- Matteo Bonifazi -

Sviluppare applicazioni per Android

in 7 giorni

2ª EDIZIONE

BESTSELLER

Implementazione del Material Design nelle applicazioni >>

Pubblicazione e monetizzazione con il Play Store >>

Servizi di localizzazione con Google Maps >>

Rendere le app wearable con Android Wear >>

*pro DigitalLifeStyle

*pro
Digital*LifeStyle

Sviluppare applicazioni per Android

in 7 giorni

Matteo Bonifazi

EDIZIONI
LSWR

Sviluppare applicazioni per Android | in 7 giorni, 2ª ed.

Autore: Matteo Bonifazi

Collana: *pro DigitalLifeStyle

Editor in Chief: Marco Aleotti
Progetto grafico: Roberta Venturieri
Immagine di copertina: Fabio Prati
Realizzazione editoriale e impaginazione: Studio Dedita di Davide Gianetti
Redazione: Stefano Andreini

ISBN: 978-88-6895-268-6

EDIZIONI LSWR

Via G. Spadolini, 7
20141 Milano (MI)
Tel. 02 881841
www.edizionilswr.it

Printed in Italy

Finito di stampare nel mese di febbraio 2016 presso "LegoDigit" Srl., Lavis (TN)

(*) Edizioni Lswr è un marchio di La Tribuna Srl. La Tribuna Srl fa parte di LSWR GROUP.

Sommario

Prefazione

Android, nato come un sistema operativo per dispositivi mobili e con una diffusione ormai globale, quasi non necessita di un'introduzione. L'obiettivo che Google si è prefissato di superare, ossia "i 2 miliardi di utenti", è ormai vicinissimo.
Ci sono diverse ragioni dietro questa diffusione capillare in continuo aumento. La più importante è la varietà di dispositivi che offrono Android come sistema operativo: smartphone, tablet, orologi, automobili. Si parla, ormai, di *ubiquitous computing* (computazione ubiqua), intesa come la possibilità di accedere alle informazioni e ai servizi digitali con dispostivi che fanno parte della nostra quotidianità. Inoltre, la convenienza di alcuni di questi dispositivi rende possibile la distribuzione in mercati ancora in via di sviluppo.
Con la certezza di un pubblico così vasto, è naturale che l'interesse da parte degli sviluppatori abbia iniziato a gravitare sempre più di frequente verso Android. Se a questo si uniscono la facilità di apprendimento garantita da Java, il linguaggio su cui Android si basa, e i vantaggi dati dal fatto che il codice sorgente è open source, si hanno tutti gli ingredienti per una comunità di sviluppatori vastissima e affiatata, che mette a disposizione migliaia di librerie per rendere ancora più produttivo lo sviluppo delle applicazioni. Non è da meno l'impegno che Google sta dimostrando verso questa comunità, andando a offrire tutti gli strumenti necessari per uno sviluppo efficiente.
Ed è qui che Matteo Bonifazi, sviluppatore Android da molti anni e Google Developer Expert, entra in gioco. In soli sette giorni, seguendo la sua esperta guida e i suoi preziosi consigli, sarete in grado di dare alla luce la vostra prima applicazione Android e di pubblicarla sul Play Store, dove più di un miliardo di utenti aspettano con trepidazione di scoprire che cosa avete in serbo per loro.
Il successo vi aspetta. Buono sviluppo!

