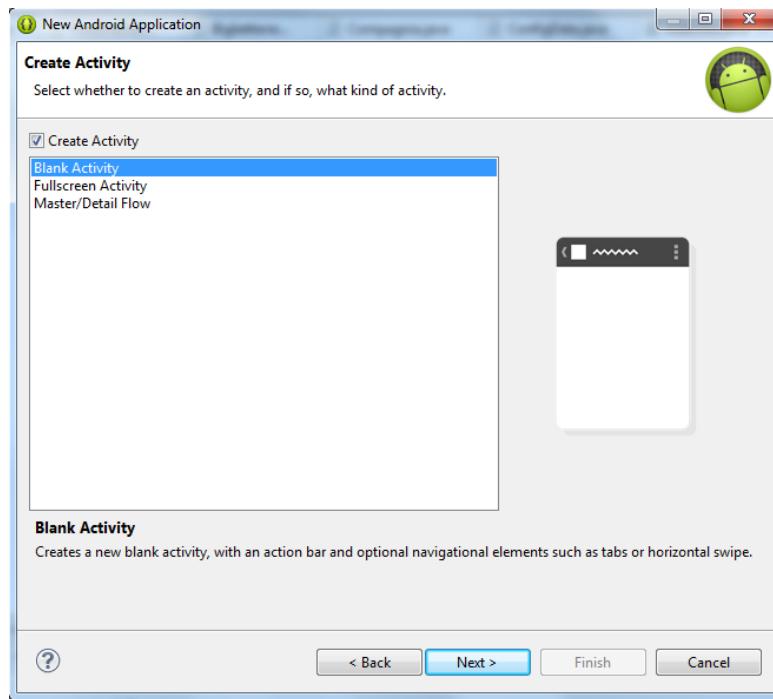
Impostazione icone

- **Dobbiamo creare una icona per la app**
 - *Dobbiamo creare con diverse risoluzioni, per adattarsi alle diverse possibili dimensioni dello schermo di un cellulare o tablet*
- **Possiamo caricare l'icona da un qualsiasi file e darle una forma**



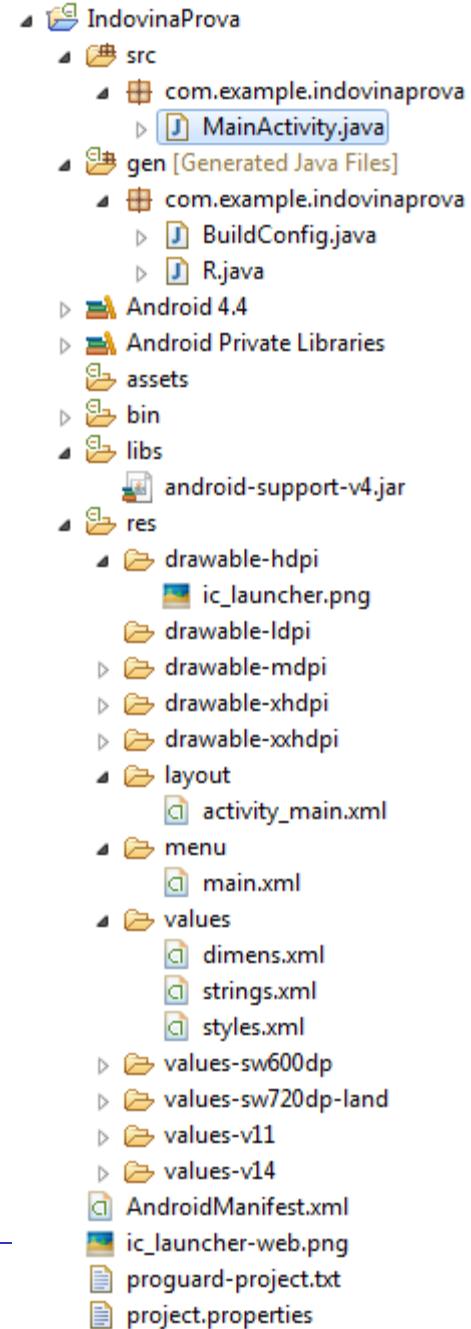
Impostazione prima Activity

- **Dobbiamo creare la prima Activity**
 - Il sistema ci suggerisce alcuni layout possibili
 - Successivamente diamo un nome all'Activity e al suo layout



Codice generato automaticamente

- **MainActivity.java**
- **File automatici BuildConfig e R (da non modificare)**
- **Librerie di sistema da importare**
- **File png rappresentativi delle icone a diverse risoluzioni, in diverse cartelle**
 - I nomi delle cartelle seguono sempre precise convenzioni e non devono essere modificati
- **Xml con la descrizione del layout**
- **Xml con la descrizione del menu**
- **Xml che riportano le stringhe di testo costanti utilizzate nel programma**
 - Anche le stringhe possono variare con la dimensione dello schermo. Ad es. su schermo piccolo si possono utilizzare abbreviazioni
- **AndroidManifest.xml**
- **Altre proprietà del progetto (utili per la compilazione)**



AndroidManifest.xml

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

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="10" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.indovinaprova.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>
    </application>

