Introduzione

Questo testo si propone come una versione aggiornata ad Android 9.0 o Pie (Api Level 28) della ormai classica Guida alla programmazione Android. Si tratta di una versione molto importante, in quanto non descrive solamente gli strumenti principali della piattaforma Android, ma lo fa utilizzando un nuovo linguaggio, Kotlin, che, come annunciato al *Google I/O* del 2017, è supportato ufficialmente da Google come linguaggio di programmazione per la piattaforma Android. Per questo motivo, l'idea iniziale era quella di dedicare una parte del testo a Kotlin. Ma, data la sua importanza e la quantità di informazioni da descrivere, ho deciso di dargli dignità propria, attraverso un testo dedicato, sempre pubblicato da Apogeo.
La presente guida si compone quindi di tre parti principali, che corrispondono alle nozioni più importanti quando si deve creare una applicazione Android.

- La piattaforma Android.
- Gli *architecture component*.
- Le tecniche di test.

Prima di descrivere in dettaglio ogni sezione e capitolo è importante osservare che tutti gli esempi presentati sono disponibili in GitHub. Specialmente per quello che riguarda gli argomenti della seconda parte, le API sono in continuo aggiornamento per cui i 60 e più *repository* creati consentono di mantenere il tutto sempre aggiornato, oltre a rappresentare un punto nel quale poter discutere di eventuali soluzioni alternative e, perché no, risolvere qualche *bug*. Per accedere all'elenco degli esempi è sufficiente consultare il mio sito personale, `https://www.massimocarli.eu`, oppure la scheda del libro sul sito dell'editore, all'indirizzo `https://bit.ly/apo-android9`.

Parte I: La piattaforma Android

In questa prima parte ci occuperemo dello studio dei concetti di base della programmazione Android. Si tratta di tutti gli strumenti e i concetti che un programmatore Android deve necessariamente conoscere per poter comprendere e utilizzare le varie librerie che andremo a descrivere nella Parte II o altre disponibili online come progetti *open-source*.

Capitolo 1: Introduzione ad Android

In questo primo capitolo ci occupiamo di descrivere le tecnologie alla base della piattaforma Android e realizziamo la nostra prima applicazione. Si tratta di un capitolo di fondamentale importanza, in quanto descrive i principali passi nella realizzazione ed esecuzione di un progetto Android. La creazione di un semplice progetto sarà anche l'occasione per descrivere gli strumenti messi a disposizione da *Android Studio* e le principali caratteristiche di *Gradle* ovvero degli strumenti di *build*. Concludiamo il capitolo con gli strumenti di *logging* e la descrizione dei principali tipi di risorse.

Capitolo 2: Activity e flusso di navigazione

Le `Activity` sono sicuramente tra i componenti più importanti della piattaforma Android e la conoscenza del corrispondente ciclo di vita è essenziale per la realizzazione di applicazioni che utilizzino in modo efficiente tutte le risorse messe a disposizione dai vari dispositivi. In questo capitolo ci occupiamo di descrivere tutto quello che riguarda la creazione di istanze di questo fondamentale componente. In questo capitolo vediamo anche come creare e gestire un tipo particolare di risorse: i documenti XML di layout. Una parte essenziale del capitolo riguarda la descrizione delle modalità di comunicazione tra `Activity`, da cui l'importantissimo concetto di `Intent` e `IntentFilter`. Nella parte finale esaminiamo alcune API molto utili che ci permettono di gestire il *multiwindow* e il PIP (*Picture in Picture*).

Capitolo 3: Fragment

Sebbene non si tratti formalmente di un componente della piattaforma Android, i `Fragment` sono diventati strumenti essenziali per poter gestire in modo efficiente schermi di dimensioni differenti e soprattutto configurazioni differenti. Per configurazioni si intende anche l'utilizzo dell'applicazione in modalità *portrait* o *landscape*. Questo capitolo presenta tutto quello che riguarda i `Fragment` e in particolare come gestire la transizione tra più elementi passando eventualmente dei parametri nel modo più efficiente.