Sebastiano Gottardo
Android Engineer @ Musixmatch
Google Developer Expert – Android

Introduzione

Il 23 settembre 2008 Google svela al mondo il suo sistema operativo open source mobile: Android. L'obiettivo dichiarato – particolarmente audace, dal momento che Nokia era ancora il leader indiscusso del mercato e Apple era in forte ascesa con il suo iPhone – è quello di far diventare Android il sistema operativo mobile più utilizzato al mondo.
A oggi, l'obiettivo è più che mai raggiunto: Android è il sistema operativo mobile adottato da oltre l'80% dei dispositivi mobili a livello mondiale e alimenta centinaia di milioni di dispositivi in più di 200 Paesi. Ogni giorno, più di un milione di utenti accende per la prima volta un dispositivo Android per l'utilizzo di applicazioni, giochi e contenuti digitali. Tutti i più grandi produttori di tecnologia mondiale (Samsung, Asus, LG, Sony, solo per citarne alcuni) hanno in catalogo prodotti con Android a bordo. Grazie ai suoi partner tecnici e alla community open source che si è sviluppata in questi anni, Android sta costantemente spingendo i limiti hardware e software per portare nuove possibilità e funzionalità agli utenti e agli sviluppatori. La grande duttilità permette a questo sistema operativo di essere integrato all'interno degli oggetti più disparati: dalle televisioni alle automobili, dall'Internet of Things ai dispositivi indossabili (ne sono esempio i progetti Android Wear, Android TV e Android Auto).
La duttilità e l'innovazione garantiscono agli sviluppatori lo stimolo per sperimentare, nelle proprie applicazioni, idee sempre nuove. Attraverso il Google Play Store, gli sviluppatori possono distribuire le proprie applicazioni a pagamento o gratuitamente utilizzando un marketplace aperto. Con un miliardo e mezzo di applicazioni scaricate mensilmente, il Google Play Store permette di arrivare a un pubblico di scala mondiale. Android rappresenta, quindi, un ecosistema che deve sicuramente essere preso in considerazione da tutte quelle persone che vogliono sviluppare le proprie idee in ambito mobile e non solo.

A chi si rivolge questo libro

Questo libro è rivolto a coloro che vogliono portare le proprie idee e applicazioni nell'ecosistema Android. A questo gruppo appartengono sia sviluppatori principianti, desiderosi di acquisire skill di programmazione mobile, sia sviluppatori esperti in altre piattaforme mobile (iOS, Windows Phone ecc.), intenzionati a conoscere funzionalità e caratteristiche di un sistema operativo mobile concorrente. È importante che il lettore abbia familiarità con lo sviluppo del software e con i concetti base della programmazione a oggetti. È richiesta una comprensione di base della sintassi Java e XML; una preparazione dettagliata rappresenta sicuramente un vantaggio, tuttavia non è strettamente necessaria.

La struttura

Il libro è diviso in sette capitoli, ognuno dei quali corrisponde idealmente a un giorno della settimana.

Nelle giornate di lunedì e martedì è introdotto il sistema operativo Android. I capitoli spiegano come impostare l'ambiente di sviluppo, come comporre un'applicazione Android attraverso i suoi pezzi fondamentali e gli elementi per la costruzione dell'interfaccia grafica, utilizzando componenti di user interface legate al Material Design. Nella giornata di mercoledì si vede come i diversi elementi di un'applicazione Android possono comunicare tra loro attraverso l'utilizzo di Intent, Fragment e messaggi broadcast.

Giovedì si affrontano i diversi temi legati alla persistenza dei dati all'interno di un'applicazione: dal meccanismo di salvataggio attraverso le preferenze alla gestione dei file, dai database ai Content Provider e ai Cursor.

Nella giornata di venerdì si esamina il tema del background, utilizzando AsyncTask, Services, Loader e background thread. L'integrazione con Google Maps e i servizi di geolocalizzazione sono descritti nella giornata di sabato. Per finire, domenica si illustra come pubblicare la propria applicazione su Google Play.

Dopo sette giorni di teoria e sperimentazione, il lettore sarà in grado di sviluppare in totale autonomia la propria applicazione e distribuirla su Google Play.

A compendio del lavoro svolto durante una settimana, sono state aggiunte due appendici. La prima riguarda lo sviluppo di applicazioni su Android Wear, per portare le proprie applicazioni su dispositivi indossabili. La seconda, invece, introduce il nuovo sistema di build Android Gradle, utile per produrre le proprie applicazioni in maniera automatica.