</manifest>
```

Editor del layout

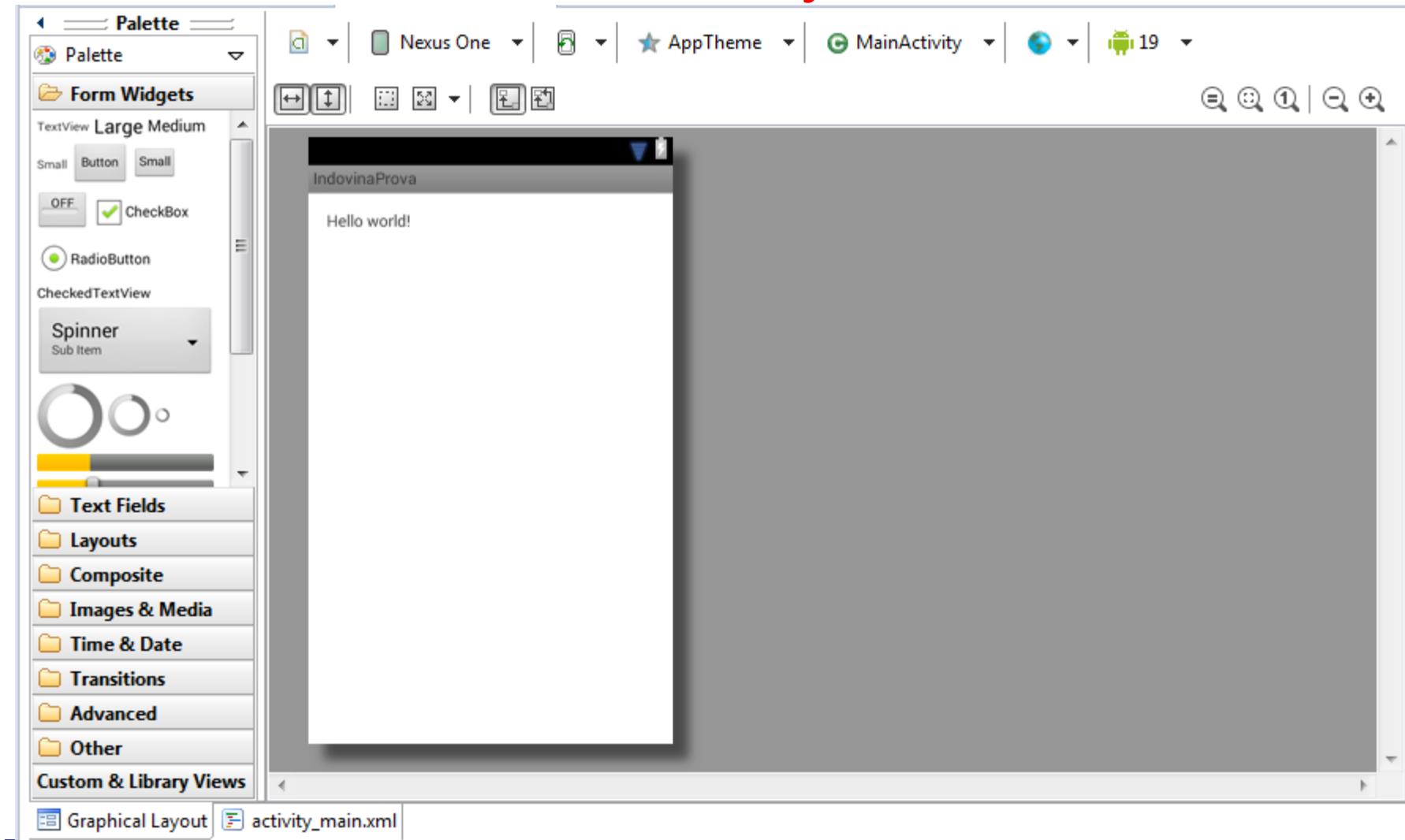


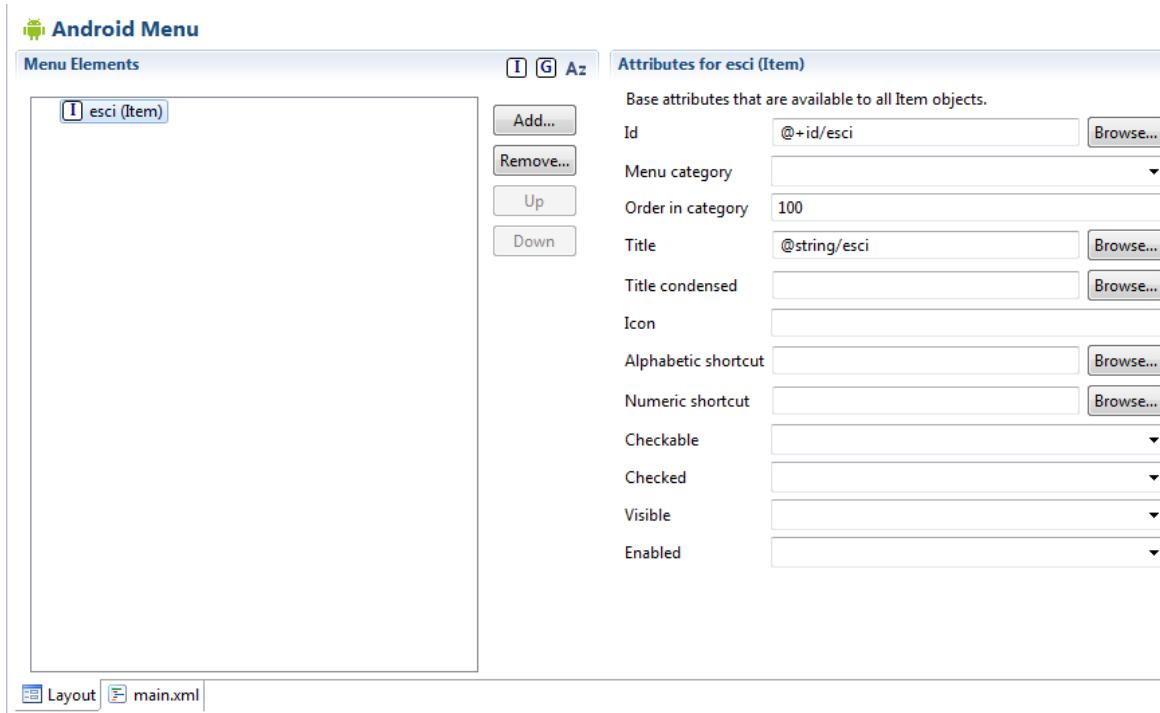
Immagine di sfondo e icona

- **Modifichiamo la casella di testo «Hello, World»**
 - Scriviamo «Indovina l'ora del delitto», scriviamola in rosso su sfondo nero e con font più grande
 - Cerchiamo di definire delle resources, anziché mettere costanti nel codice xml
 - In particolare i colori vanno in un file di colori della cartella res/values
 - Il background può essere editato tramite Other properties/all By Name/Background nel menu contestuale
- **Mettiamo un'immagine di background**
 - Dobbiamo prima copiarla in una o più delle cartelle res/drawable
- **Creiamo un pulsante «Ipotizza un orario»**



Immagine di sfondo e icona

- **Creiamo un menu (menu.xml nella cartella menu) con un'unica voce: esci**
 - La stringa «Esci» la salviamo come risorsa



MainActivity.java

- **Codice generato automaticamente**

