



Università degli Studi di Milano Bicocca

Scuola di Scienze

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di laurea in Informatica

VR World Explorer: un'applicazione Android per la promozione dell'attività fisica tramite realtà virtuale

Relatore: Prof. Federico Antonio Niccolò Amedeo Cabitza

Co-relatore: Prof. Fabio Sartori

Relazione della prova finale di:

Gianluca Puleri

Matricola 807064

Anno Accademico 2017-2018

*Ai Professori Cabitza e Sartori, per avermi dato la possibilità di approfondire
le mie conoscenze in questo ambito e per la passione
con cui affrontano il proprio lavoro.*

*Ai miei genitori, per avermi supportato (e sopportato) moralmente,
psicologicamente ed economicamente in questi tre anni.*

*A Paolo e Simone, per avermi fatto divertire durante le giornate universitarie
e per aver condiviso con me gioie e dispiaceri di questo percorso.*

*Ad Alessio, per essere da sempre il mio punto di riferimento, per accompagnarmi
e sostenermi in ogni momento.*

*A Giulia, per avermi ascoltato quando avevo bisogno di sfogarmi, per l'infinita
pazienza e per avermi sostenuto con fiducia ed affetto.*

*A Beatrice, per avermi ricordato l'importanza di migliorarsi giorno dopo giorno,
per i consigli e per le critiche.*

*A me stesso, per aver sempre trovato la forza di reagire nei momenti
più bui, per la mia tenacia.*

*A tutti coloro che non ho citato, per aver contribuito a loro modo
al raggiungimento di questo obiettivo.*

Prefazione

La realtà virtuale spesso rimanda la mente ad altre parole chiave come videogames, cinema o turismo; ingenuamente potremmo parlare di realtà virtuale descrivendola solamente come un'altra invenzione del nuovo millennio in ambito ludico. La storia di questa invenzione, seppur sia breve, ha però fin da subito mostrato al mondo le sue potenzialità anche nel settore militare e medico.

È stato proprio quest'ultimo campo ad attirare la mia curiosità e la mia attenzione perché ritengo sia sensazionale e gratificante poter mettere a disposizione la propria conoscenza e gli strumenti informatici a servizio di altre persone che necessitano un aiuto. Per questa motivazione, quando i Professori Cabitza e Sartori mi hanno messo a conoscenza del progetto, ho colto la sfida con grande interesse.

Gianluca Puleri
Milano, Ottobre 2018

Abstract

In questo lavoro di stage si studia come sia possibile realizzare un'applicazione Android in realtà virtuale in grado di poter ricercare e visualizzare carte geografiche terrestri tramite l'ausilio di un CardBoard. Il suo utilizzo è rivolto principalmente ad un'utenza che ha necessità di essere seguita ed incentivata ad un'attività fisica ma può essere utilizzata da chiunque voglia provare un'esperienza in realtà virtuale.

Per cominciare, viene introdotta la motivazione che ha portato all'utilizzo del sistema operativo Android, ne verranno descritte le funzionalità e le caratteristiche che lo distinguono da altri competitors.

L'elaborato procede mostrando la struttura di un'applicazione, descrivendone il ciclo di vita ed i principali elementi che la costituiscono e che rendono possibile l'interazione con l'utente e la visualizzazione dei contenuti.

Viene poi introdotto il software utilizzato per la realizzazione dell'applicativo e la presentazione di altri servizi necessari.

Successivamente viene esposto, nelle sue particolarità più tecniche, il progetto realizzato mostrando al lettore sia parti di codice scritto, sia la logica che vi è dietro ad esso.

A conclusione di questa relazione viene presentato uno studio di Usabilità in merito a due versioni prototipali di questa applicazione al fine di mostrare l'effettiva validità del lavoro svolto.

Indice

1	Introduzione	1
1.1	Perché Android	2
1.2	Android vs. alternative	3
1.3	Perché VR World Explorer	5
2	Struttura di un'app	7
2.1	Cos'è un'app	7
2.2	Struttura di un progetto.	7
2.2.1	AndroidManifest.xml	8
2.3	Struttura logica	10
2.3.1	Activity	10
2.3.2	Intent	12
3	Ambiente di sviluppo	13
3.1	Unity3D	14
3.2	Android Studio	15
3.2.1	Introduzione	15
3.2.2	Funzionalità	15
3.3	Google SDK per Android	17
3.3.1	Introduzione	17
3.3.2	Places SDK per Android	18
3.3.3	Maps SDK per Android	18
3.3.4	Google VR SDK per Android	19
4	Sviluppo di VR World Explorer	21
4.1	MapsActivity	22
4.1.1	Attributi.	22
4.1.2	Metodi	23

4.2	StreetViewActivity	29
4.2.1	Attributi.	29
4.2.2	Metodi	30
4.3	DirectionsJSONParser	33
5	Studio di Usabilità	35
5.1	Valutazione euristica	35
5.2	Test utente	37
5.3	Questionario	39
6	Conclusioni e prospettive future	43
	Postfazione	45
	Bibliografia	47

Introduzione

L'utilizzo degli smartphone, i telefonini intelligenti, è un fenomeno che da pochi anni a questa parte è cresciuto esponenzialmente, grazie alle nuove tecnologie che permettono di costruire prodotti sempre più affidabili e completi ad un prezzo contenuto. Questo ha permesso di potersi estendere ad una fascia di clienti sempre maggiore: i primi smartphone, infatti, avevano un prezzo decisamente elevato e quindi non accessibile a tutti, mentre ora se ne può comprare uno senza troppe pretese per poche decine di euro.

Si è inoltre passati da telefoni la cui unica funzione era quella di effettuare chiamate, a telefoni in grado di avere svariate funzionalità, quali per esempio: fotocamera, sensori, connessione Internet e quant'altro. Da un lato, l'introduzione di queste funzionalità ha reso il dispositivo più complicato da utilizzare, soprattutto per una clientela non giovane, dall'altro lo ha reso più completo e versatile, creando numerose opportunità agli sviluppatori per inventare e re-inventare applicazioni per svolgere le funzioni più svariate.

Esistono però moltissimi tipi di smartphone che montano sistemi operativi differenti: spaziando da Android a iOS, da Windows Phone a BlackBerry OS. Ognuno di questi ha caratteristiche diverse, che lo rendono migliore per una certa clientela piuttosto che per un'altra. Tra i tanti sistemi operativi presenti, uno di cui vale la pena parlare è Android.

Tutto il mondo Android ruota attorno alla parola *open source*: infatti si ha libero accesso a tutti gli strumenti utilizzati dagli sviluppatori stessi di Android e quindi il potenziale delle applicazioni sviluppate da terzi non è soggetto a limitazioni, visto anche che applicazioni native e di terze parti hanno lo stesso peso. Inoltre, chiunque può sviluppare per Android senza dover acquisire software o licenze particolari.

1.1 Perché Android

Chi oggi vuole acquistare uno di questi “telefonini intelligenti”, di sicuro si imbatte in un logo molto conosciuto: un simpatico robottino verde. Questo è il simbolo di Android (fig. 1.1), una piattaforma di esecuzione gratuita per dispositivi mobili, creata nel 2003 da Andy Rubin e acquistata da Google nel 2005.

Va fatta da subito una precisazione sul termine piattaforma, spesso abusato: una piattaforma è una base software/hardware sulla quale vengono eseguite o sviluppate applicazioni. È importante quindi distinguere le piattaforme di esecuzione, cioè quelle dedicate all'esecuzione di programmi, comunemente note come sistema operativo, da quelle di sviluppo, cioè quelle utilizzate per sviluppare programmi.

I due concetti non sono comunque così distanti: una piattaforma di sviluppo solitamente è anche di esecuzione, ma non necessariamente una piattaforma di esecuzione è anche di sviluppo^[1].



Figura 1.1 Logo Android

Come abbiamo detto, nel mercato sono attualmente disponibili molte piattaforme di esecuzione: perché allora molti sviluppatori scelgono proprio Android?

Quando un programmatore decide di sviluppare un'app, deve anche scegliere su quale piattaforma questa potrà essere eseguita.

Si ritrova quindi davanti a due strade:

- Scegliere un linguaggio come Java, per Android, che è conosciuto, diffuso e molto utilizzato ma che dà forse troppe libertà.
- Scegliere un linguaggio meno flessibile ma che permetta di concentrarsi di più sul programma e meno sull'interfaccia, in quanto questa sarà gestita dal sistema operativo. Anche qui si arriva ad un bivio: il C# di Windows Phone oppure l'Objective-C di iOS.

Il linguaggio Java, rispetto al C, gestisce automaticamente il ciclo di vita di un'applicazione e anche l'allocazione della memoria, rendendo più semplice lo sviluppo.

Android inoltre, come è stato detto, è completamente *open source*, quindi rende possibile reperire, modificare e ridistribuire il codice sorgente adattandolo ad ogni dispositivo e creandone la propria versione personalizzata ed ottimizzata. Ha una gestione di grafica e audio di alta qualità, le applicazioni native e di terze parti sono di uguale importanza e possono essere aggiunte o rimosse senza alcun vincolo.

La scelta di Android non porta tuttavia solo vantaggi dal punto di vista informatico, ma anche dal punto di vista economico. Pubblicare applicazioni sullo store di Android (Google Play), ha un costo di iscrizione di 25€ ed è possibile programmare l'app con tutti i sistemi operativi per desktop più diffusi: Windows, OS X e Linux.

Scegliere iOS invece comporta un costo 4 volte superiore per l'iscrizione all'App Store e serve necessariamente un computer Mac per programmare, anche questo molto costoso.

1.2 Android vs. alternative

Vediamo ora le principali differenze che distinguono i diversi sistemi operativi che dominano il mercato.

Android è la piattaforma più diffusa^[2], seguita in seconda in posizione da Apple. Il sistema proprietario Apple è sicuramente più *user friendly*, cioè è più adatto ad una clientela poco abituata ad utilizzare smartphone, ma d'altro canto per poter sincronizzare il telefono con il PC (scambio di foto, musica...) o ottenere aggiornamenti software, richiede il costante utilizzo di programmi forniti da Apple, come iTunes.

La libertà di poter installare qualsiasi applicazione, anche non ufficiale, rende Android migliore per gli sviluppatori, che possono così provare le loro applicazioni sui terminali senza dover per forza ricorrere ad un emulatore. Poter ottenere il codice sorgente ha come effetto la possibilità di avere sempre diverse versioni del sistema operativo ottimizzate per ogni tipo di telefono e con funzionalità che la versione originale non aveva.

Una famosa distribuzione di Android modificata, detta Custom ROM, è la *LineageOS*.

Essa non è nient'altro che il frutto di una comunità di sviluppatori che hanno deciso di mantenere vivo il progetto *CyanogenMod* dopo che l'azienda proprietaria ha annunciato l'abbandono dello sviluppo.

L'utilizzo di una custom ROM dà all'utente i *privilegi di root*, cioè quei permessi che consentono all'utente di fare qualsiasi cosa col telefono, anche accedere o modificare risorse il cui accesso era stato impedito dalla ROM originale. Si può fare qualcosa di simile anche in iOS tramite il cosiddetto *jailbreak*, che permette sì di avere nuove funzionalità, ma la licenza del sistema rimane comunque di Apple.

Dal punto di vista della flessibilità, iOS è molto meno personalizzabile di Android, il quale lascia maggior voce in capitolo all'utente.

Guardando al kernel, invece, entrambe sono basate su kernel Linux. Symbian, invece, sistema proprietario Nokia fino a poco tempo fa, è stato abbandonato e sostituito da Windows Phone, sviluppato da Microsoft.

Quest'ultimo ha introdotto una grande novità: le *tiles*. Le *tiles* sono delle tessere che si trovano al posto delle classiche icone e che rendono la ricerca di app e informazioni nel proprio telefono estremamente rapida. La sostanziale differenza tra i due sistemi operativi sta nel fatto che Windows Phone si concentra maggiormente sulla condivisione di informazioni sui social network. Per il resto i sistemi sono quasi equivalenti, se non per il fatto che lo store di Android è sicuramente più ricco di app, motivo per il quale molta gente opta per il robottino verde piuttosto che per Windows Phone.