Ringraziamenti

A Eleonora per la pazienza e l'aiuto. Il tuo supporto rende possibile qualsiasi cosa io faccia.
A Stefano Sanna, perché questo libro è il sunto di cinque anni di duro lavoro spalla a spalla.
All'Android Lab e al gruppo iOS di Open Reply. Le "opportunità" e la passione per ciò che facciamo ogni giorno hanno reso possibile la stesura di questo libro.
Alla mia famiglia per avermi permesso di arrivare dove sono ora.

Sito web:

http://androidinsettegiorni.wordpress.com

Repository per il codice sorgente degli esempi:

https://github.com/dekrandroid/android-in-sette-giorni

Email:

androidinsettegiorni@gmail.com

Twitter:

@androidinsettegiorni

1

Lunedì - Introduzione all'ecosistema Android

Questo capitolo introduce lo sviluppatore nell'ecosistema Android. Dopo una breve presentazione del framework, saranno analizzati i diversi strumenti di sviluppo e la struttura di un progetto Android utili per creare una prima applicazione di prova.

Android Software Stack

Le applicazioni Android sono generalmente scritte utilizzando Java come linguaggio di programmazione. Esse sono eseguite attraverso l'uso di una virtual machine custom, in luogo della tradizionale Java Virtual Machine. Ogni applicazione è eseguita su un processo separato all'interno della propria istanza della virtual machine. Le responsabilità di gestione della memoria e dei processi sono lasciate al sistema operativo Android, il quale, se necessario, uccide i processi e libera risorse.

L'Android Software Stack è composto da un Linux Kernel e da una collezione di librerie C/C++ esposte attraverso un framework applicativo. Quest'ultimo fornisce servizi per la gestione delle applicazioni. Nella Figura 1.1 sono illustrate le diverse parti che compongono l'Android Software Stack.

- **Livello applicativo** – tutte le applicazioni, sia native sia di terze parti, sono costruite a questo livello e vengono eseguite utilizzando le classi e i servizi disponibili dall'Application Framework;
- **Application Framework** – fornisce classi per la creazione di applicazioni Android. Esso astrae l'accesso all'hardware e alle risorse del dispositivo;

- **Android RunTime** – rappresenta il motore del sistema e, insieme alle librerie, forma la base per l'Application Framework. Questo runtime è quello che differenzia un dispositivo Android da uno basato su un'implementazione mobile di un sistema Linux;
- **Librerie** – Android include diverse librerie C/C++ per l'integrazione di web browser e internet security, per il mediaplayer, per la gestione di database ecc.;
- **Linux Kernel** – i servizi core (driver hardware, network, security ecc.) sono gestiti da un Linux Kernel 2.6, che fornisce un livello di astrazione tra l'hardware e gli elementi di livello più alto dello stack.