```
package com.example.indovinaprova;
```

```
import android.os.Bundle;  
import android.app.Activity;  
import android.view.Menu;
```

```
public class MainActivity extends Activity {
```

```
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }
```

```
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        // Inflate the menu; this adds items to the action bar if it is present.  
        getMenuInflater().inflate(R.menu.main, menu);  
        return true;  
    }
```

```
}
```

Collegamento dinamico col layout creato visualmente (e trasformato dal framework in codice java e compilato)

Collegamento dinamico al menu creato visualmente (e trasformato dal framework in codice java e compilato)

Codice per il menu e il dialog corrispondente 1/3

- **Vogliamo che sulla voce di menu «esci» si apra un dialog di conferma e eventualmente venga chiusa la app**

– In onCreate

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Sei sicuro di voler uscire?")
.setCancelable(false)
.setPositiveButton("Si", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        MainActivity.this.finish(); ←
    }
})
.setNegativeButton("No", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        dialog.cancel(); ←
    }
});
quitDialog = builder.create(); ←
```

AlertDialog.Builder è in grado di «costruire» un alert dialog

Metodo collegato al pulsante «SI»

Metodo collegato al pulsante «No»

Codice per il menu e il dialog corrispondente 2/3

- **Vogliamo che sulla voce di menu «esci» si apra un dialog di conferma e eventualmente venga chiusa la app**

@Override

```
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.esci:
            showDialog(QUIT_DIALOG_ID); ←
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Metodo per collegare i click sulle voci di menu alle azioni corrispondenti

Avvio del dialog
QUIT_DIALOG_ID

Codice per il menu e il dialog corrispondente 3/3

- **Vogliamo che sulla voce di menu «esci» si apra un dialog di conferma e eventualmente venga chiusa la app**

```
@Override  
protected Dialog onCreateDialog(int id){  
    switch (id) {  
        case QUIT_DIALOG_ID:  
            return quitDialog;  
        }  
        return null;  
    };
```

Metodo che viene chiamato tramite showDialog e che abbina ai codici dei dialog le azioni corrispondenti

Avvio del dialog
QUIT_DIALOG_ID



Codice per il pulsante e il dialog corrispondente 1/3

- **Vogliamo che sul pulsante «Ipotizza» si avvii un componente che fa scegliere un orario e venga confrontato con quello da indovinare**
 - In onCreate

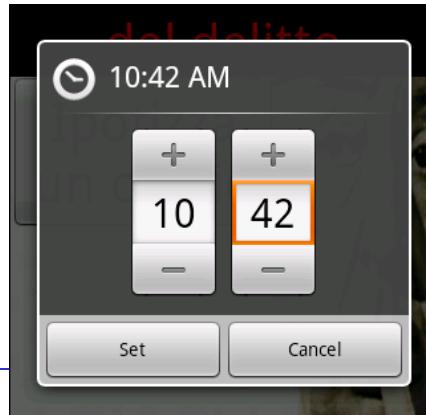
//pulsante Ipotizza orario

```
mPickTime = (Button) findViewById(R.id.ipotizza);
mPickTime.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        showDialog(TIME_DIALOG_ID);
    }
});
```

Abbiniamo al pulsante
chimato ipotizza nel layout
l'oggetto Button mPickTime

Ascoltatore del pulsante

Avvio del dialog
TIME_DIALOG_ID



Codice per il pulsante e il dialog corrispondente 2/3

- **Vogliamo che sul pulsante «Ipotizza» si avvii un componente che fa scegliere un orario e venga confrontato con quello da indovinare**

```
@Override  
protected Dialog onCreateDialog(int id) {  
    switch (id) {  
        case TIME_DIALOG_ID:  
            return new TimePickerDialog(this,mTimeSetListener, mHour, mMinute, true);  
        case QUIT_DIALOG_ID:  
            return quitDialog;  
    return null;  
};  
}
```

Aggiungiamo
TIME_DIALOG_ID

Codice per il pulsante e il dialog corrispondente 3/3

- **Vogliamo che sul pulsante «Ipotizza» si avvii un componente che fa scegliere un orario e venga confrontato con quello da indovinare**