Figura 1.2 Android Pie

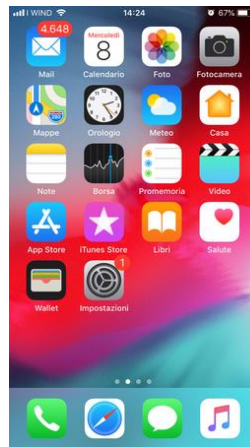


Figura 1.3 iOS 12



Figura 1.4 Windows Mobile 10

1.3 Perché VR World Explorer

Come è appena stato detto, negli ultimi anni vi è stato un forte incremento di smartphone con sensori integrati. L'aggiunta di questi nuovi componenti ha fatto sì che molti sviluppatori abbiano cominciato a progettare diverse applicazioni per agevolare gli utenti nello svolgere attività fisica.

Ma allora perché scegliere di utilizzare VR World Explorer anziché una delle applicazioni già esistenti e consolidate sul mercato come ad esempio Runtastic, RunKeeper o Nike+?

Il mondo delle applicazioni fitness propone funzionalità adatte a persone che già svolgono una media o intensa attività fisica, non riuscendo però a coinvolgere quella parte di persone poco dinamiche.

VR World Explorer è stata pensata per essere un'applicazione differente rispetto a quelle sopra citate. L'esigenza di avere un'app di questo tipo è, quindi, molto semplice: stimolare coloro che hanno necessità di essere seguiti e incentivati ad un'attività fisica.

In particolare, utilizzando la modalità VR, l'utente può percorrere il percorso ancor prima di eseguirlo effettivamente, in modo tale da constatare con i propri occhi se il tragitto selezionato è di suo gradimento e permettergli di essere a proprio agio quando dovrà effettivamente percorrerlo.

Inoltre, tra le funzionalità principali dell'applicazione vi è anche la possibilità di visualizzare semplicemente un luogo (anziché un percorso) in modalità VR; di cercare una località o calcolare un percorso manualmente o tramite l'inserimento a tastiera; di cambiare lo stile della mappa (ad esempio satellite, ibrida, ecc.); di geolocalizzare la propria posizione tramite l'apposito tasto; di visualizzare lo StreetView di un determinato luogo con la possibilità di effettuare lo zoom. Esse, la rendono un'applicazione di semplice utilizzo ma al contempo completa e, pertanto, utilizzabile agevolmente da persone di età, genere e professioni differenti.

Struttura di un'app

2.1 Cos'è un'app?

Un'app è un'applicazione software dedicata ai dispositivi di tipo mobile^[3]. In particolare, le applicazioni Android vengono scritte in linguaggio Java. Gli strumenti dell'SDK compilano tutto il codice e producono un file APK (*Android Package*) e, grazie a tale file, l'applicazione può essere installata sul dispositivo.

2.2 Struttura di un progetto

Quando creiamo un nuovo progetto, viene generata una struttura a cartelle e sottocartelle, che si mantiene anche per applicazioni più complesse e di maggiori dimensioni.

Queste cartelle sono comuni a tutte le applicazioni Android e suddividono in maniera ordinata i file necessari a far girare l'applicazione.

Le directory principali sono:

- *manifests*: contiene il file *AndroidManifest.xml* ovvero la definizione XML degli aspetti principali dell'applicazione, come ad esempio la sua identificazione (nome,

versione, logo), i suoi componenti (views, messaggi) o i permessi necessari per la sua esecuzione.

- *java*: contiene tutto il codice sorgente dell'applicazione, il codice dell'interfaccia grafica, le classi ecc. Inizialmente Android Studio genererà in automatico il codice della prima activity.
- *res*: contiene tutti i file e le risorse necessarie del progetto (immagini, file xml per il layout, menù, stringhe, stili, colori, icone).

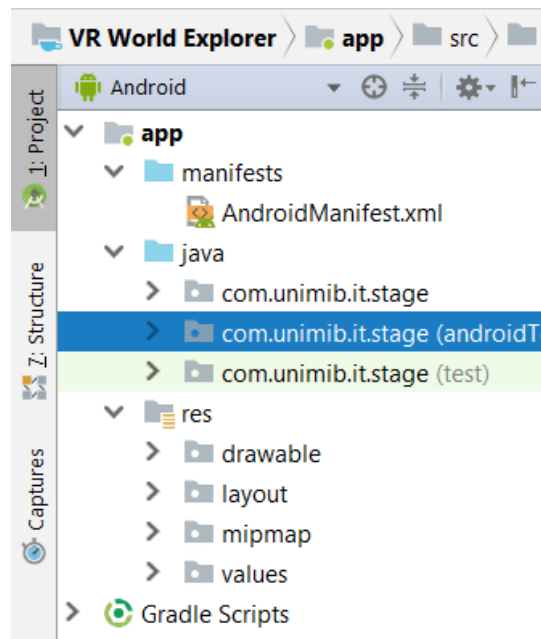


Figura 2.1 Struttura di un progetto

2.2.1 AndroidManifest.xml

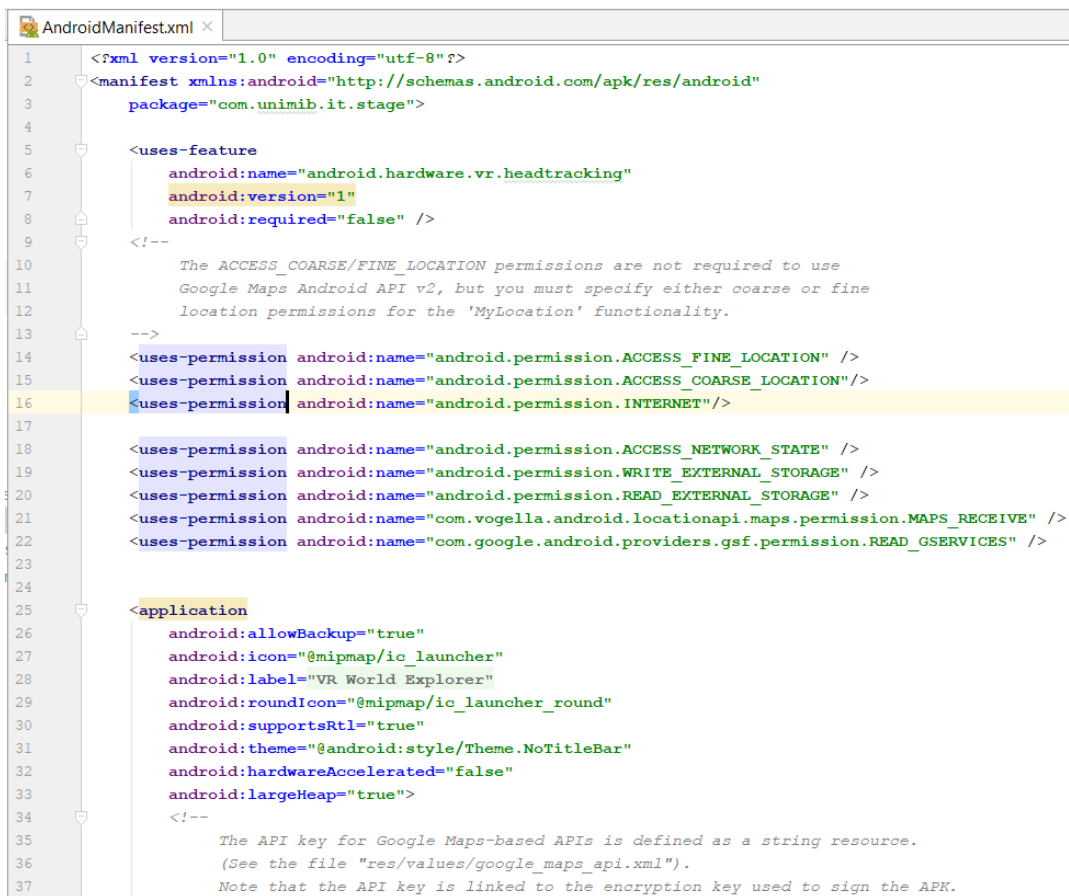
Come detto, all'interno della directory *manifests* vi è il file *AndroidManifest.xml*, di importanza fondamentale dato che, in sua assenza o in caso di sua errata compilazione, può compromettere l'utilizzo dell'intera applicazione.

Esso viene generato automaticamente dal plugin ADT una volta creato il progetto ed è memorizzato nella directory principale. Si distingue dagli altri file XML per la presenza nell'intestazione del tag `<manifest>`, che serve per includere i nodi che definiscono i componenti dell'applicazione, l'ambiente di sicurezza e tutto ciò che fa parte dell'applicazione.

Nella sua interezza contiene tutte le informazioni specificate durante la creazione del progetto, l'icona, la versione dell'applicazione e tutte le Activities e i Services che compaiono al suo interno, le quali saranno trattate successivamente.

Un'altra funzione importante dell'AndroidManifest.xml è la dichiarazione dei permessi di cui necessita l'applicazione. Di default infatti, l'accesso a particolari funzioni del dispositivo quali, ad esempio, la connessione ad Internet, il Bluetooth o la memoria non sarebbe permesso.

Pertanto, queste risorse per poter essere utilizzate devono essere dichiarate proprio all'interno di questo file. In questo modo, sia il gestore dei pacchetti può avere sotto controllo i privilegi necessari all'applicativo, sia l'utente è sempre avvisato prima dell'installazione su quali risorse l'applicazione potrà andare ad accedere.



```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.unimib.it.stage">
4
5     <uses-feature
6         android:name="android.hardware.vr.headtracking"
7         android:version="1"
8         android:required="false" />
9
10    <!--
11        The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
12        Google Maps Android API v2, but you must specify either coarse or fine
13        location permissions for the 'MyLocation' functionality.
14    -->
15    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
16    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
17    <uses-permission android:name="android.permission.INTERNET"/>
18
19    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
20    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
21    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
22    <uses-permission android:name="com.vogella.android.locationapi.maps.permission.MAPS_RECEIVE" />
23    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
24
25    <application
26        android:allowBackup="true"
27        android:icon="@mipmap/ic_launcher"
28        android:label="VR World Explorer"
29        android:roundIcon="@mipmap/ic_launcher_round"
30        android:supportRtl="true"
31        android:theme="@android:style/Theme.NoTitleBar"
32        android:hardwareAccelerated="false"
33        android:largeHeap="true">
34
35        <!--
36            The API key for Google Maps-based APIs is defined as a string resource.
37            (See the file "res/values/google_maps_api.xml").
38            Note that the API key is linked to the encryption key used to sign the APK.
  
```

Figura 2.2 AndroidManifest.xml

2.3 Struttura logica

Ogni applicazione Android, indipendentemente dalla finalità che si prefigge, affida le sue funzionalità a diversi tipi di componenti:

- Activity
- Intent
- Service
- Content Provider
- Broadcast Receiver

Service, Content Provider e Broadcast Receiver sono componenti adatti a particolari tipi di applicazioni, rispettivamente per svolgere servizi in background, per interagire con le notifiche o per condividere i dati. I componenti che, invece, sono alla base della quasi totalità delle applicazioni sono Activity e Intent. Se si rende necessario infatti visualizzare delle informazioni, configurare delle impostazioni o richiedere dei dati all'utente tramite interfaccia grafica, non si può prescindere dall'utilizzare questi componenti.

2.3.1 Activity

La classe Activity è uno degli elementi centrali della programmazione delle app in Android. Un'activity è una finestra che contiene l'interfaccia dell'applicazione con lo scopo di gestire l'interazione tra l'utente e l'applicazione stessa^[4].

Le applicazioni possono definire un qualsiasi numero di activity per poter trattare diversi parti del software: ognuna svolge una particolare funzione e deve essere in grado di salvare il proprio stato in modo da poterlo ristabilire successivamente come parte del ciclo di vita dell'applicazione.

Nello *switch* tra un'activity e l'altra è possibile fare in modo che la prima restituisca un particolare valore alla seconda.

Le activity sospese sono conservate nell'*activity stack*, ovvero una pila di tipo LI-FO (last-in-first-out). È dispendioso conservare molte activity nello stack ed è compito del sistema operativo provvedere a rimuovere quelle inutili.

A partire dalla versione Android 3.0 sono stati introdotti i *fragments*. È possibile combinare più fragment in una singola activity, per costruire un'interfaccia utente multipannello, e riutilizzare un fragment in molteplici activities.