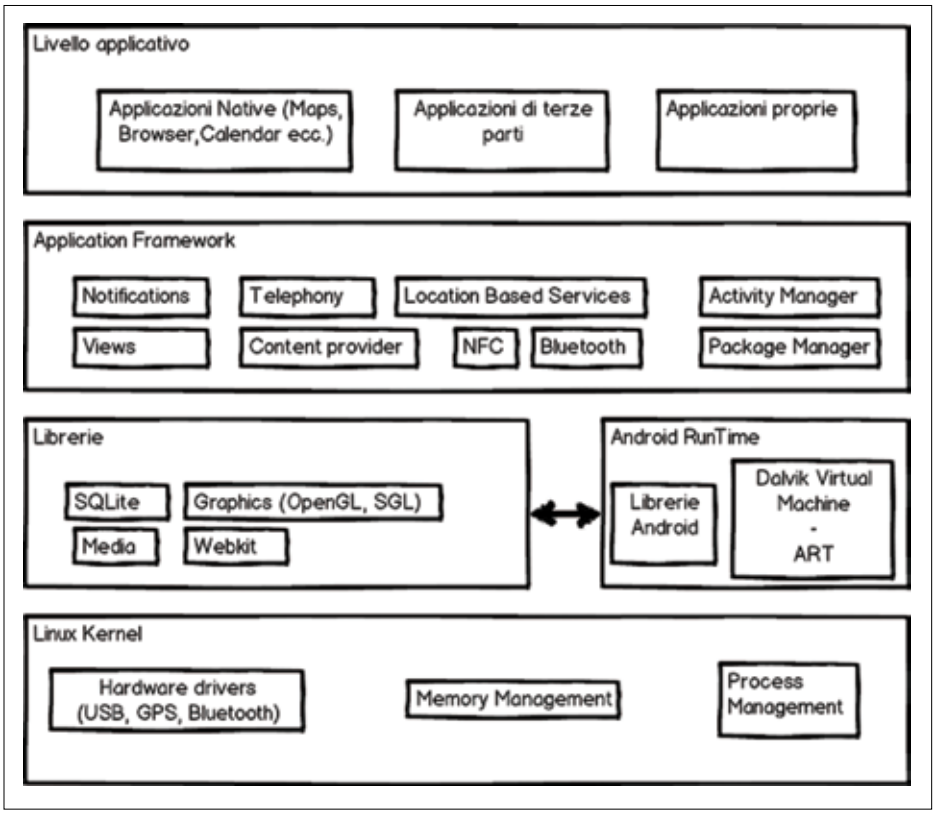

Figura 1.1 – Android Software Stack.

Android RunTime

Android RunTime (ART) è la nuova macchina virtuale presente nel sistema operativo Android per l'esecuzione delle applicazioni e di alcuni servizi di sistema. Questo runtime relativamente nuovo è stato introdotto dalla versione di Android 4.4, sostituendo

il suo predecessore Dalvik. Come quest'ultimo, ART garantisce l'esecuzione di istanze multiple in modo efficiente su un singolo dispositivo. L'accesso ai servizi di sistema e all'hardware del dispositivo è gestito utilizzando ART come livello intermedio. Utilizzando la VM (acronimo di *virtual machine*) come host per l'esecuzione di applicazioni, gli sviluppatori hanno un livello di astrazione che copre le differenti implementazioni dell'hardware. ART esegue file in Dex bytecode e permette la retrocompatibilità con tutte quelle applicazioni compilate con versioni precedenti ad Android 4.4.
Una delle caratteristiche chiave di ART è la compilazione Ahead-Of-Time (AOT), che permette in fase d'installazione di generare un'app compilata eseguibile sul dispositivo che la sta installando. Questo approccio permette un'esecuzione delle applicazioni molto più veloce, una volta che queste siano state installate.

Installazione dell'ambiente di sviluppo

Per iniziare a produrre applicazioni Android, è necessario installare e impostare l'ambiente di sviluppo sulla propria macchina di lavoro. L'ambiente di sviluppo si compone dei seguenti elementi: Java Development Kit (JDK 7 o JDK 8), Android SDK e Android Studio. Android SDK è compatibile con Windows (Windows Vista, Windows 7 e Windows 8), Mac OS X e Linux. I passi da eseguire sono gli stessi per tutte le piattaforme supportate.

Installazione di Java

La piattaforma Android per lo sviluppo di applicazioni è costruita sul framework Java Standard. Tutte le applicazioni sono create sulla base della piattaforma Java, quindi è obbligatorio installare il pacchetto Java Development Kit (JDK) per iniziare a lavorare con Android. È importante assicurarsi di avere installato la versione 7 o successive di JDK, oltre al Java Runtime Environment (JRE), che di solito è già preinstallato nel sistema. Per verificare se e quale versione di Java è installata sulla nostra macchina di sviluppo, bisogna aprire il terminale e digitare il comando `java -version`, come nell'esempio seguente:

```
MacBook-Pro-di-Matteo:~ matteobonifazi$ java -version
java version "1.7.0_65"
Java(TM) SE Runtime Environment (build 1.7.0_65-b17)
Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04, mixed mode)
```