```
private TimePickerDialog.OnTimeSetListener mTimeSetListener =  
    new TimePickerDialog.OnTimeSetListener() {  
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {  
            mHour = hourOfDay;           ←———— L'ora inserita in  
            mMinute = minute;          input  
            if (mHour<oraOK || (mHour==oraOK && mMinute<minutoOK)) {  
                txtStato.setText("Piu' tardi! (finora "+tentativi+" tentativi)");  
                tentativi++;  
            }  
            else if (mHour>oraOK || (mHour==oraOK && mMinute>minutoOK)) {  
                txtStato.setText("Prima! (finora "+tentativi+" tentativi)");  
                tentativi++;  
            }  
            else if (mHour==oraOK && mMinute==minutoOK)  
                txtStato.setText("Hai indovinato in "+tentativi+" tentativi ed e' il miglior risultato  
di oggi!");  
        }  
    }
```

Resto del codice

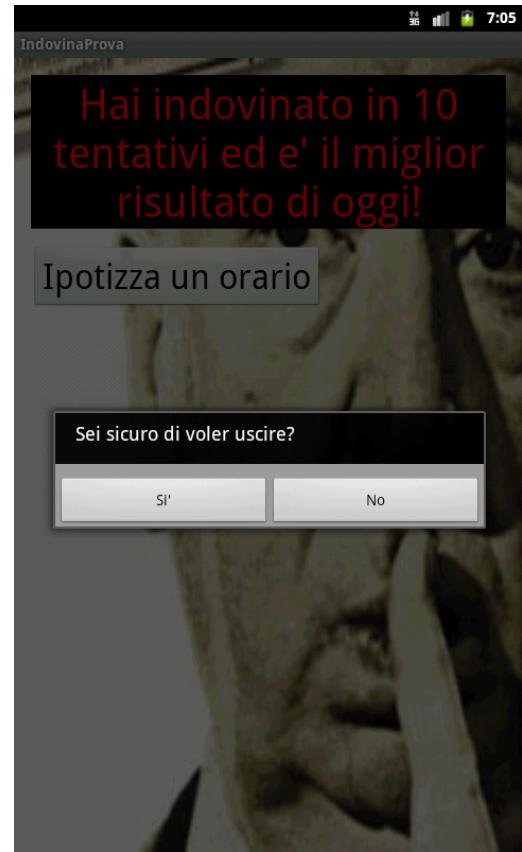
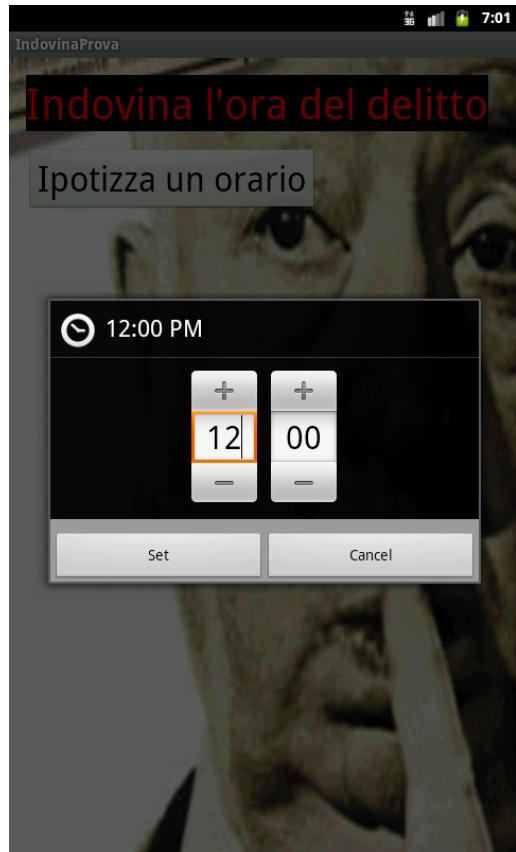
- **Costanti e variabili**

```
private static final Random RNG = new Random();
private int oraOK;
private int minutoOK;
private int tentativi=1;
static final int TIME_DIALOG_ID = 0;
static final int QUIT_DIALOG_ID = 1;
```

- **In onCreate**