I fragments hanno un ciclo di vita uguale o minore del ciclo di vita dell'activity che li ospita, possono rilevare eventi di input e possono essere aggiunti (o rimossi) dinamicamente mentre l'activity è in esecuzione. Non solo il ciclo di vita dei fragments è fortemente influenzato dall'activity, ma anche il loro stato.

I fragments permettono di gestire tali design senza la necessità di effettuare cambiamenti complessi alla struttura della vista. Suddividendo il layout di un'activity in fragments, si è in grado di modificare l'aspetto della propria activity in fase di esecuzione e di separare la parte grafica da quella decisionale, in modo da adattare l'applicazione al dispositivo in uso a runtime.

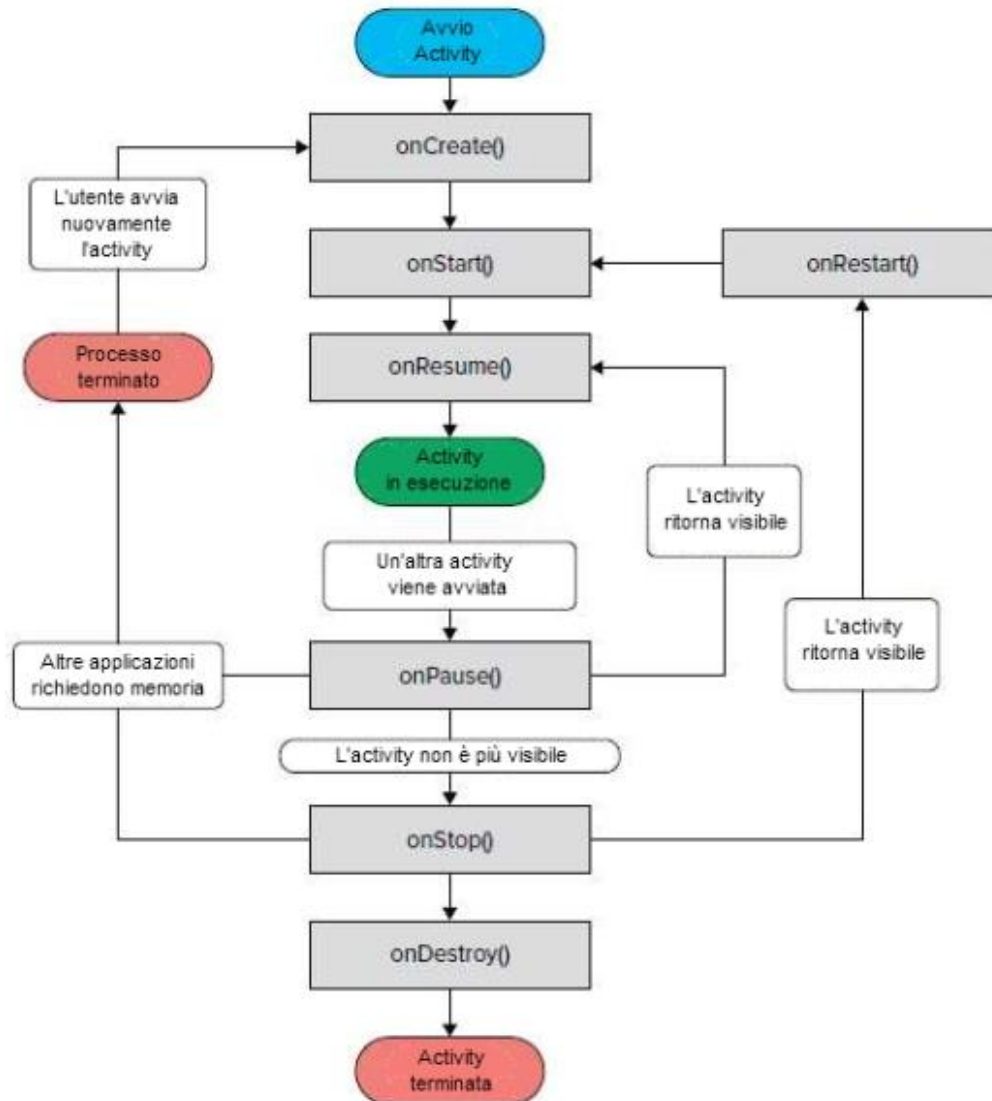


Figura 2.3 Ciclo di vita di un'activity^[5]

2.3.2 Intent

Come detto, un'applicazione Android può contenere una o più activity. Quando sono presenti più activity solitamente è necessario fornire anche una funzionalità di navigazione tra di esse e per fare questo si utilizzano i cosiddetti Intent^[6].

Possiamo vedere quindi un intent come un'azione che possiamo far invocare ad Android: un intent può essere molto specifico ed essere richiamabile da una specifica activity oppure può essere generico e disponibile per qualsiasi activity che rispetta determinati criteri.

La sua struttura dati è composta da:

- *action*: l'azione da eseguire (es. ACTION VIEW, ACTION DIAL, ACTION MAIN, ACTION EDIT, ecc.).
- *data*: i dati su cui eseguire l'azione.

Schematizzando, un intent può essere utilizzato per:

- Trasmettere l'informazione per cui un particolare evento (o azione) si è verificato;
- Esplicitare un'intenzione che un'activity o un service possono sfruttare per eseguire un determinato compito o azione, solitamente con o su un particolare insieme di dati;
- Lanciare una particolare activity o service;
- Supportare l'interazione tra qualsiasi applicazione installata sul dispositivo Android, senza doversi preoccupare di che tipo di applicazione sia o di quale componente software gli intent facciano parte.

Lavorare senza intent è quasi impossibile e comunque sconsigliato: il suo utilizzo infatti trasforma ogni dispositivo Android in una collezione di componenti indipendenti ma facenti parte di un singolo sistema interconnesso^[7].

Gli intent, inoltre, possono essere di due tipi: espliciti o impliciti.

In quelli espliciti si specifica, all'interno del costruttore, l'oggetto da eseguire (solitamente il nome dell'activity).

In quelli impliciti si specifica l'azione da eseguire senza sapere a chi effettivamente sarà delegata l'esecuzione.

Ambiente di sviluppo

Inizialmente si è utilizzato l'ambiente di sviluppo Unity3D, che sembrava prestarsi perfettamente all'esigenza di creare una realtà tridimensionale. Successivamente però, sebbene buona parte del lavoro fosse stato svolto, è stata abbandonata questa pista procedendo verso un altro ambiente di sviluppo: Android Studio. Nel prossimo paragrafo verranno elencate più nel dettaglio le motivazioni che hanno portato a questo cambiamento.

Per la realizzazione del progetto è stato necessario avere a disposizione un pc per lo sviluppo del codice, uno smartphone Android e un visore compatibile con Google CardBoard. Rispettivamente, è stato scelto di utilizzare un Lenovo Z51, un Samsung Galaxy S7 e un visore BOBOVR Z5. Il pc e lo smartphone erano già in mio possesso, il visore, invece, è stato necessario acquistarlo. Non si entrerà nel dettaglio delle caratteristiche dei dispositivi, perché non rilevanti ai fini di questa tesi; nei prossimi paragrafi, invece, verranno illustrati i punti chiave del software Android Studio e del ruolo fondamentale che hanno avuto gli SDK di Google Maps e Google VR per lo svolgimento del progetto.

3.1 Unity3D

Unity3D è un software di sviluppo principalmente utilizzato per la realizzazione di videogames e animazioni tridimensionali; per questa ragione, esso sembrava adattarsi perfettamente all'esigenza di realizzare un'ambiente tridimensionale nel quale ogni panorama a 360°, fornito dinamicamente da StreetView, definiva una proiezione su una sfera. L'utente, inoltre, essendo posizionato nel centro di questa sfera, avrebbe avuto la possibilità di guardarla nella sua interezza e senza alcuna distorsione visiva provocata dal raggio di distanza più (o meno) ampio rispetto alla superficie sferica. Per di più, sarebbe stato in grado anche di muoversi da una sfera all'altra proprio come se stesse camminando o correndo.

Sebbene però vi siano questi vantaggi, la realizzazione su Unity3D presenta alcune difficoltà che sono più facilmente risolvibili con Android Studio.

Per esempio, per lo scaricamento automatico e in modo dinamico delle immagini panoramiche in base alla posizione dell'utente nel mondo, è necessario l'acquisto di specifici *assets* dall'Asset Store di Unity3D che, invece, è possibile reperire in modo del tutto gratuito utilizzando gli SDK che verranno presentati.

Un altro problema è rappresentato dalla numerosità di sfere che vi si sarebbero create nella percorrenza di un lungo tragitto che, invece, è risolvibile svincolandosi dal concetto di associazione di un panorama a una sfera. Pertanto, è stato preferito delegare a Google il compito di fornire il tragitto nella maniera più opportuna tramite l'utilizzo dei loro SDK.

Inoltre, dovendo essere un'applicazione in grado di poter ricercare (e non solo visualizzare) carte geografiche terrestri, fornire all'utente un planisfero e permettere la sua navigazione con funzionalità come una barra di ricerca o lo zoom tramite l'utilizzo di Unity3D risulterebbe di una dispendiosità superflua che, invece, è facilmente trascurabile tramite l'utilizzo delle classi di Android Studio.

Infine, avendo ipotizzato un possibile inserimento di questo progetto all'interno di una serie di progetti più ampi è stata presa in considerazione la vasta diffusione di applicazioni sviluppate su Android Studio a discapito di una minoranza realizzata con Unity3D e, dunque, la facilità con cui questo progetto possa integrarsi con altri.

Per tutto ciò, è stato concluso che Unity3D è un ambiente di sviluppo più adatto alla progettazione di applicazioni *stand-alone* per la realizzazione di percorsi brevi e predefiniti in cui non vi è la necessità di scaricare runtime immagini panoramiche.

3.2 Android Studio

3.2.1 Introduzione

Android Studio^[8] è l'ambiente di sviluppo integrato ufficiale firmato Google, basato su IntelliJ IDEA, per sviluppare su piattaforma Android, liberamente disponibile per Windows, Mac OS X e Linux, sotto licenza Apache 2.0.

È un editor intelligente capace di offrire: completamento avanzato del codice, refactoring e analisi; queste caratteristiche rendono lo sviluppatore più produttivo e veloce nel completamento dell'applicazione.

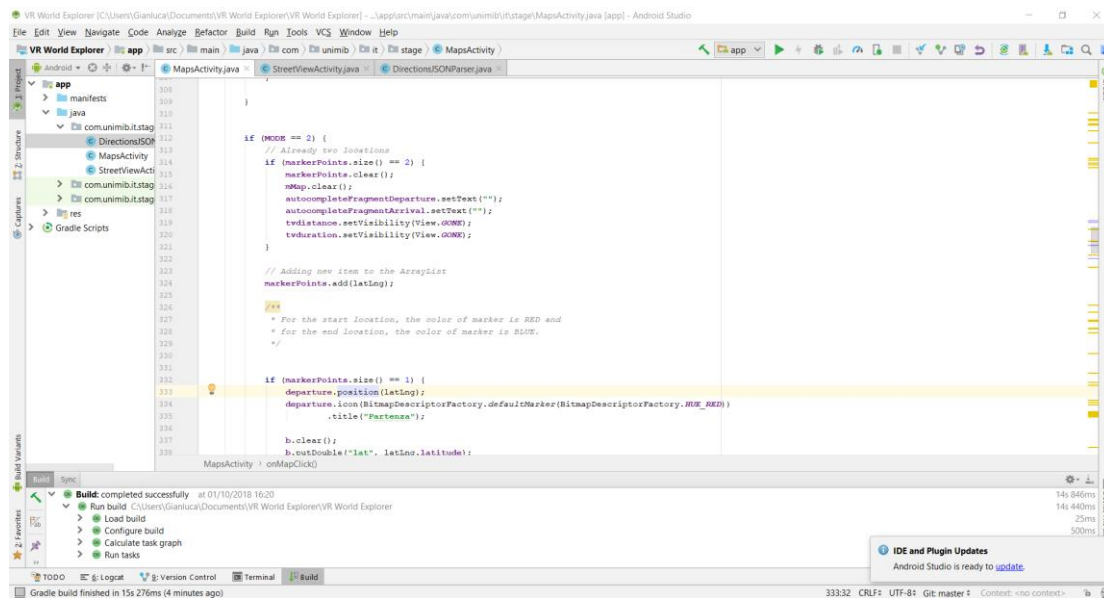


Figura 3.1 Schermata di sviluppo di Android Studio

3.2.2 Funzionalità

Con Android Studio risulta molto semplice sviluppare applicazioni multi-schermo (smartphone Android, tablet, Android Wear, TV Android, Android Auto e Google Glass) poiché è integrato un modulo capace di facilitare la gestione delle risorse e delle risoluzioni. Infatti, ad ogni esecuzione/debug dell'applicazione è possibile scegliere il profilo del dispositivo Android da emulare tra quelli più comuni.