Questo comando è stato eseguito su un Mac; ma può essere eseguito in modo similare sia su Windows sia su Linux. Il codice in grassetto evidenzia la versione corrente installata sul dispositivo (nel nostro caso la 1.7). Se il comando `java -version` generasse un errore, oppure se evidenziasse che la versione Java installata sul dispositivo

è inferiore a Java 7, allora sarebbe necessario il JDK installation package relativo al proprio sistema operativo direttamente dal sito di Java (http://www.oracle.com/technetwork/java/javase/downloads/index.html).
Il pacchetto JDK comprende il compilatore, il debugger e altri diversi tool utili per lo sviluppo software. Il pacchetto JRE, invece, è ciò che serve a questi tool per essere eseguiti.

Installazione di Android Studio

L'IDE ufficiale per lo sviluppo di applicazioni Android è Android Studio, scaricabile dal sito https://developer.android.com/sdk/index.html. All'interno di questo pacchetto troviamo tutto il necessario per cominciare a sviluppare la nostra applicazione:

- Android Studio IDE;
- Android SDK Tools;
- Android Platform Tools;
- l'ultima versione della piattaforma Android (a oggi, Android 6.0 Marshmallow – API 23);
- un'immagine dell'emulatore compatibile con l'ultima versione della piattaforma.

Una volta scaricato Android Studio, si prosegue con l'installazione e la configurazione dello stesso. Al suo primo avvio, seguendo un'installazione standard, l'IDE si occupa di scaricare l'Android SDK e i relativi tool di supporto (immagini dei diversi emulatori, librerie di supporto ecc.). Una volta completata l'installazione, è possibile creare il primo progetto Android.

> **NOTA**
>
> Poiché Android Studio è in continuo aggiornamento da parte di Google, alcuni passi nel processo di creazione di un progetto potrebbero essere leggermente differenti. Nonostante ciò, le informazioni necessarie e la struttura del progetto saranno comunque coerenti.

In questa fase, bisogna definire la cartella di sistema in cui deve essere salvato il codice sorgente, il nome e il package name della propria applicazione. Il package name è un parametro molto importante che bisogna scegliere con cura e deve essere necessariamente unico. Sarà esso, infatti, a differenziare un'applicazione dalle altre. Definiti questi primi parametri, è necessario selezionare i dispositivi sui quali sarà compatibile la nostra applicazione (smartphone, tablet, televisioni, automobili ecc.).
Per questo primo esempio, scegliamo solo la categoria "Smartphone e tablet", selezionando il minimo SDK compatibile. La selezione del valore minimo SDK supportato

dalla nostra applicazione deve essere valutata con attenzione. Un valore basso, corrispondente a una versione di Android obsoleta, garantisce infatti un'alta retrocompatibilità ma, al tempo stesso, preclude l'utilizzo di tutte le nuove funzionalità disponibili con versioni più recenti del framework. È possibile eseguire una scelta oculata confrontando la distribuzione delle diverse versioni di Android al sito https://developer.android.com/about/dashboards/index.html.