```
txtStato = (TextView)findViewById(R.id.textView1);
oraOK=RNG.nextInt(24);
minutoOK=RNG.nextInt(60);
```

Esecuzione

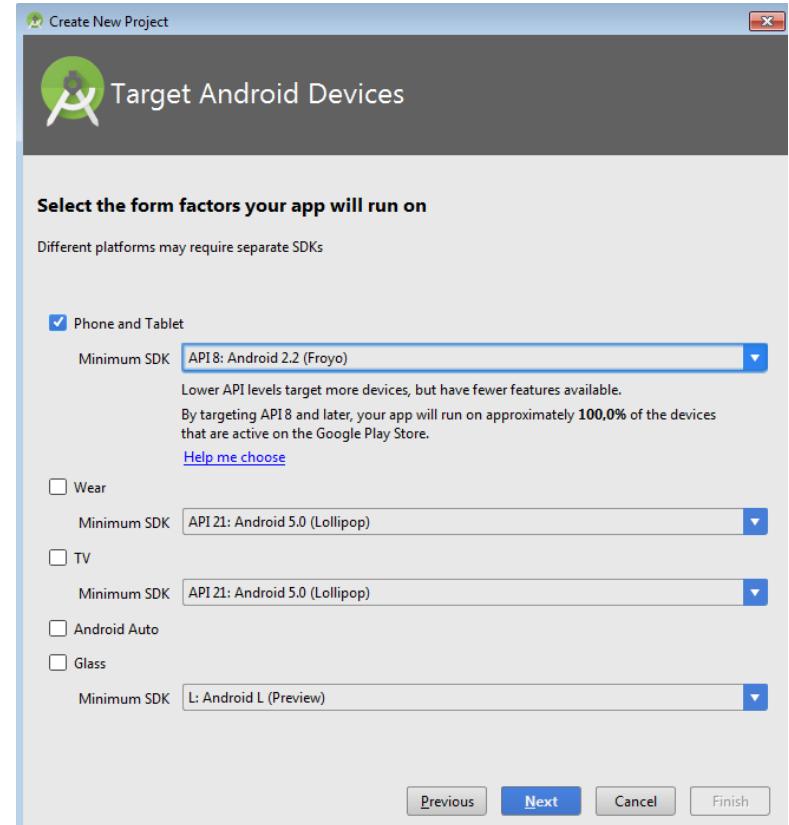
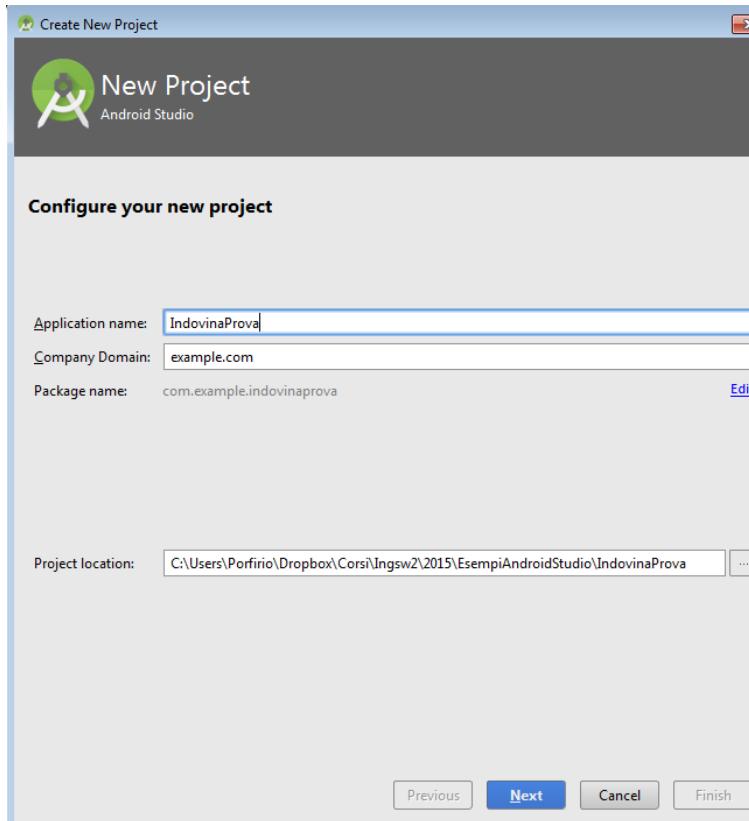


Tutorial: IndovinaOraDelitto (versione Android Studio)

- **Si tratta di un progetto molto più piccolo**
 - Piccolo gioco nel quale bisogna indovinare “l’orario del delitto” (un orario scelto a caso nelle 24 ore)
 - Ad ogni tentativo, il sistema risponde dicendo se l’orario esatto è precedente o successivo
 - Un’unica interfaccia → un’unica Activity
 - Gestione delle Shared Preferences (per memorizzare il punteggio migliore)
 - Riuso di un componente (widget) per la richiesta dell’orario
 - Visualizzazione di un’immagine di sfondo e di un’icona



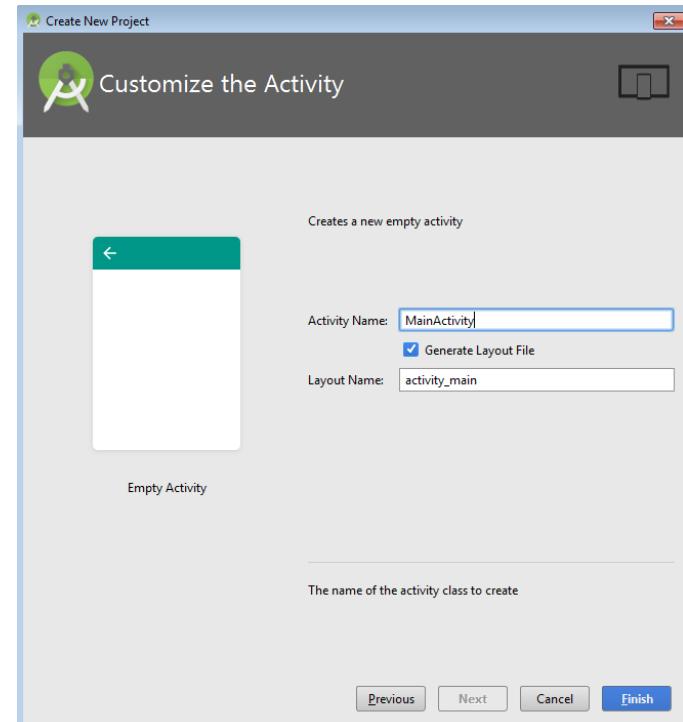
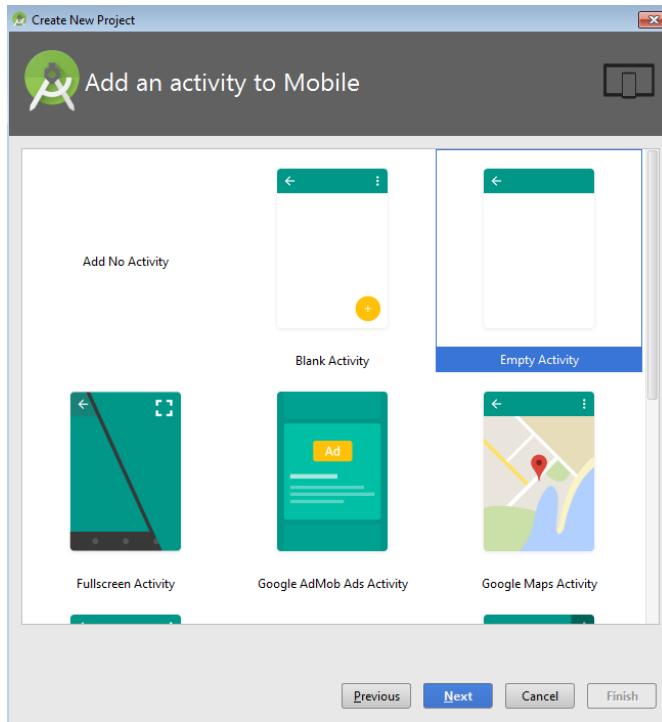
Crea una nuova Android Application



- Oltre a nome e package bisogna impostare la versione minima della SDK
 - Più è bassa più la app sarà compatibile per dispositivi vecchi

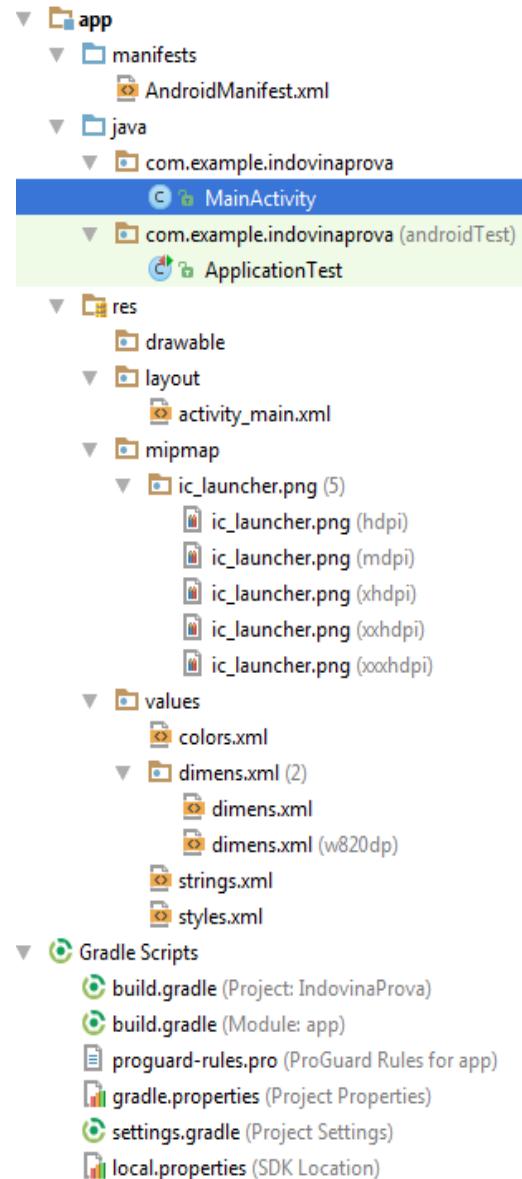
Impostazione prima Activity

- **Dobbiamo creare la prima Activity**
 - Il sistema ci suggerisce alcuni layout possibili
 - Successivamente diamo un nome all'Activity e al suo layout



Codice generato automaticamente

- **MainActivity.java**
- **File automatici BuildConfig e R (da non modificare)**
- **Librerie di sistema da importare**
- **File png rappresentativi delle icone a diverse risoluzioni, in diverse cartelle**
 - I nomi delle cartelle seguono sempre precise convenzioni e non devono essere modificati
- **Xml con la descrizione del layout**
- **Xml con la descrizione del menu**
- **Xml che riportano le stringhe di testo costanti utilizzate nel programma**
 - Anche le stringhe possono variare con la dimensione dello schermo. Ad es. su schermo piccolo si possono utilizzare abbreviazioni
- **AndroidManifest.xml**
- **Script gradle di compilazione**



AndroidManifest.xml

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="IndovinaProva"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Editor del layout