Capitolo 4: ActionBar e Toolbar

Questo capitolo si occupa di due componenti molto importanti, utilizzati in moltissime applicazioni rappresentando, di fatto, un modo standard di utilizzo delle applicazioni. Stiamo parlando di `ActionBar` e `Toolbar`. Si tratta di due componenti con scopi e risultati confrontabili che vengono però rappresentati da API differenti. Vediamo quindi come utilizzare ciascuno di essi, ma soprattutto come adattarli alle nostre esigenze.

Capitolo 5: View e layout

Le `Activity` descritte nel Capitolo 1 sono sicuramente una parte essenziale di ciascuna applicazione Android. Esse permettono la visualizzazione di interfacce utente definite attraverso delle risorse di layout le quali, a loro volta, non fanno altro che comporre quelli che si chiamano `Widget`. Stiamo parlando dei vari `Button`, `Checkbox` e simili. Si tratta in realtà di componenti creati attraverso opportune specializzazioni della classe `View`. Alcune di queste specializzazioni hanno la responsabilità di aggregarne altre e prendono il nome

di *layout*. In questo capitolo vediamo come utilizzare i principali *layout* ed i `Widget` più significativi. Concludiamo il capitolo attraverso l'implementazione di componenti *custom* che prendono il nome di *custom view*.

Capitolo 6: Gestire le liste con RecyclerView

La maggior parte delle applicazioni permette la visualizzazione di elenchi di informazioni. Per questo motivo tutte le piattaforme mettono a disposizione strumenti che non solo permettono la visualizzazione di elenchi ma offrono anche meccanismi di selezione singola o multipla. Android non è da meno e mette a disposizione due tipi di componenti. Il primo è quello che potremmo definire *legacy* ed è descritto dalla classe `ListView`. Il secondo permette invece di gestire le risorse in modo più efficiente e ha come classe di riferimento `RecyclerView`. In questo capitolo realizziamo delle applicazioni che ci permetteranno di vedere nel dettaglio entrambe le soluzioni, dando maggior risalto al concetto di `Adapter`.

Capitolo 7: Gestione della persistenza

Il Capitolo 6 descrive come visualizzare degli elenchi di informazioni in liste o `RecyclerView`. Per farlo servono comunque dei dati, che possono provenire da un server o essere memorizzati localmente. In questo capitolo esaminiamo tutti i possibili modi in cui è possibile memorizzare informazioni nel dispositivo. Vediamo come gestire i `File`, le `SharedPreferences` fino alla gestione del database. In particolare vediamo come gestire una parte importante di ciascuna applicazione, ovvero i *settings*. È bene sottolineare che in questo capitolo vediamo quello che la piattaforma Android mette a disposizione per la gestione della persistenza, mentre nel Capitolo 14 trattiamo una libreria particolare, `Room`, la quale permetterà di gestire un database in modo molto semplice e dichiarativo.

Capitolo 8: Multithreading e servizi

Questo è probabilmente il capitolo più importante del libro, in quanto contiene i concetti fondamentali di programmazione concorrente e gli strumenti messi a disposizione dalla piattaforma Android per sfruttare al massimo la natura multiprocessore dei dispositivi moderni. In questo capitolo vediamo anche come implementare un `Service` e le possibili modalità di gestione dell'IPC (*InterProcess Communication*) ovvero la comunicazione tra processi.

Capitolo 9: Cenni di sicurezza

La sicurezza dei dispositivi è sicuramente uno degli aspetti più importanti di cui è sempre bene tenere conto. Per esaurire l'argomento non basterebbe un libro intero, per cui in questo capitolo descriviamo solo gli aspetti principali, che impattano maggiormente il codice di ciascuna applicazione.

Capitolo 10: Gestione delle animazioni

In questo capitolo ci occupiamo della descrizione di tutte le API che la piattaforma Android mette a disposizione per l'implementazione delle animazioni in senso generale. Per animazione intendiamo una qualunque variazione nel tempo di una proprietà vi-

sibile. Un'animazione è sicuramente una `View` che si sposta sullo schermo, ma anche un colore che esegue un'operazione di dissolvenza. In questo capitolo ci occupiamo anche di transizioni descrivendo i vari meccanismi che permettono di rendere più gradevole la transizione tra `Activity` o `Fragment`.