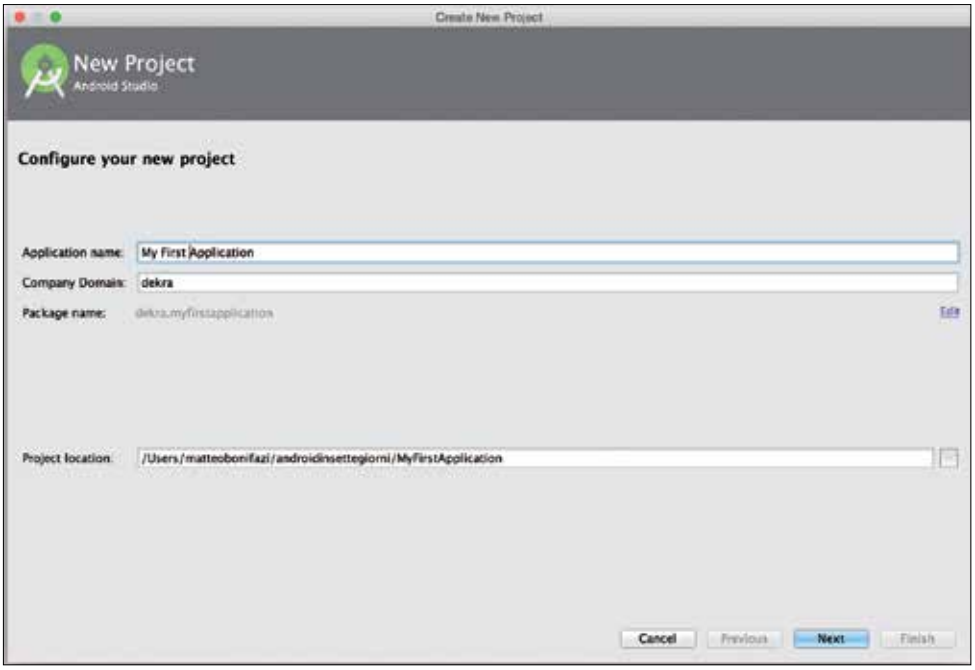

Figura 1.2 – Android Studio: creazione del progetto.

Nello step successivo, bisogna selezionare una nuova Activity o il tipo di user interface desiderata. Selezionando e specificando il nome della Blank Activity, saranno create in automatico le diverse risorse necessarie. Da questo punto in poi, l'applicazione è pronta per essere installata su un dispositivo Android.

Configurazione di un dispositivo per lo sviluppo

Il modo più veloce e semplice per testare applicazioni Android è attraverso l'utilizzo di un dispositivo Android. È possibile usare quasi tutti i device Android per lo sviluppo. Per configurare un dispositivo per lo sviluppo bisogna abilitarlo tramite le impostazioni.

Selezionare **Impostazioni** -> **Opzioni sviluppatore** -> **Debug USB**.

Figura 1.3 – Opzioni sviluppatore di un dispositivo Android.

Se la voce **Opzioni sviluppatore** non è disponibile, bisogna attivarla andando su **Impostazioni** -> **Info sul telefono** e cliccando ripetutamente (circa dieci volte) sulla voce **Numero Build**. Di seguito, verrà mostrato a video un messaggio che vi informerà che siete diventati degli sviluppatori. Tornando indietro, potrete vedere la voce **Opzioni sviluppatore** attivata. Sempre dal menu **Info sul telefono**, quando si clicca ripetutamente sulla voce **Versione di Android**, è possibile attivare una seconda opzione segreta, che mostra un wallpaper interattivo legato alla versione sul dispositivo.
Per connettere il proprio device per il testing su una macchina Windows, è necessario scaricare i relativi USB driver, che possono essere facilmente recuperati sul sito dell'azienda produttrice. Questo passo non è necessario su Linux e Mac OS X.
Per verificare che l'installazione e il settaggio siano corretti, si connette il dispositivo al proprio computer e si verifica se appare la notifica, come in Figura 1.4, che indica che il device è installato e pronto all'utilizzo.

Figura 1.4 – Dispositivo connesso correttamente.