The screenshot shows the Android Studio Layout Editor interface. On the left, the **Palette** is open, displaying categories for **Layouts**, **Widgets**, and **Text Fields**. In the center, a preview of a Nexus 4 device shows a single **TextView** with the text "Hello World!". The **Component Tree** panel on the right shows the hierarchy: **Device Screen** contains a **RelativeLayout**, which contains a **TextView** with the text "Hello World!". The **Properties** panel on the far right lists various view properties like layout_width, layout_height, style, and background.

Palette

Layouts

- FrameLayout
- LinearLayout (Horizontal)
- LinearLayout (Vertical)
- TableLayout
- TableRow
- GridLayout
- RelativeLayout

Widgets

- Plain TextView
- Large Text
- Medium Text
- Small Text
- Button
- Small Button
- RadioButton
- CheckBox
- Switch
- ToggleButton
- ImageButton
- ImageView
- ProgressBar (Large)
- ProgressBar (Normal)
- ProgressBar (Small)
- ProgressBar (Horizontal)
- SeekBar
- RatingBar
- Spinner
- WebView

Text Fields

- Plain Text
- Person Name
- Password

Component Tree

- Device Screen
 - RelativeLayout
 - TextView - "Hello World!"

Properties

layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfter	
accessibilityTraversalBefore	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	<input type="checkbox"/>
contentDescription	
contextClickable	<input type="checkbox"/>
elevation	
focusable	<input type="checkbox"/>

Immagine di sfondo e icona

- **Modifichiamo la casella di testo «Hello, World»**
 - Scriviamo «Indovina l'ora del delitto», scriviamola in rosso su sfondo nero e con font più grande
 - Cerchiamo di definire delle resources, anziché mettere costanti nel codice xml
 - In particolare i colori vanno in un file di colori della cartella res/values
 - Il background può essere editato tramite Other properties/all By Name/Background nel menu contestuale
- **Mettiamo un'immagine di background**
 - Dobbiamo prima copiarla in una o più delle cartelle res/drawable
- **Creiamo un pulsante «Ipotizza un orario»**



Menu

- ***Creiamo un menu (menu.xml nella cartella menu) con un'unica voce: esci***
 - La stringa «Esci» la salviamo come risorsa
- ***In Android Studio non c'è un editor grafico per i menu***
 - Probabilmente perché stanno lentamente andando in disuso

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:enabled="true" android:orderInCategory="1" android:title="Ricomincia" android:visible="true" android:id="@+id/restartmenu"></item>
    <item android:enabled="true" android:id="@+id/esci" android:orderInCategory="2" android:title="Esci" android:visible="true"></item>
</menu>
```