Altri punti di forza di Android Studio sono la possibilità di avviare nuovi progetti utilizzando template messi a disposizione dal software stesso e la possibilità di creare APK multipli dell'applicazione con caratteristiche differenti specificate nel file

build.gradle del progetto. Entrando nel dettaglio, Gradle è un sistema open source per l'automazione dello sviluppo, che introduce un Domain-Specific Language (DSL) basato su Groovy, al posto della più tradizionale modalità XML usata per dichiarare la configurazione del progetto. Gradle è stato progettato per sviluppi multi-progetto che possono crescere fino a divenire abbastanza grandi e supporta sviluppi incrementali determinando in modo intelligente quali parti del build tree sono aggiornate (up-to-date), in modo che tutti i processi che dipendono solo da quelle parti non avranno bisogno di essere ri-eseguiti; così facendo, il software riduce significativamente il tempo di costruzione del progetto, in quanto, durante il nuovo tentativo di costruzione, verranno eseguite solo le attività il cui codice è effettivamente stato alterato a partire dall'ultima costruzione completata. Gradle supporta anche la costruzione del progetto per processi concorrenti, il che consente di svolgere alcuni compiti durante la costruzione (ad esempio, i test automatizzati attraverso gli unit test), eseguiti in parallelo su più core della medesima CPU, su più CPU o su più computer^[9].

Inoltre, è possibile configurarlo con GitHub^[10] in modo da importare codice dal canale di Google oppure esportare e rendere pubblico il proprio progetto.

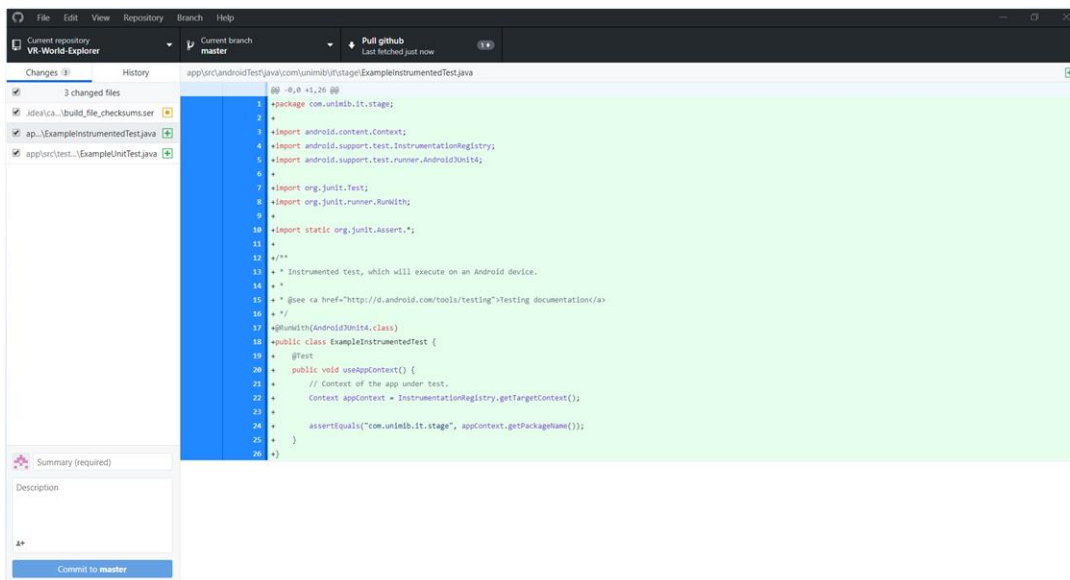


Figura 3.2 Schermata di GitHub

3.3 Google SDK per Android

3.3.1 Introduzione

Per la realizzazione del progetto è stato indispensabile usufruire di alcuni SDK (*Software Development Kit*) messi a disposizione dal fornitore Google.

Gli SDK sono una suite di strumenti di sviluppo utilizzabili per realizzare, in maniera più semplice e immediata, programmi e applicativi per ogni genere di piattaforma^[11].

Essi possono variare considerevolmente in quanto a dimensioni e tecnologie utilizzate, ma tutti possiedono alcuni strumenti fondamentali:

- un compilatore, per tradurre il codice sorgente in un eseguibile;
- librerie standard dotate di interfacce pubbliche dette *API - Application Programming Interface*;
- documentazione sul linguaggio di programmazione per il quale l'SDK è stato sviluppato e sugli strumenti a disposizione nell'SDK stesso;
- informazioni sulle licenze da utilizzare per distribuire programmi creati con l'SDK.

All'interno dell'SDK le API svolgono un ruolo chiave. Come anticipato, esse sono delle librerie dotate di interfacce pubbliche che sviluppatori e programmatori terzi possono utilizzare per espandere le funzionalità di programmi, applicazioni e piattaforme di vario genere (software e non solo). Rappresentano, quindi, l'interfaccia aperta attraverso la quale interagire con programmi (o parti di essi) altrimenti inaccessibili^[12].

La finalità è quella di ottenere un'astrazione a più alto livello, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. Le API permettono infatti di evitare ai programmatori di riscrivere ogni volta tutti i metodi necessari al programma da zero, rientrando quindi nel più vasto concetto di riuso di codice^[13].

Google ha sviluppato diversi SDK che consentono la comunicazione con i suoi servizi e la loro integrazione con altri. Esempi di questi includono Search, Gmail, Translate o Google Maps. Le app di terze parti possono dunque utilizzare queste interfacce per sfruttare o estendere la funzionalità dei servizi esistenti.

In particolare, per la realizzazione di questo progetto sono stati utilizzati tre SDK: Places SDK per Android, Maps SDK per Android e Google VR SDK per Android.

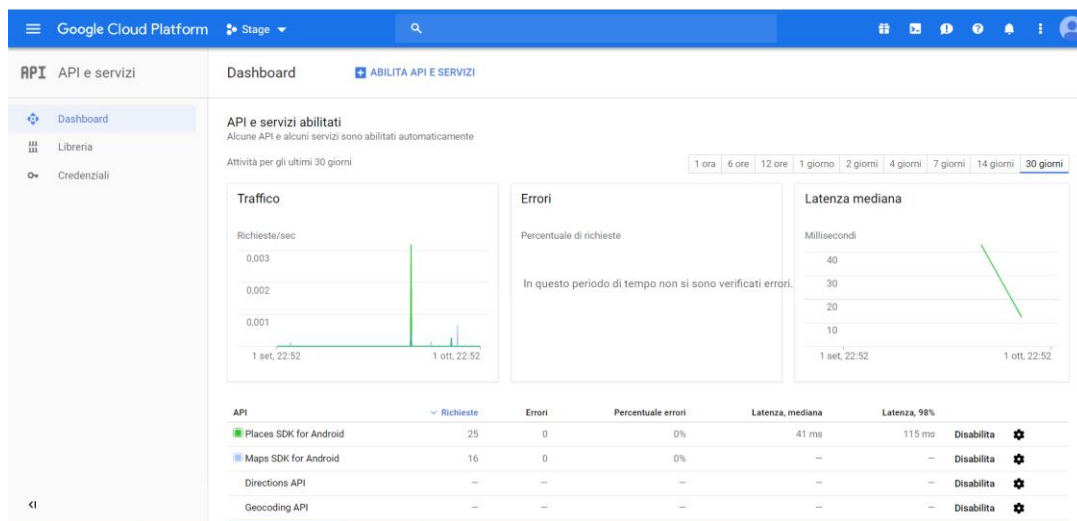


Figura 3.3 Console Google per la gestione dei servizi

3.3.2 Places SDK per Android

Places SDK per Android consente di creare app che riconoscono la posizione e che rispondono contestualmente alle attività commerciali locali e ad altri luoghi vicino al dispositivo. È possibile utilizzare le API al suo interno per aggiungere barre di ricerca, pulsanti per la geolocalizzazione ed altro^[14].

3.3.3 Maps SDK per Android

Con l'SDK di Maps per Android, è possibile aggiungere mappe basate sui dati di Google Maps alla propria applicazione. Al suo interno vi sono diversi pool di API relativi alle mappe, allo StreetView, ai dispositivi wearable ecc. Per questo applicativo è stato necessario l'utilizzo delle Google Maps API e delle Google StreetView API.

Il pool di API relativo alle mappe gestisce automaticamente l'accesso ai server di Google Maps, il download dei dati, la visualizzazione delle mappe e la risposta ai gesti delle mappe, come ad esempio zoom, rotazione e scorrimento. È possibile utilizzare anche esplicitamente altre API per aggiungere segnalibri, per modificare lo stile di visualizzazione di una particolare area della mappa ecc. Questi oggetti forniscono

informazioni aggiuntive per le posizioni delle mappe e consentono l'interazione dell'utente con la mappa.

Il pool di API relativo allo StreetView, invece, offre un servizio per ottenere e manipolare le immagini utilizzate in Google StreetView. Le immagini vengono restituite come panorami. Ogni panorama è un'immagine, o una serie di immagini, che offre una visione completa a 360 gradi da un'unica posizione. Il risultante panorama a 360 gradi definisce una proiezione su una sfera.

3.3.4 Google VR SDK per Android

Google VR fornisce SDK per molti ambienti di sviluppo popolari. Questi SDK forniscono API native per le principali funzionalità di VR come input dell'utente (tramite movimento della testa e l'uso di un controller) che è possibile utilizzare per creare nuove esperienze VR su Daydream o Cardboard.

Sviluppo di VR World Explorer

L'intero progetto è disponibile all'indirizzo web <https://github.com/GPuleri/VR-World-Explorer>. All'interno di esso è disponibile tutto il codice, il logo e i file necessari all'installazione dell'applicazione.

Per questa applicazione sono state ideate 2 activities:

- *MapsActivity*
- *StreetViewActivity*

Oltre queste due activities è stata necessaria l'implementazione di una classe di supporto: *DirectionsJSONParser*.

Nei prossimi paragrafi verrà presentata la classe e le activities create, mostrando al lettore alcuni pezzi di codice ed esponendo la logica che vi è dietro ad essi.



Figura 4.1 Logo VR World Explorer

4.1 MapsActivity

MapsActivity è l'activity iniziale che viene eseguita all'avvio dell'applicazione; essa permette l'esplorazione del planisfero, la ricerca di un luogo, il calcolo di un percorso e altre funzionalità secondarie.

Si presenta come una classe composta da diversi attributi e metodi e che implementa varie interfacce.

4.1.1 Attributi

```
public class MapsActivity extends FragmentActivity implements OnMyLocationButtonClickListener,
    OnMyLocationClickListener, OnMapReadyCallback, AdapterView.OnItemClickListener,
    GoogleMap.OnMapClickListener, GoogleMap.OnInfoWindowClickListener,
    GoogleMap.OnMarkerClickListener, GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener {

    private GoogleMap mMap; // map
    private Bundle b; // object used for passing parameters to other activities

    Button bsearch_location; // button for searching a location
    Button bsearch_route; // button for searching a route
    PlaceAutocompleteFragment autocompleteFragmentDeparture; // departure/location search bar
    PlaceAutocompleteFragment autocompleteFragmentArrival; // arrival search bar
    View view1;
    View view2;
    TextView tvdistance;
    TextView tvduration;

    public int MODE; // search mode (1=location, 2=route)
    private MarkerOptions departure = new MarkerOptions(); // departure marker
    private MarkerOptions arrival = new MarkerOptions(); // arrival marker

    ArrayList<LatLng> markerPoints;
    ArrayList<LatLng> points = null;

    private Spinner mSpinner;
```

Figura 4.2 Attributi della classe MapsActivity

Tra questi attributi meritano un'attenzione particolare:

- **mMap**, un oggetto utilizzato per la ricezione di un planisfero interattivo fornito da Google Maps e che consente l'interazione con l'utente. Con l'utilizzo di questo oggetto il programmatore ha la possibilità, infatti, di aggiungere alla mappa un pulsante per la geolocalizzazione, la bussola, i tasti per effettuare lo zoom. Ha, inoltre, la possibilità di consentire l'aggiunta/rimozione di *bookmarks* e di potere far utilizzare all'utente le proprie dita sul touchscreen per muoversi, ruotare e zoomare sulla mappa.
- **bsearch_location** e **bsearch_route**, due pulsanti in mutua esclusione utilizzati, rispettivamente, per il passaggio alla modalità "cerca un luogo" e "cerca un percorso".