Sviluppare applicazioni per smartphone e tablet Android significa testare il proprio codice sorgente su differenti dispositivi. Questa attività può essere davvero impegnativa, considerando che i dispositivi Android supportati nel Google Play Store sono oltre 4000. CPU, dimensione e densità dello schermo, versione dell'OS e memoria sono le principali caratteristiche che possono influenzare il corretto funzionamento della propria applicazione.
L'utilizzo di un device fisico è particolarmente consigliato se si vogliono testare funzionalità quali servizi di localizzazione, uso avanzato di sensori, rendering di giochi. È impossibile, tuttavia, avere ogni tipo di device fisico. L'utilizzo dell'emulatore diventa pertanto necessario per tutte le configurazioni che non si possiedono materialmente. Com'è ovvio, per quanto l'emulatore cerchi di riprodurre fedelmente un dispositivo fisico, ci sono delle limitazioni di cui tenere conto, quali:

- impossibilità di eseguire telefonate reali e spedire messaggi di testo;
- impossibilità di integrare accessori al dispositivo (USB, cuffie ecc.);
- nessun supporto per applicazioni particolarmente onerose in termini di prestazioni;
- nessun accesso ai servizi di Google Play Services (Gmail, Google Play Store ecc.).

Android Virtual Devices

Android Virtual Devices è un'applicazione, presente all'interno dello SDK di Android, che permette allo sviluppatore di creare diverse immagini di emulazione che rappresentano i differenti dispositivi, specificando le opzioni hardware e software. È anche presente come plugin all'interno dell'IDE di sviluppo Android Studio.
Per ogni device virtuale bisogna specificare il nome, la versione Android di riferimento, la risoluzione dello schermo e la capacità della memoria SD card. Poiché l'avvio dell'emulatore non è immediato, è possibile abilitare la funzione "snapshots" che permette di salvare lo stato dell'emulatore una volta chiuso; avviare l'emulatore da uno "snapshots" è significativamente più veloce. All'interno dell'SDK Android non è inclusa alcuna immagine; lo sviluppatore ne dovrà creare almeno una prima di lanciare la propria applicazione sull'emulatore.

Figura 1.5 – Creazione di un nuovo Android Virtual Device.

Emulatore Android

L'emulatore, come anticipato, è disponibile per il test e il debug delle proprie applicazioni. Esso rappresenta un'implementazione dell'Android VM e possiede tutte le peculiarità hardware e software tipiche di un dispositivo mobile. Fornisce, infatti, diverse funzionalità per il controllo e la navigazione; tramite la tastiera o il mouse è possibile generare eventi verso l'applicazione. Include, inoltre, diverse funzionalità aggiuntive di debug utili per lo stress test delle applicazioni: è possibile, infatti, simulare eventi di interrupts, come l'arrivo di un sms o di una telefonata, oppure la latenza o la perdita di connettività. Tutte queste caratteristiche lo rendono una valida piattaforma alternativa per sviluppo, al pari di un dispositivo Android reale.
Tutte le differenti opzioni possibili possono essere consultate alla pagina http://developer.android.com/tools/help/emulator.html; per ulteriori informazioni sull'emulatore è possibile consultare il sito http://developer.android.com/tools/devices/emulator.html.

Android Debug Monitor Service (ADMS)

Android Debug Monitor Service è un potente tool di debug che permette allo sviluppatore di interrogare i diversi processi attivi sui dispositivi, osservare e mettere in pausa i thread e navigare all'interno del file system di qualsiasi device Android.
All'interno del tool ADMS sono integrate anche le funzionalità per il recupero dei log di sistema (tramite LogCat) e per eseguire screenshot del dispositivo.

Per tutti i dispositivi che supportano Android KitKat 4.4, è possibile inoltre registrare in un video ad alta qualità tutto ciò che è visualizzato a schermo sul dispositivo connesso. Questo tool ADMS è completamente integrato in Android Studio, dove è possibile attivare la visualizzazione ADMS. Si può avviare l'ADMS anche dalla linea di comando tramite il comando `adms`.