Parte II: I componenti architetturali

La prima parte del testo contiene i concetti che tutti gli sviluppatori Android devono conoscere per implementare le proprie applicazioni. Negli anni ci si è però accorti che la maggior parte di questi sono strumenti di basso livello e che i casi d'uso da implementare contenevano aspetti comuni che era possibile implementare all'interno di alcuni piccoli *framework*. Questa è stata l'idea alla base della creazione di alcuni componenti architetturali messi a disposizione da Google per la risoluzione di problemi ricorrenti, a ciascuno dei quali ho dedicato un proprio capitolo.
Il lettore potrà verificare come siano stati implementati molti progetti che sono da intendersi come *toy example*. Il lettore potrà infatti scaricare il corrispondente codice da GitHub ed eseguire i propri esperimenti.

Capitolo 11: Lifecycle

Nel Capitolo 1 e 2 introduco un concetto fondamentale di ciascun componente Android ovvero il fatto di essere sottoposto a un ciclo di vita. La creazione di un componente consiste infatti nella creazione di una classe che estende quella dell'ambiente Android (`Activity` per esempio) e quindi eseguire l'*override* di alcuni metodi di *callback*. Il componente architetturale `Lifecycle` permette di ottenere lo stesso risultato utilizzando uno dei principi più importanti della programmazione a oggetti: "*Composition over inheritance*". L'*implementation inheritance* è infatti il grado più forte di dipendenza ed è quindi un qualcosa che è bene utilizzare il meno possibile. *Composition* consiste invece nel definire il comportamento dipendente dal ciclo di vita in un componente distinto, che viene poi usato da quello che si chiama `LifecycleOwner`.
Si tratta di un capitolo di fondamentale importanza, in quanto descrive un concetto utilizzato in tutti i componenti architetturali che andremo a descrivere nei capitoli successivi.

Capitolo 12: LiveData

In questo capitolo trattiamo le API che vanno sotto il nome di `LiveData`. Si tratta dell'implementazione di Google del modello *Reactive*. Un `LiveData` è una sorgente di eventi cui è possibile registrarsi come ascoltatori, o meglio, come *Observer*. Il vantaggio di `LiveData` è che si tratta di un componente *lifecycle aware* ovvero sensibile allo stato del particolare `LifecycleOwner` che può essere una `Activity`, un `Fragment` o in generale un qualunque componente dotato di un ciclo di vita.

Capitolo 13: ViewModel

Anche in questo caso si tratta di un componente che intende risolvere un problema ricorrente nello sviluppo delle applicazioni Android. In questo caso si tratta della gestione dello stato in corrispondenza alla modifica di alcune informazioni di configurazione. Il caso tipico è quello della rotazione del dispositivo, ma ne esistono altri, come il cambio

dell'ora o della lingua del dispositivo. Il `ViewModel` permette di incapsulare una serie di riferimenti e di gestirne la persistenza in memoria a seguito di variazioni di configurazione.

Capitolo 14: Room

Questo capitolo è un po' atipico, in quanto tratta una libreria vera e propria piuttosto che un componente architetturale. `Room` è infatti una libreria, molto simile ad analoghe nel mondo *enterprise*, che permette di gestire la persistenza di alcune entità in modo dichiarativo. In questo capitolo vediamo tutto quello che riguarda `Room`, dalla definizione delle entità, del DAO fino alla gestione del ciclo di vita del database stesso. Parte importante sarà quella relativa alla modalità di test.

Capitolo 15: Data binding

Sviluppando applicazioni per Android ci si rende conto che spesso ci si deve occupare di mappare alcune proprietà di un modello di dati ad altrettanti elementi di visualizzazione, come potrebbe essere una `TextView`. Questa operazione prende il nome di *binding* ed è l'argomento di questo capitolo. Attraverso l'utilizzo di alcuni esempi vedremo come definire dei documenti di *layout* che permettano di eseguire, appunto, delle operazioni di *binding* senza l'utilizzo di alcun tipo di codice.