- **autocompleteFragmentDeparture** e **autocompleteFragmentArrival**, due barre di ricerca utilizzate per trovare un singolo luogo o calcolare un percorso.
- **MODE**, è un attributo utilizzato puramente a scopo logico, pertanto, non modifica in alcun modo la grafica dell'applicazione ed è trasparente all'utente. Il suo fine è quello di facilitare la lettura del codice ed evitare chiamate a metodi per il controllo della modalità attuale (ricerca luogo/calcolo percorso).

4.1.2 Metodi

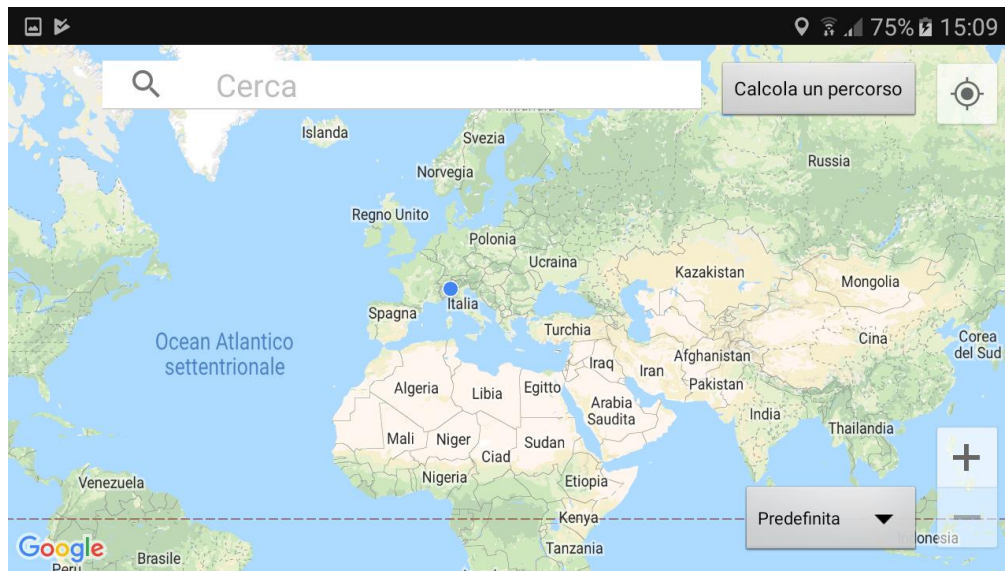


Figura 4.3 Screenshot avvio applicazione

La figura 4.3 rappresenta la schermata che viene presentata all'utente all'avvio dell'applicazione.

Il suo layout è stato sviluppato utilizzando il linguaggio XML ed il codice è disponibile all'interno della directory *res/layout/activity_maps.xml*.

La sua creazione, invece, avviene tramite i metodi *onCreate()* e *onMapReady()*.

Come descritto nel paragrafo dedicato al ciclo di vita di un activity al capitolo 2, *onCreate()* è il primo metodo ad essere chiamato e per questo motivo è di fondamentale importanza per la creazione di tutti gli elementi necessari al funzionamento dell'app.

Dal codice è possibile notare come esso svolga diverse funzionalità: inizializza la mappa e crea le istanze degli oggetti della classe, imposta *MODE=1* (ovvero che la modalità attuale è la ricerca di un luogo), abilita il pulsante per il passaggio alla modalità di calcolo di un percorso e disabilita quello per il passaggio alla ricerca di un luogo (in quanto è già quella attualmente utilizzata).

Successivamente rende disponibile un'unica barra di ricerca in modo da poter ricercare un luogo e non consentire il calcolo di un percorso tra 2 punti.

Inizialmente, nella prima versione dell'applicazione, non veniva mostrata alcuna barra di ricerca e, per la ricerca di un luogo o il calcolo di un percorso, venivano automaticamente eseguiti rispettivamente uno o due *widget* di Google con funzionalità di predizione e auto-completamento del testo. Come verrà mostrato nel successivo capitolo, su questa applicazione è stato condotto uno studio sperimentale, il quale, ha evidenziato come questa strategia di sviluppo creava confusione agli utenti. Nella modalità di calcolo di un percorso, l'esecuzione automatica di due widget non permetteva, infatti, una chiara distinzione tra la partenza e l'arrivo, creando così una bassa usabilità del sistema. Pertanto, per questo motivo, in sostituzione ai due widget sono state implementate due barre di ricerca che, al loro selezionamento, richiamano esclusivamente le funzionalità di predizione e auto-completamento del testo (e non l'intero widget), rendendo il sistema molto più comprensibile e usabile all'utente.

Infine, il metodo *onCreate()*, crea un oggetto di tipo *Spinner* per permettere all'utente di selezionare un tipo di mappa diverso da quello predefinito.

Il metodo *onMapReady()* è un metodo che viene attivato nel momento in cui la mappa viene recuperata dai server di Google e resa disponibile all'applicazione.

Esso si occupa di aggiungere alla mappa i pulsanti per lo zoom e la bussola, di disabilitare il funzionamento dell'app ufficiale Google StreetView in modo da poter richiamare l'activity proprietaria e non l'eventuale applicazione Google già installata, di controllare se i permessi concessi dall'utente consentono di geolocalizzarlo e, in caso affermativo, creare il pulsante apposito.

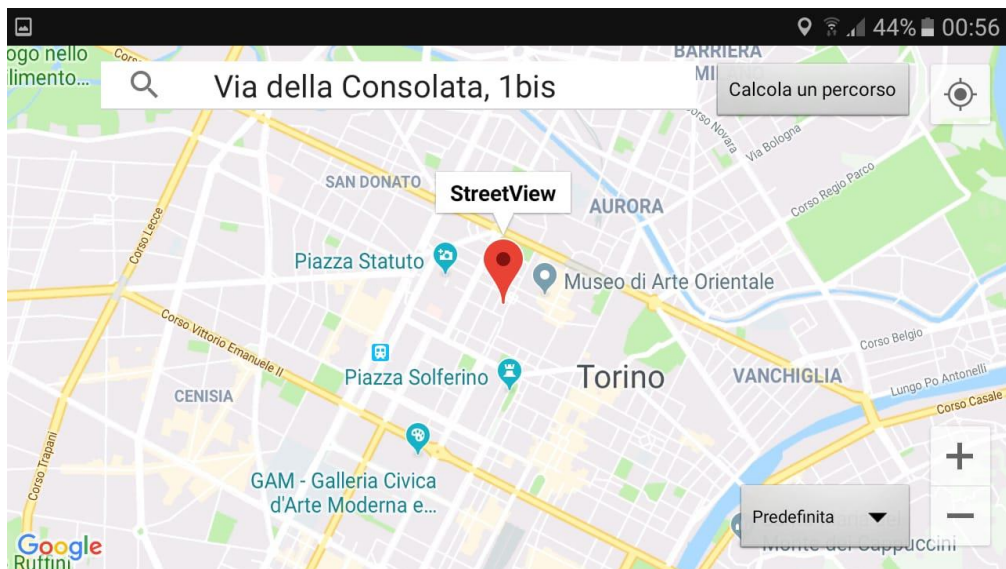


Figura 4.4 Modalità “cerca un luogo”

Nella figura sottostante viene mostrata la lista completa di tutti i metodi e delle classi interne presenti nella classe e che verranno di seguito presentati seguendo l'ordine in figura.

```
//method performed for creating activity
@Override
protected void onCreate(Bundle savedInstanceState) {...}

// method activated for searching a place
private void search_location() {...}

// method activated for calculating a route
private void search_route() {...}

@Override
public void onMapReady(GoogleMap googleMap) {...}

@Override
public void onMyLocationClick(@NonNull Location location) {...}

@Override
public boolean onMyLocationButtonClick() {...}

@Override
public void onMapClick(LatLng latLng) {...}

@Override
public boolean onMarkerClick(Marker marker) {...}

@Override
public void onInfoWindowClick(Marker marker) {...}

private String getDirectionsUrl(LatLng origin, LatLng dest) {...}

private String downloadUrl(String strUrl) throws IOException {...}

private class DownloadTask extends AsyncTask<String, Void, String> {...}

private class ParserTask extends AsyncTask<String, Integer, List<List<HashMap<String, String>>>> {...}

public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {...}

private void updateMapType() {...}
```

Figura 4.5 Metodi e classi interne di MapsActivity

Il metodo *search_location()* è un metodo chiamabile esclusivamente nella modalità di calcolo di un percorso. Viene eseguito quando l'utente preme sul pulsante “cerca un luogo” e la sua funzione è quella di cambiare modalità. Si occupa di eliminare possibili marker o possibili *textview* presenti per la visualizzazione dei dati relativi al percorso precedentemente calcolato e, inoltre, modifica il valore dell'attributo MODE ponendolo uguale a 1.

Simmetricamente, il metodo *search_route()* è un metodo chiamabile esclusivamente nella modalità di ricerca di un luogo. Viene eseguito quando l'utente preme sul pulsante “calcola un percorso” e la sua funzione è quella di cambiare modalità. Modifica il valore dell'attributo MODE ponendolo uguale a 2.

onMyLocationClick() è un metodo ereditato da una classe di Google; si occupa di gestire il click dell'utente sulla posizione corrente. Quando viene chiamato, richiama a sua volta il metodo *onMapClick()* passando come parametri la longitudine e la latitudine della posizione.

onMyLocationButtonClick() è un metodo ereditato da una classe di Google e si occupa della gestione del click dell'utente sul pulsante della geolocalizzazione. Il suo compito è semplicemente quello di notificare all'utente che sta avvenendo la ricerca della propria posizione.

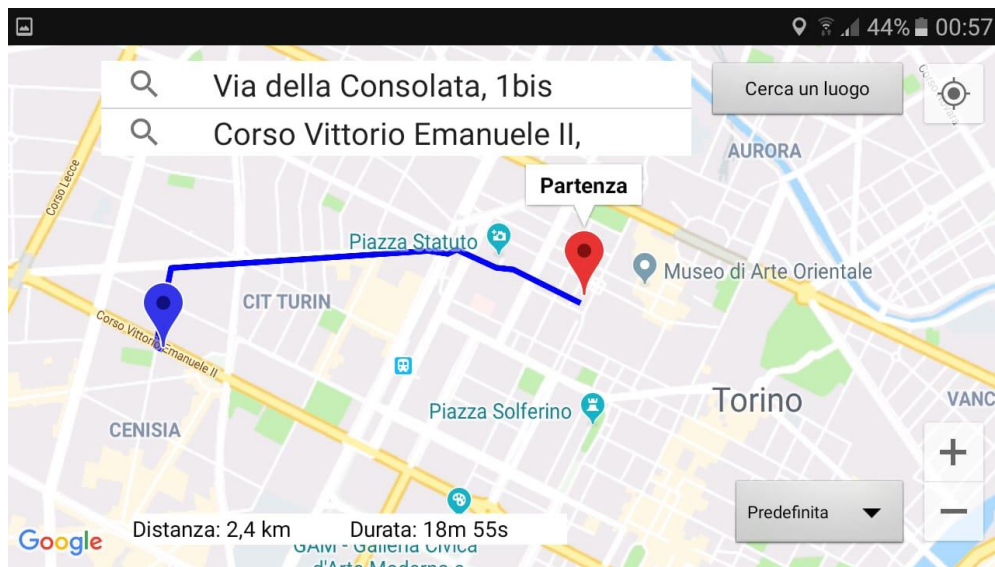


Figura 4.6 Modalità "calcola un percorso"

onMapClick() è un metodo ereditato da una classe di Google che si occupa dell'interazione tra l'utente e la mappa. È possibile richiamare questo metodo in due maniere differenti: tramite il click sulla mappa oppure tramite l'inserimento di testo nelle barre di ricerca.