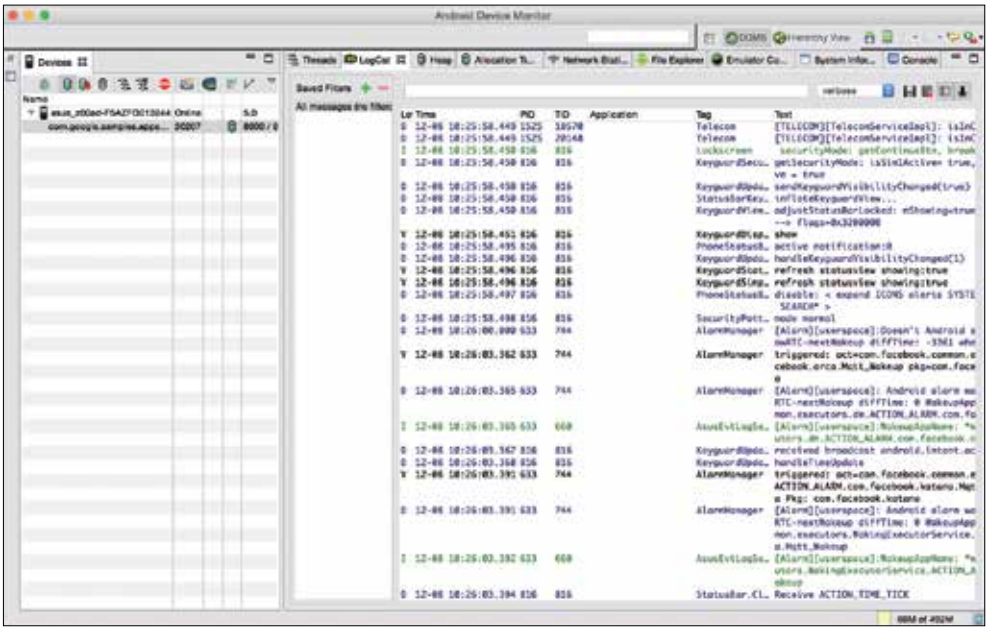

Figura 1.6 – Finestra dell'Android Debug Monitor Service.

Altri IDE di sviluppo

Si possono sviluppare applicazioni Android anche partendo da altri IDE di sviluppo. Tra questi, il più importante è sicuramente Eclipse con il suo plugin ADT (https://dl-ssl.google.com/android/eclipse/). Fino a poco tempo fa, questo era l'IDE di sviluppo ufficiale per applicazioni Android. Oggi è stato sostituito da Android Studio, come vi abbiamo mostrato in precedenza. Proprio per questo Google ha deciso di sospendere il supporto al plugin ADT, non rilasciando più aggiornamenti. Nonostante Eclipse sia ancora largamente utilizzato tra gli sviluppatori, il consiglio è quello di migrare quanto prima ad Android Studio, in modo da garantire che lo sviluppo sia sempre allineato a quello della piattaforma Android. La migrazione di un progetto Android da Eclipse ADT ad Android Studio richiede di adattare il progetto alla nuova struttura richiesta da Android Studio. Per semplificare il processo di migrazione, Android Studio fornisce un tool di import automatico che permette una transizione tra il vecchio e il nuovo

IDE in modo veloce e indolore, traducendo il workspace di Eclipse ADE e i file Ant build scripts in un progetto Android Studio con file di compilazione basati su Gradle. Maggiori informazioni possono essere trovate a questo link https://developer.android.com/sdk/installing/migrate.html.
Eclipse e il plugin ADT, utilizzati come IDE per sviluppare applicazioni Android, ora sono supportati solo da community open source. Un altro IDE disponibile e molto utilizzato è quello legato alla piattaforma NetBeans (http://wiki.netbeans.org/IntroAndroidDevNetBeans).

Struttura di un progetto Android

La complessità e le dimensioni di un'applicazione Android possono variare a seconda dei casi, anche se la struttura sarà sempre similare. La Figura 1.7 mostra la struttura di un progetto di esempio, "cap1".

Figura 1.7 – Struttura di un progetto Android.

Creazione del progetto tramite Android Studio e Gradle

Android Studio è un IDE di sviluppo con funzionalità avanzate. Esso ha al suo interno un editor con un sosfisticato sistema di completamento automatico, componenti per