MainActivity.java

```
package com.example.indovinaprova;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

Collegamento dinamico col layout creato visualmente (e trasformato dal framework in codice java e compilato)

Collegamento dinamico al menu creato visualmente (e trasformato dal framework in codice java e compilato)

Codice per il menu e il dialog corrispondente 1/3

- **Vogliamo che sulla voce di menu «esci» si apra un dialog di conferma e eventualmente venga chiusa la app**

– In onCreate

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Sei sicuro di voler uscire?")
.setCancelable(false)
.setPositiveButton("Si", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        MainActivity.this.finish(); ←
    }
})
.setNegativeButton("No", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        dialog.cancel(); ←
    }
});
quitDialog = builder.create(); ←
```

AlertDialog.Builder è in grado di «costruire» un alert dialog

Metodo collegato al pulsante «SI»

Metodo collegato al pulsante «No»

Codice per il menu e il dialog corrispondente 2/3

- **Vogliamo che sulla voce di menu «esci» si apra un dialog di conferma e eventualmente venga chiusa la app**

@Override

```
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.esci:
            showDialog(QUIT_DIALOG_ID); ←
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Metodo per collegare i click sulle voci di menu alle azioni corrispondenti

Avvio del dialog
QUIT_DIALOG_ID

Codice per il menu e il dialog corrispondente 3/3

- **Vogliamo che sulla voce di menu «esci» si apra un dialog di conferma e eventualmente venga chiusa la app**

```
@Override  
protected Dialog onCreateDialog(int id){  
    switch (id) {  
        case QUIT_DIALOG_ID:  
            return quitDialog;  
        }  
        return null;  
    };
```

Metodo che viene chiamato tramite showDialog e che abbina ai codici dei dialog le azioni corrispondenti

Avvio del dialog
QUIT_DIALOG_ID



Codice per il pulsante e il dialog corrispondente 1/3

- **Vogliamo che sul pulsante «Ipotizza» si avvii un componente che fa scegliere un orario e venga confrontato con quello da indovinare**
 - In onCreate

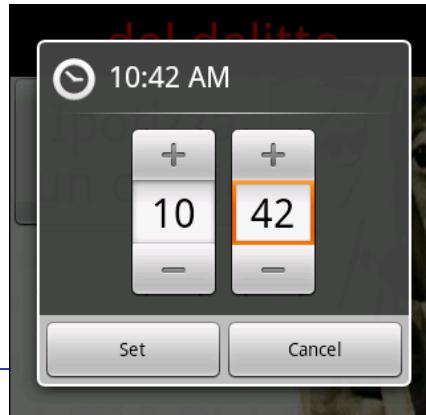
//pulsante Ipotizza orario

```
mPickTime = (Button) findViewById(R.id.ipotizza);
mPickTime.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        showDialog(TIME_DIALOG_ID);
    }
});
```

Abbiniamo al pulsante
chimato ipotizza nel layout
l'oggetto Button mPickTime

Ascoltatore del pulsante

Avvio del dialog
TIME_DIALOG_ID



Codice per il pulsante e il dialog corrispondente 2/3

- **Vogliamo che sul pulsante «Ipotizza» si avvii un componente che fa scegliere un orario e venga confrontato con quello da indovinare**

```
@Override  
protected Dialog onCreateDialog(int id) {  
    switch (id) {  
        case TIME_DIALOG_ID:  
            return new TimePickerDialog(this,mTimeSetListener, mHour, mMinute, true);  
        case QUIT_DIALOG_ID:  
            return quitDialog;  
    return null;  
};  
}
```

Aggiungiamo
TIME_DIALOG_ID

Codice per il pulsante e il dialog corrispondente 3/3

- **Vogliamo che sul pulsante «Ipotizza» si avvii un componente che fa scegliere un orario e venga confrontato con quello da indovinare**

```
private TimePickerDialog.OnTimeSetListener mTimeSetListener =  
    new TimePickerDialog.OnTimeSetListener() {  
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {  
            mHour = hourOfDay;           ←———— L'ora inserita in  
            mMinute = minute;          input  
            if (mHour<oraOK || (mHour==oraOK && mMinute<minutoOK)) {  
                txtStato.setText("Piu' tardi! (finora "+tentativi+" tentativi)");  
                tentativi++;  
            }  
            else if (mHour>oraOK || (mHour==oraOK && mMinute>minutoOK)) {  
                txtStato.setText("Prima! (finora "+tentativi+" tentativi)");  
                tentativi++;  
            }  
            else if (mHour==oraOK && mMinute==minutoOK)  
                txtStato.setText("Hai indovinato in "+tentativi+" tentativi ed e' il miglior risultato  
di oggi!");  
        }  
    }
```

Resto del codice

- **Costanti e variabili**

```
private static final Random RNG = new Random();
private int oraOK;
private int minutoOK;
private int tentativi=1;
static final int TIME_DIALOG_ID = 0;
static final int QUIT_DIALOG_ID = 1;
```

- **In onCreate**