È bene sottolineare che, nonostante questi due modi siano diversi tra loro, essi siano sincroni; difatti, al click sulla mappa all'utente sarà mostrato il luogo in forma testuale sulla barra di ricerca e viceversa.

Il suo funzionamento si articola in due casi distinti in base alla modalità corrente.

Nel caso della modalità di ricerca di un luogo si occupa di eliminare possibili marker presenti e di crearne uno sulla posizione cliccata (o ricercata).

Nel caso della modalità di calcolo di un percorso all'inizio si occupa di verificare la presenza di markers. Nel caso in cui siano due significa che precedentemente è già stato calcolato un percorso e quindi li rimuove e ne crea uno nuovo sulla posizione appena selezionata; nel caso in cui sia uno significa che precedentemente si è ricercato un luogo

e ora, molto probabilmente, si ha l'esigenza di calcolare un percorso da quel punto e, dunque, quel marker sarà il punto di partenza e l'arrivo sarà il punto appena selezionato.

Non appena il metodo ha a disposizione partenza e arrivo, richiama il metodo *getDirectionsUrl()* passandogli i due luoghi come parametri e crea l'istanza della classe *DownloadTask*.

onMarkerClick() è un metodo ereditato da una classe di Google che si occupa della rimozione del marker quando esso viene cliccato.

onInfoWindowClick() è un metodo ereditato da una classe di Google che permette il passaggio alla *StreetViewActivity*, alla quale vengono forniti i dati relativi alla modalità corrente e alla posizione del luogo. È possibile richiamarlo cliccando sul titolo di un marker.

getDirectionsUrl() è un metodo che viene richiamato da *onMapClick()*. Il suo compito è quello di comporre una URL con tutti i parametri necessari, quali partenza, arrivo, modalità di percorrenza (in questo caso, a piedi) e formato di risposta (JSON) per effettuare una richiesta al server di Google.

downloadUrl() è un metodo che viene richiamato all'interno della classe *DownloadTask*. Esso si occupa di ricevere la URL costruita dal metodo *getDirectionsUrl()*, di stabilire una connessione *http* per poter effettuare la richiesta del percorso al server di Google e di ricevere la risposta contenente i dati relativi al percorso richiesto.

DownloadTask è una classe che estende le funzionalità della classe astratta *AsyncTask*. Il suo funzionamento consiste nella chiamata di *downloadUrl()* e nella memorizzazione della risposta fornita da questo metodo. Successivamente crea l'istanza della classe *ParserTask*.

ParserTask è anch'essa una classe che estende le funzionalità della classe astratta *AsyncTask*. È la classe che permette (insieme alla classe ausiliaria *DirectionsJSONParser* che verrà illustrata nei prossimi paragrafi) di effettuare il parsing della risposta ricevuta in formato JSON e di memorizzarla in una struttura dati ben organizzata. Ulteriormente, si occupa anche di disegnare sulla mappa il percorso che collega i due punti e di far visualizzare all'utente i dati relativi a distanza e durata del tragitto. Siccome il percorso viene calcolato per una persona che si muove a piedi, Google fornisce una durata calcolata approssimativamente su una velocità di 4.8km/h; per questo motivo, questo valore viene normalizzato a 8km/h per rendere il dato il più veritiero possibile

(considerato che l'applicazione è utilizzata per fare corsa a bassa intensità) e la durata ricalcolata autonomamente da un metodo interno alla classe.

onItemSelected() è un metodo che gestisce lo spinner relativo al cambio di tipo di mappa. Si attiva ogni volta che l'utente cambia tipo di visualizzazione e il suo unico compito è quello di richiamare il metodo *updateMapType()*.

updateMapType() è il metodo che controlla quale tipo di mappa è stato selezionato dall'utente e imposta il tipo in base alla richiesta.

4.2 StreetViewActivity

StreetViewActivity è l'activity che viene eseguita in seguito al click sul titolo di un marker sulla mappa; essa permette l'esplorazione di un luogo o di un percorso tramite l'utilizzo del touchscreen oppure tramite un visore VR facendo uso del giroscopio e dell'accelerometro dello smartphone (se presenti).

Questa classe è stata sviluppata insieme a Roberto Lotterio, un altro studente del mio corso di studi. In particolare, il mio compito è stato quello di permettere la visualizzazione del panorama di un luogo o di un percorso nella modalità classica, la creazione di uno zoom personalizzato e l'interazione di questa activity con la precedente, che è stata interamente sviluppata in autonomia. Il suo compito invece è stato quello di mostrare il panorama di un luogo o di un percorso nella modalità VR. Nel corso di questo paragrafo verranno, pertanto, illustrati soprattutto i metodi da me sviluppati, ma verrà data ugualmente anche una breve descrizione del resto in modo da permettere al lettore di conoscere il funzionamento dell'intera applicazione.

4.2.1 Attributi

```
public class StreetViewActivity extends FragmentActivity
    implements OnStreetViewPanoramaReadyCallback {

    private static final float ZOOM_BY = 0.25f;

    private StreetViewPanorama mStreetViewPanorama;
    private LatLng latLng;

    private static final String TAG = StreetViewActivity.class.getSimpleName();
    private VrPanoramaView panoWidgetView;
    public boolean loadImageSuccessful;
    /** Tracks the file to be loaded across the lifetime of this app. */
    private Uri fileUri;
    /** Configuration information for the panorama. */
    private VrPanoramaView.Options panoOptions = new VrPanoramaView.Options();
    private ImageLoaderTask backgroundImageLoaderTask;
```

Figura 4.7 Attributi della classe StreetViewActivity

Tra questi attributi meritano un'attenzione particolare:

- **mStreetViewPanorama**, un oggetto utilizzato per la ricezione del panorama fornito da Google StreetView e che consente all'utente l'esplorazione dello stesso. Con l'utilizzo di questo oggetto il programmatore ha la possibilità di aggiungere o rimuovere dal panorama funzionalità quali zoom, rotazione, nomi delle vie o il movimento tra un panorama e un altro.
- **latLng**, un oggetto utilizzato per la memorizzazione di longitudine e latitudine di un luogo.

4.2.2 Metodi

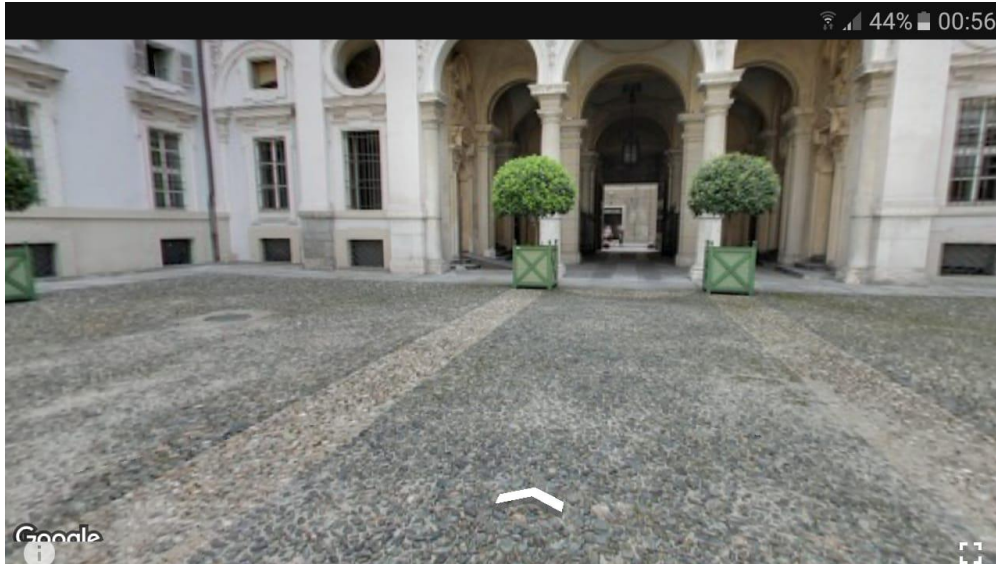


Figura 4.8 Screenshot StreetView

La figura 4.8 rappresenta la schermata che viene presentata all'utente all'avvio di questa activity.

Il suo layout è stato sviluppato utilizzando il linguaggio XML ed il codice è disponibile all'interno della directory *res/layout/activity_streetview.xml*.

La sua creazione, invece, avviene tramite i metodi *onCreate()* e *onStreetViewPanoramaReady()*.

```
@Override
protected void onCreate(final Bundle savedInstanceState) {...}

@Override
protected void onNewIntent(Intent intent) {...}

private void handleIntent(Intent intent) {...}

@Override
public void onStreetViewPanoramaReady(StreetViewPanorama streetViewPanorama) {...}

@Override
public boolean dispatchKeyEvent(KeyEvent event) {...}

private boolean checkReady() {...}

public void onZoomIn() {...}

public void onZoomOut() {...}

class ImageLoaderTask extends AsyncTask<Pair<Uri, VrPanoramaView.Options>, Void, Boolean> {...}

private class ActivityEventListener extends VrPanoramaEventListener {...}
```

Figura 4.9 Metodi e classi interne di StreetViewActivity

Il metodo *onCreate()* è il primo ad essere chiamato e, come è stato precedentemente spiegato, viene richiamato tramite un click sul titolo di un marker.

Da ciò, è facile intuire come le prime operazioni che esso svolge siano la ricezione e la memorizzazione delle coordinate relative al marker cliccato.

In generale, il metodo si occupa della inizializzazione di alcuni attributi della classe e dell'esecuzione dell'activity.

Il metodo *onStreetViewPanoramaReady()* è un metodo che viene attivato nel momento in cui l'applicazione è pronta a recuperare un panorama dai server di Google e renderlo disponibile all'applicazione per la sua visualizzazione.

Esso si occupa di controllare la presenza di un panorama per il luogo selezionato (o per i luoghi vicini ad esso) e, in caso di esito positivo, della sua visualizzazione.

Illustrati i primi due metodi che vengono eseguiti, vengono ora presentati tutti gli altri metodi e le altre classi procedendo seguendo l'ordine in *figura 4.9*.

onNewIntent() è un metodo utilizzato per il salvataggio dello stato dell'intent in modo che il sistema non ne crei uno nuovo ogni volta che l'activity viene eseguita. Esso richiama il metodo *handleIntent()*.

handleIntent() è un metodo utilizzato per la modalità VR e viene usato per diversi scopi quali: la determinazione della presenza di file da caricare necessari all'esecuzione in VR, l'impostazione di vari parametri per la modalità e la creazione dell'istanza della classe *ImageLoaderTask* necessaria allo scaricamento delle immagini dal server di Google StreetView.

dispatchKeyEvent() è un metodo che viene richiamato quando l'utente preme un qualsiasi pulsante dello smartphone. Nel dettaglio, alla pressione del tasto "indietro" viene interrotta l'activity e, se in corso, lo scaricamento di immagini e il controllo torna a *MapsActivity*; alla pressione del tasto "volume su" viene richiamato il metodo *onZoomIn()* e, simmetricamente, alla pressione del tasto "volume giù" viene richiamato il metodo *onZoomOut()*. La pressione di qualsiasi altro tasto non menzionato viene gestita dal sistema operativo.

checkReady() è un metodo utilizzato all'interno di *onZoomIn()* e *onZoomOut()* per controllare la presenza di un panorama e quindi autorizzare l'esecuzione dello zoom.

onZoomIn() è il metodo che permette di aumentare lo zoom su un panorama. La sua implementazione avviene servendosi di metodi predefiniti dell'attributo *mStreetViewPanorama*, in grado di effettuare lo zoom in maniera graduale.

onZoomOut() è il metodo che permette di diminuire lo zoom su un panorama. La sua implementazione avviene servendosi di metodi predefiniti dell'attributo *mStreetViewPanorama*, in grado di effettuare lo zoom in maniera graduale.