Capitolo 16: Navigation

Un aspetto che caratterizza ciascuna applicazione è il flusso di navigazione. Le varie possibilità di utilizzo dell'applicazione da parte dell'utente, spesso sono un qualcosa di non definito in modo esplicito, ma che si può dedurre da come i vari `Intent` vengono lanciati tra i componenti. Per questo motivo Google ha deciso di definire alcune API e un *tool* che si chiama *Navigation Editor*, che sono l'argomento di questo capitolo.

Capitolo 17: Paging

Le API di *Paging* sono molto importanti e interessanti in quanto permettono di risolvere un problema che capita spesso durante lo sviluppo di applicazioni Android e non solo. La maggior parte delle applicazioni non fa altro che accedere a informazioni, locali o remote, e quindi visualizzarle all'interno di una `ListView` o `RecyclerView`. Nel caso in cui queste informazioni siano in grande quantità esistono problemi di memoria o comunque di risorse che rappresentano un grosso guaio nel caso in cui si volessero caricare tutti i dati in memoria. Attraverso le API di *Paging* è invece possibile implementare una sorta di caricamento *lazy* dei dati, attraverso un meccanismo di paginazione. Questo capitolo presenta vari progetti che permettono di risolvere il problema utilizzando diversi tipi di architetture; da quello che usa `Room` a quello che accede direttamente alla rete.

Capitolo 18: WorkManager

L'ultimo componente architetturale di cui ci occupiamo si chiama `WorkManager` e permette di gestire l'esecuzione di *task* in cui l'aspetto più importante non è il tempo di esecuzione, ma la garanzia che esso venga eseguito. In questo capitolo vediamo quali sono i tipi di task che possiamo eseguire e le modalità di utilizzo e *testing*.

Parte III: Tecniche di test

La Parte III è dedicata alle tecniche di test, le quali rappresentano una parte fondamentale nello sviluppo di un qualunque prodotto software. Anche in questo caso è impossibile trattare tutte le librerie disponibili e tutte le modalità di test, per cui ci siamo concentrati su quelle principali.

Capitolo 19: Introduzione al testing

In questo capitolo introduttivo creo una semplice applicazione che permetta di mostrare i vari tipi di test che è possibile eseguire su un'applicazione Android. Parlo degli *Unit test* e di come utilizzare JUnit e Mockito. Parlo inoltre degli *instrumentation test* e degli *UI test* per il test funzionale delle applicazioni, utilizzando una libreria come Espresso che approfondisco poi nel Capitolo 21.

Capitolo 20: Test dei componenti standard

In questi ultimi mesi, Google ha dedicato molto tempo alla creazione di alcune librerie di supporto per il test dei componenti principali dell'architettura di Android. In questo capitolo vediamo come sottoporre a test `Activity`, `Fragment` e quindi `Service`, `BroadcastReceiver` e `ContentProvider`. Si tratta di API spesso ancora in versione *beta*, ma comunque molto interessanti.

Capitolo 21: UI Test con Espresso

L'ultimo capitolo è dedicato alla libreria forse più utilizzata per l'esecuzione di *UI Test* in Android, ovvero *Espresso*. In questo capitolo vediamo i suoi componenti principali e come crearne di propri.

Conclusione

In questo testo ho cercato di affrontare tutti i principali concetti e strumenti necessari allo sviluppo di applicazioni Android. I concetti descritti nella Parte I rappresentano la base della piattaforma e non verranno modificati nelle versioni future, se non in alcuni dettagli. Altre funzionalità verranno probabilmente aggiunte. La Parte II è quella forse più volatile, in quanto la creazione dei componenti architetturali è tutt'ora in fase di sviluppo, ma sicuramente i concetti rimarranno gli stessi. Infine, la Parte III, relativa ai test, è molto importante e resterà utile per molto tempo ancora.
Non mi resta che augurarvi una buona lettura.