```
txtStato = (TextView)findViewById(R.id.textView1);
oraOK=RNG.nextInt(24);
minutoOK=RNG.nextInt(60);
```

Progetto d'esempio: Caccia Al Tesoro

- **Sfruttando le capacità di ricezione GPS contenute in un dispositivo mobile Android, si vuole realizzare una sorta di Caccia al Tesoro**
- **Una Caccia al Tesoro si compone di una sequenza di tappe da completare**
 - L'obiettivo di ogni tappa è il raggiungimento di un determinato luogo, che dovrà essere notificato dalle coordinate GPS lette dal dispositivo stesso
 - In pratica il giocatore dovrà recarsi fisicamente nel luogo richiesto

Modello delle informazioni

- **Il modello delle informazioni è molto semplice:**
 - Una Caccia è composta di Tappe (3 in questa versione) e può essere giocata da un unico giocatore
 - Bisogna memorizzare il tempo trascorso dall'inizio della tappa
 - Ogni Tappa ha un nome e una coppia di coordinate dell'obiettivo
 - C'è una ovvia relazione di aggregazione tra Caccia e Tappa

Modello dell'interfaccia

- L'interfaccia utente consiste di un'unica classe che estende Activity, CacciaAlTesoro, con 4 campi di testo, definiti staticamente in xml

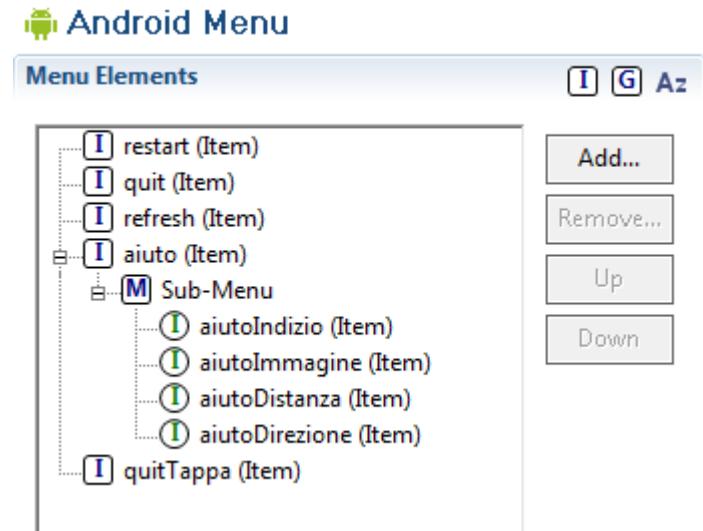


Extra features

- **Ogni tappa ha un tempo massimo e un punteggio dipendente dal tempo impiegato**
- **E' possibile chiedere degli aiuti, che fanno diminuire il punteggio. In particolare:**
 - L'aiuto distanza visualizza la distanza in linea d'aria dal punto obiettivo
 - L'aiuto direzione indica la direzione (rispetto al nord) nella quale andare per raggiungere, in linea d'aria, l'obiettivo
- **Il gioco non si interrompe se un'altra applicazione (ad esempio il telefono) si attiva**
 - Il gioco termina solo su esplicita richiesta dell'utente

Menu

- **Un menu definito staticamente per l'unica Activity con 4 opzioni semplici e un sottomenu per gli aiuti con 4 opzioni**
- **9 handler da implementare (nella classe CacciaAITesoro che estende Activity e fa da Delegate)**



Suddivisione delle responsabilità

- **La classe CacciaAITesoro estende Activity, gestisce l'interfaccia utente**
 - ha la delega per la gestione degli eventi da menu
 - Implementa e gestisce i Dialog
 - Gestisce gli eventi legati al GPS
 - Gestisce il ciclo di vita dell'activity

```
public class CacciaAITesoro extends Activity implements  
    LocationListener
```

- **La classe Caccia implementa l'algoritmo del gioco**
 - Calcolo del punteggio, gestione del tempo, etc.
- **La classe Tappa modella le strutture dati di una singola tappa**
 - Per semplicità, in questa versione le tappe sono istanziate nel codice di Caccia

Gestione del GPS

- **La classe Activity implementa**
 - myManager = (LocationManager) getSystemService(LOCATION_SERVICE);
 - myManager.getLastKnownLocation(LocationManager.GPS_PROVIDER)
 - Forza una valutazione sincrona della posizione
 - private void startListening()
 myManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
 - private void stopListening() {
 myManager.removeUpdates(this);
 - public void onLocationChanged(Location location) { ... }
 - Parte in maniera asincrona col programma e sincrona con il rilevamento GPS
- **La classe android.location.Location fornisce tra l'altro i metodi getLongitude() e getLatitude()**
- **Nel Manifest.xml bisogna settare**
`<uses-permission
 android:name="android.permission.ACCESS_FINE_LOCATION" />`

Gestione di chiusura e riapertura

- **Nella classe CacciaAITesoro**
 - onCreate istanzia tutti i dialog, inizializza gli attributi e avvia la Caccia
 - onPause salva lo stato della Caccia
 - onResume ripristina lo stato della Caccia
 - onDestroy distrugge i dati della Caccia e chiude l'applicazione
- **La Caccia prosegue anche se l'applicazione non è visibile sull'interfaccia sfruttando il fatto che il tempo (di sistema) continua a scorrere**
 - L'eventuale tempo scaduto su di una caccia viene in realtà notificato solo quando l'applicazione è riaperta

SharedPreferences

- Come memorizzare il punteggio migliore?
- Android ha diversi metodi per gestire la persistenza. Il più semplice si basa sulle SharedPreferences, simili alle variabili di sessione delle applicazioni Web
- Esempio di lettura
 - private SharedPreferences settings;
 - settings = getSharedPreferences("RecordFile", 0);
 - min = settings.getInt("record", 1000);
- Esempio di scrittura (salva record)
 - SharedPreferences.Editor editor = settings.edit();
 - editor.putInt("record", min);
 - editor.commit();