ImageLoaderTask è una classe che estende le funzionalità della classe astratta *AsyncTask* e viene utilizzata per la modalità VR. Essa riceve da *MapsActivity* la modalità di visualizzazione e la lista di punti che vi sono tra la partenza e l'arrivo.

Nel caso della modalità “cerca un luogo”, il metodo stabilisce una connessione con il server di Google StreetView e costruisce varie URL per lo scaricamento delle immagini di quel luogo. Dato che le API non consentono lo scaricamento di immagini a 360°, il metodo combina le varie immagini scaricate a 120° creando un panorama a 360°.

Il caso della modalità “calcola un percorso” è del tutto analogo ma, per la costruzione del percorso appunto, vengono scaricate in background tutte le immagini del tragitto e combinate panorama per panorama, mentre il *thread* principale continua a mostrare all'utente il percorso, caricando automaticamente panorama per panorama ogni 2 secondi.

ActivityEventListener è una classe ausiliaria che viene utilizzata nella modalità VR per il controllo del corretto funzionamento del sistema.



Figura 4.10 Modalità VR

4.3 DirectionsJSONParser

DirectionsJSONParser è una classe di appoggio utilizzata per il *parsing* di dati in formato *JSON*, ovvero, per l'analisi sintattica di un flusso continuo di dati in ingresso in modo da determinarne una struttura grazie ad una grammatica formale^[15].

Il formato JSON è un formato utilizzato per l'interscambio di dati fra applicazioni. Esso è spesso utilizzato in ambito di programmazione web con il linguaggio di programmazione JavaScript, il quale non è supportato dall'ambiente di sviluppo Android Studio. Per questo motivo non esistono funzioni predefinite in grado di eseguire il parsing, dunque, è stata necessaria la creazione di questa classe ausiliaria.

La classe viene utilizzata durante il calcolo di un percorso e fornisce longitudine e latitudine per una matrice di punti, in base al percorso creato tra il punto iniziale e quello finale.

Gli sviluppatori di Google hanno messo a disposizione degli altri sviluppatori il codice di questa classe, ma che si è rivelata utile solamente in modo parziale per le finalità di questo progetto data la mancanza di dati relativi a distanza e durata di percorrenza.

È stato indispensabile, perciò, lo studio approfondito del funzionamento della classe e del formato di risposta con il quale, il server di Google, fornisce i dati relativi a distanza e durata, in modo da poter comprendere come modificare la classe in linea con le esigenze.

Nella figura sottostante viene mostrato un esempio di risposta in formato JSON.

```
"routes" : [
  {
    "bounds" : {
      "northeast" : {
        "lat" : 45.5257529,
        "lng" : 9.2193209
      },
      "southwest" : {
        "lat" : 45.5138622,
        "lng" : 9.208023499999999
      }
    },
    "copyrights" : "Dati mappa ©2018 Google",
    "legs" : [
      {
        "distance" : {
          "text" : "1,3 mi",
          "value" : 2030
        },
        "duration" : {
          "text" : "6 min",
          "value" : 353
        },
        "end_address" : "Piazza della Scienza, 4, 20126 Milano MI, Italia",
        "end_location" : {
          "lat" : 45.5139321,
          "lng" : 9.210380899999999
        },
        "start_address" : "Edificio U14, Viale Sarca, 336, 20126 Sesto San Giovanni MI, Italia",
        "start_location" : {
          "lat" : 45.5235976,
          "lng" : 9.2193209
        }
      }
    ]
  }
]
```

Figura 4.11 Esempio di un percorso in formato JSON

Studio di Usabilità

Sull'applicazione appena presentata è stato svolto uno studio di Usabilità in collaborazione con il mio collega Roberto Lotterio.

L'intero studio è disponibile all'indirizzo web <https://drive.google.com/file/d/1t-sKzbzknzZNXcuUWjrRR-tud2-fpEpM>).

Il lavoro è stato suddiviso in tre sezioni:

- Valutazione euristica
- Test Utente
- Questionario

5.1 Valutazione euristica

Per la valutazione euristica sono stati coinvolti esperti di dominio, esperti di usabilità e tecnici informatici. Essendo un'applicazione ancora in fase di sviluppo, io e il mio collega, abbiamo cercato di coinvolgere principalmente un campione di persone a noi vicine, dunque, giovani con familiarità medio-alta di sistemi simili a quello proposto e che abbiano buona dimestichezza con i sistemi informatici.

ID	GENERE	PROFESSIONE	FASCIA DI ETÀ	FAMILIARITÀ CON APP	FAMILIARITÀ TECNOLOGIE INFORMATICHE
1	M	Esperto di Usabilità	15-30	Alta	Alta
2	M	Esperto di Usabilità	15-30	Alta	Alta
3	M	Studente di Informatica	15-30	Media	Alta
4	M	Studente di Informatica	15-30	Media	Alta
5	M	Studente di Informatica	15-30	Alta	Alta
6	M	Tecnico informatico	15-30	Alta	Alta

■ : Autori dello studio di usabilità.

Figura 5.1 Campione valutazione euristica

La valutazione euristica è stata svolta sulla prima versione del sistema.

Ai valutatori è stata proposta una prima fase olistica della durata di pochi minuti (disponibile all'indirizzo web https://docs.google.com/presentation/d/1HGq_83UjqSBmWWWh_LurMi0Zn6MCNMHPVgNFGOT3EOw); dopodiché, gli è stato chiesto di compiere delle operazioni generiche e un task specifico che permettessero una rapida esplorazione del sistema e, infine, la compilazione di un questionario relativo ai problemi trovati (disponibile all'indirizzo web <https://goo.gl/forms/NLwC3rd6rif4Seka2>).

Qui di seguito l'elenco dei problemi trovati, ordinato per priorità.

ID	Descrizione breve	Descrizione	Euristiche violate (ID) *	Popolarità	Valutatori (ID) **	Priorità ***	Severità		
							Media	Dev. Stand.	Mediana
7	Impossibilità calcolo percorso	Occasionalmente diventa impossibile calcolare un percorso	10	4	1-2-3-6	ALTA	3,66	0,51	4
4	Partenza-arrivo non distinguibili	Nella modalità "calcola percorso" non è chiara la distinzione tra partenza e arrivo	5	5	2-3-4-5-6	ALTA	3,5	0,83	4
5	Visualizzazione volume in zoom	Facendo lo zoom con i tasti del volume in StreetView viene modificato anche il livello del volume	6	3	1-3-4	MEDIA	3,5	0,54	3,5
2	Info non chiare	La visualizzazione del tempo e della distanza è poco chiara o non presente	2	3	1-2-6	MEDIA	3,16	0,4	3
6	Rimozione marker	Nella modalità "calcola percorso" vi è la possibilità di rimuovere i marker, lasciando il sistema in uno stato inconsistente	5	3	1-3-5	MEDIA	3	1,09	3
9	Crash app	Nella modalità "calcola percorso", inserendo solo la partenza e provando a entrare in modalità StreetView l'app va in crash	3	1	5	MEDIA	3	0,89	3
8	Visualizzazione arrivo in calcola percorso	Nella modalità "calcola percorso" viene visualizzato l'arrivo come se fosse la partenza	5-6	5	2-3-4-5-6	MEDIA	2,83	0,75	3
1	Lingua stile mappe diversa	La lingua dello stile delle mappe è diversa rispetto al resto dell'applicazione	2	2	1-2	BASSA	1,5	0,54	1,5
3	Tasto stile mappe ingombrante	Il tasto dello stile delle mappe è troppo grosso rispetto all'importanza che ricopre	6-8	2	1-2	BASSA	1,16	0,4	1

Figura 5.2 Elenco dei problemi

5.2 Test utente

Tutti i valutatori che hanno partecipato a questa fase dello studio di Usabilità hanno firmato una liberatoria per effettuare registrazioni audio/video. Hanno partecipato volontariamente e non hanno ricevuto alcun compenso.

Per il test utente è stato adottato un livello di confidenza del 95%. Sono state ritenute statisticamente significative le differenze associate a P-value inferiori al 5%.

Ai valutatori è stato sottoposto lo svolgimento di tre task su due versioni dell'applicazione:

- versione 1: prima versione prototipo;
- versione 2: evoluzione della versione 1 con miglioramenti sui problemi rilevati nella valutazione euristica.

Abbiamo considerato due campioni differenti per le due versioni, ognuno composto da 12 persone. Per ogni valutatore sono state svolte un totale di tre prove, una per ogni task.

Il primo campione ha eseguito i tre task sulla prima versione, mentre il secondo campione ha eseguito gli stessi tre task sulla seconda versione.

CAMPIONE 1:						CAMPIONE 2:					
ID	GENERE	PROFESSIONE	FASCIA DI ETÀ	FAMILIARITÀ CON APP	FAMILIARITÀ TECNOLOGIE INFORMATICHE	ID	GENERE	PROFESSIONE	FASCIA DI ETÀ	FAMILIARITÀ CON APP	FAMILIARITÀ TECNOLOGIE INFORMATICHE
1	M	Commercialista	50+	Media	Media	13	M	Pizzaiolo	15-30	Media	Alta
2	M	Studiante di informatica	15-30	Media	Alta	14	F	Impiegata	15-30	Media	Alta
3	F	Studiante universitario	30-50	Bassa	Media	15	F	Studiante universitario	15-30	Media	Media
4	M	Tecnico informatico	15-30	Alta	Alta	16	F	Impiegata	30-50	Bassa	Bassa
5	M	Dirigente	50+	Media	Media	17	M	Carrozziere	50+	Bassa	Bassa
6	M	Impiegato	30-50	Alta	Alta	18	M	Studiante di informatica	15-30	Alta	Alta
7	F	Impiegata	50+	Bassa	Bassa	19	F	Estetista	30-50	Media	Bassa
8	M	Studiante di informatica	15-30	Alta	Alta	20	M	Libero professionista	50+	Media	Media
9	M	Studiante di informatica	15-30	Media	Alta	21	F	Impiegata	50+	Bassa	Bassa
10	M	Studiante di informatica	15-30	Media	Alta	22	M	Impiegato	30-50	Media	Media
11	F	Impiegata	30-50	Media	Media	23	F	Studiante universitario	15-30	Media	Bassa
12	F	Commercialista	50+	Bassa	Bassa	24	M	Impiegato	15-30	Media	Media

Figura 5.3 Campione 1 TestUtente

Figura 5.4 Campione 2 TestUtente

Viene resa disponibile al lettore una presentazione di questi task all'indirizzo web https://docs.google.com/presentation/d/1gf8EXtfHoOejdkTJrm3_KLn7k9AJqYZ7ocFX3idciM.

Per misurare l'efficienza d'uso del sistema abbiamo rilevato i tempi di esecuzione di ogni task di ogni utente attraverso l'ausilio delle registrazioni effettuate.

Abbiamo poi rappresentato graficamente le distribuzioni dei tempi ottenuti sui singoli task e confrontato la performance delle due versioni tramite test statistici.

Sono stati eseguiti tre t-test per campioni indipendenti per confrontare i tempi di esecuzione dei tre task usando la versione 1 di VR World Explorer e la versione 2.

A seguito di questi test, è stato riscontrato un P-value superiore a 0,05 e, dunque, per nessuno di questi task si è trovata una differenza significativa per i tempi di esecuzione tra la versione 1 di VR World Explorer e la versione 2.

Questo risultato suggerisce che gli utenti non impiegano un tempo significativamente minore per eseguire i tre task usando la versione 1 di VR World Explorer rispetto a quando usano la versione 2.

Per quanto riguarda l'efficacia è stato valutato il tasso di errori rilevato da ciascun valutatore per ogni task di entrambe le versioni durante il test utente.

In questa fase abbiamo assegnato il colore verde ai task eseguiti correttamente, giallo a quelli terminati a seguito di suggerimenti o dopo alcuni errori lievi e rosso ai task non portati a termine in modo adeguato. Abbiamo poi deciso di valutare i task gialli come se fossero stati eseguiti in modo errato (rossi).

CAMPIONE 1:				CAMPIONE 2:			
ID	TASK 1	TASK 2	TASK 3	ID	TASK 1	TASK 2	TASK 3
1				13			
2				14			
3				15			
4				16			
5				17			
6				18			
7				19			
8				20			
9				21			
10				22			
11				23			
12				24			

Figura 5.5 Efficacia campione 1

Figura 5.6 Efficacia campione 2

Sono stati eseguiti tre test del Chi-quadro per confrontare il numero di successi nel compiere i tre task usando la versione 1 di VR World Explorer e la versione 2.

A seguito di questi test, è stato riscontrato un P-value superiore a 0,05 e, dunque, per nessuno di questi task si è trovata una differenza statisticamente significativa riguardo al numero di successi tra la versione 1 di VR World Explorer e la versione 2.

Questo risultato suggerisce che gli utenti non ottengono più successi nell'esecuzione dei tre task usando la versione 1 di VR World Explorer rispetto a quando usano la versione 2.

5.3 Questionario

In questa ultima fase di raccolta dati abbiamo sottoposto gli utenti ad un'altra analisi quantitativa realizzata attraverso la somministrazione di un questionario di User Experience.

L'ISO 9241 definisce la User Experience come le percezioni e le reazioni di un utente che derivano dall'uso o dall'aspettativa d'uso di un prodotto, sistema o servizio.

Questo studio è quindi necessario per la valutazione generale dell'applicazione in quanto, un'applicazione usabile, non sempre genera un'esperienza positiva nell'utente.

Il questionario è stato sottoposto a 24 persone in totale, le stesse che hanno eseguito anche i task.

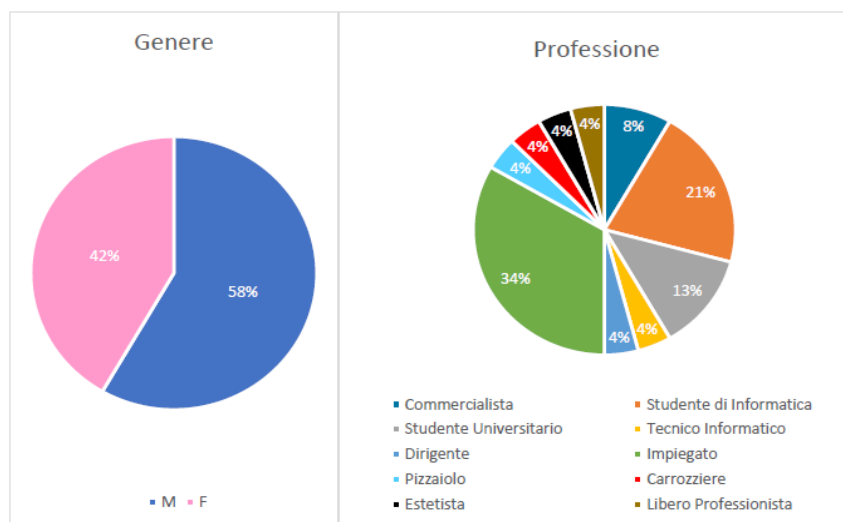


Figura 5.7 Statistiche valutatori questionario (1 di 2)

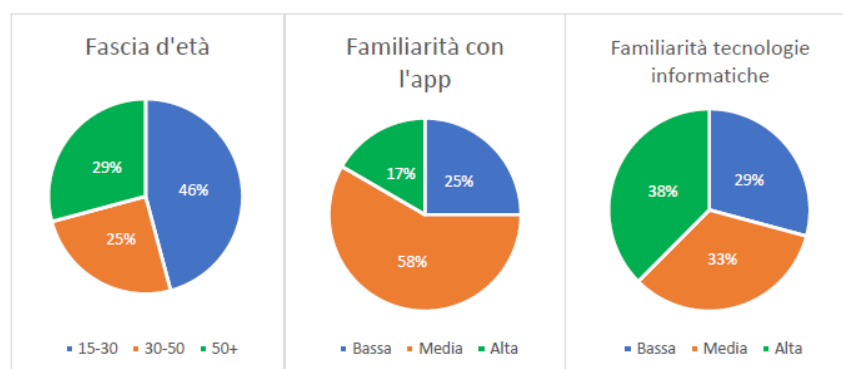


Figura 5.8 Statistiche valutatori questionario (2 di 2)

Per questa fase di test è stata considerata la seconda versione dell'applicazione, ovvero quella migliorata secondo i risultati della valutazione euristica.

Per poter procedere abbiamo dovuto scegliere uno tra i questionari di User Experience presenti in rete e abbiamo optato per lo UEQ (*User Experience Questionnaire*). È stato scelto perché più breve e semplice rispetto ad altri modelli, copre aree semantiche importanti, ha una traduzione italiana, possiede un'ottima documentazione e fornisce degli strumenti per l'analisi delle risposte raccolte.

I 24 rispondenti, prima di compilare il questionario (disponibile all'indirizzo web <https://goo.gl/forms/8lq6guPE07t0oN0e2>), hanno svolto una sessione di esplorazione del sistema della durata di qualche minuto e sono stati invitati a provare le varie funzionalità dell'applicativo.

I rispondenti hanno fornito un giudizio da 1 a 7 rispetto due qualità contrapposte.

Ogni risposta è stata convertita in un valore compreso fra -3 (molto negativo) e +3 (molto positivo) in modo da considerare il valore 0 come neutro.

Valutazione superiori a 0.8 sono positive, inferiori a -0.8 sono negative, fra -0.8 e 0.8 sono considerate neutre.

Il questionario non fornisce una valutazione complessiva dell'esperienza dell'utente.

La sua struttura, però, va a coprire aree semantiche di rilevanza; questo ci permette di avere un quadro significativo su quello che l'applicazione può trasmettere ad una persona durante il suo utilizzo.

Le scale di UEQ sono: attrattività, apprendibilità, efficienza, controllabilità, stimolazione e originalità.

Tutte queste, tranne l'attrattività, vengono raggruppate in:

- qualità pragmatiche (apprendibilità, efficienza, controllabilità);
- qualità edoniche (stimolazione e originalità).

	Attrattività	Apprendibilità	Efficienza	Controllabilità	Stimolazione	Originalità
Media	1,72	1,84	1,50	1,39	1,00	0,68

Figura 5.9 Risultati questionario nelle 6 scale

	Attrattività	Qualità pragmatiche	Qualità edoniche
Media	1,72	1,58	0,84

Figura 5.10 Risultati questionario in raggruppamenti

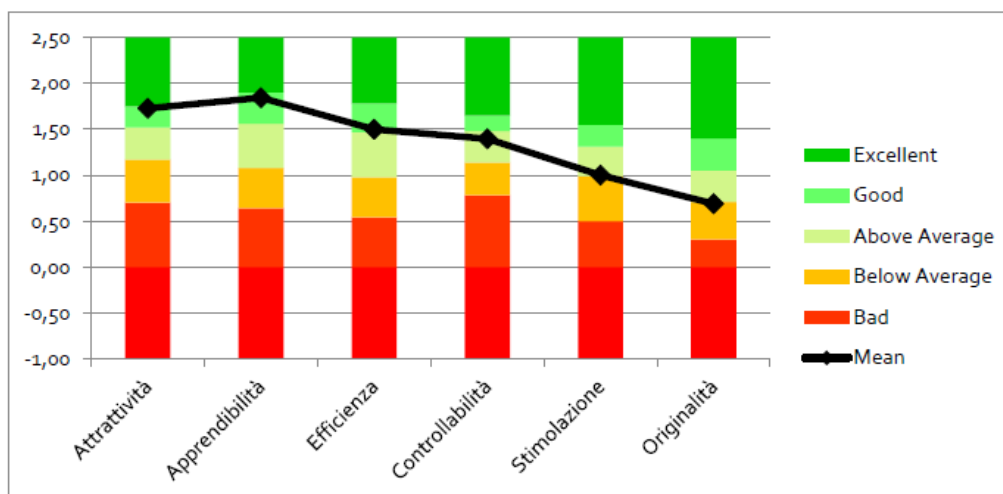


Figura 5.11 Grafico risultati questionario nelle 6 scale

In conclusione, gli utenti che hanno usato la versione 2, sviluppata tenendo conto dei risultati della valutazione euristica, si sono dimostrati soddisfatti.

Considerando che l'applicazione sarà utilizzata principalmente da un'utenza che ha necessità di essere seguita ed incentivata ad un'attività fisica è incoraggiante riscontrare dei risultati quasi sempre ben al di sopra della media, specialmente per quel che riguarda l'attrattività e l'apprendibilità.

Conclusioni e prospettive future

L'applicazione proposta, in seguito a vari miglioramenti, potrebbe risultare una valida alternativa a quelle già presenti sul mercato. Integrare la realtà virtuale all'interno di questo contesto si conferma essere un valido strumento per la stimolazione degli utenti all'attività fisica che, però, da solo non può essere considerato come unico risolutore del problema: è più sensato considerarlo come un supporto al superamento di alcune problematiche degli utenti.

In futuro, il lavoro si potrà concentrare su molti aspetti, tra i quali sicuramente:

- la creazione di percorsi predefiniti con diversi tempi di percorrenza e con diversi paesaggi, in modo da poter permettere l'uso dell'applicazione anche da strumenti come tapis roulant o cyclette.
- l'introduzione di contenuti musicali per una maggiore stimolazione dell'utente
- l'aggiunta di una modalità che permette di selezionare un luogo di partenza e una distanza da percorrere e mostrare all'utente i percorsi disponibili.

Postfazione

Lavorare a questa tesi è stata un'esperienza diversa da come me l'aspettassi: mi ha introdotto ai mondi della realtà virtuale e dello sviluppo di applicazioni e mi ha permesso di toccare con mano questi ambiti dell'informatica.

Per quanto il mio lavoro sia stato quanto più incentrato sull'interazione dell'utente con le mappe e con i percorsi nella modalità classica di visualizzazione, mi ha comunque permesso di capire meglio quali e quanti benefici possa apportare il mondo della realtà virtuale. Senz'altro ha arricchito il mio bagaglio di conoscenze concedendomi la possibilità di sperimentare un modo di lavorare che non avevo mai affrontato prima e di imparare ad utilizzare software e strumenti così importanti per questo campo. Penso che la tematica trattata sia estremamente interessante, e spero che il lavoro da me svolto possa essere continuato e migliorato da altri stagisti di questo corso.

Non posso che sperare in progressi nel ramo della realtà virtuale e che i progressi fatti serviranno a migliorare effettivamente la vita di tutti noi, anche se solo in una minuscola parte.

Bibliografia

- [1] *Piattaforma*. URL: [https://it.wikipedia.org/wiki/Piattaforma_\(informatica\)](https://it.wikipedia.org/wiki/Piattaforma_(informatica))
- [2] *Diffusione Android*. URL: https://www.ilsoftware.it/articoli.asp?tag=Smartphone-piu-diffusi-e-sistemi-operativi-mobili-piu-usati-i-dati-Gartner-aggiornati_16951
- [3] *App*. URL: https://it.wikipedia.org/wiki/Applicazione_mobile
- [4] *Activity*. URL: https://www.mrwebmaster.it/android/ciclo-vita-applicazioni-activity_12219.html
- [5] *Ciclo di vita di un'Activity*. URL: https://www.mrwebmaster.it/android/ciclo-vita-applicazioni-activity_12219.html
- [6] *Intent*. URL: https://www.mrwebmaster.it/android/gestione-activity-tramite-intent_12220.html
- [7] *Intent*. URL: <http://www.html.it/pag/19500/le-azioni-intent/>
- [8] *Android Studio*. URL: <https://developer.android.com/sdk/index.html>
- [9] *Gradle*. URL: <https://it.wikipedia.org/wiki/Gradle>
- [10] *GitHub*. URL: <https://github.com/>
- [11] *SDK*. URL: <https://www.fastweb.it/web-e-digital/cosa-e-sdk/>
- [12] *API*. URL: <https://www.fastweb.it/web-e-digital/cosa-sono-le-api-e-a-cosa-servono/>
- [13] *API*. URL: https://it.wikipedia.org/wiki/Application_programming_interface
- [14] *Places SDK*. URL: <https://developers.google.com/places/android-sdk/intro>
- [15] *Parsing*. URL: <https://it.wikipedia.org/wiki/Parsing>