FACOLTÀ DI INGEGNERIA

Corso di Laurea in Ingegneria Informatica e dell'Automazione

APPercorsi: un'applicazione mobile per percorsi escursionistici del Parco Nazionale del Gran Sasso e Monti della Laga, con funzionalità di ricerca di contenuti multimediali geolocalizzati

APPercorsi: a mobile application for hiking trails of the National Park of Gran Sasso and the Laga Mountains, with geo-localized multimedia content search functionality

Relatore:
Prof. **EMANUELE STORTI**

Tesi di laurea di:
FEDERICA RIPANI

Anno Accademico 2020/2021

Indice

CAPITOLO 1

Introduzione

I progressi tecnologici dell'ultimo decennio hanno permesso una rapida diffusione dei dispositivi mobili tanto da determinarne la loro adozione da parte della quasi totalità dei consumatori. I progressi hanno riguardato in special modo la progressiva diminuzione delle dimensioni di tali dispositivi e l'evoluzione delle tecnologie wireless che hanno permesso agli utenti di accedere alla rete in qualsiasi luogo, in modo semplice e veloce. Per questo motivo, tali dispositivi, vengono utilizzati come strumenti per la visualizzazione di qualsiasi tipo di dato, o informazione, e per usufruire di funzionalità in grado di soddisfare le necessità degli utenti. L'obiettivo di questa tesi è quello di migliorare l'esperienza di coloro che vogliano conoscere ed esplorare il *Parco Nazionale del Gran Sasso e Monti della Laga*, ciò è stato possibile attraverso lo sviluppo di un'applicazione Android in grado di guidarli sia nella scelta del percorso migliore, anche attraverso tecniche di ricerca di elementi multimediali geolocalizzati studiate ad-hoc, che durante tutta la loro permanenza nel sentiero. Avere questo tipo di strumenti quando ci si trova in montagna è quasi indispensabile: aiutano ad orientarsi, capire dove ci si trova e muoversi correttamente verso la giusta direzione. Nell'antichità l'orientamento era basato sull'osservazione, le mappe erano solamente mentali o, al più, incise su roccia e legno. L'avvento della tecnologia ha fatto sì che fosse possibile utilizzare strumenti più all'avanguardia come la bussola, il sestante e l'orologio, tecnologie che hanno permesso all'uomo di conoscere la forma della terra e di misurarla con meticolosità. Il rapido sviluppo tecnologico ha poi portato l'uomo dall'utilizzo di strumenti ormai obsoleti, come la cartina o la bussola, verso l'utilizzo di meto-

di di posizionamento molto più sofisticati ed indubbiamente più comodi. Questi metodi, di più recente sviluppo, permettono di compiere azioni che sarebbero risultate impensabili nel passato. Ad esempio, grazie a servizi come Google Maps, è possibile pianificare un percorso da un punto di partenza ad un punto di arrivo, persino da remoto, in modo veloce ed agevole. Nella quotidianità è diventato ormai indispensabile l'utilizzo di servizi utili al raggiungimento di destinazioni, alla pianificazione di percorsi e alla ricerca di un particolare punto di interesse.

Figura 1.1: Strumenti di navigazione su diversi dispositivi.

I principali fornitori di questo tipo di servizi sono grandi compagnie quali Google ed Apple. Esse riescono a fornire un'esperienza di alto livello all'utente grazie a diversi anni di studi durante i quali è stata raccolta una grande mole di dati. Inoltre, Google fornisce degli strumenti utili per gli sviluppatori. Il più famoso strumento condiviso da Google è Android, il principale sistema operativo per dispositivi mobili. Vengono pubblicate sempre più applicazioni che fanno uso di queste tecnologie e che risultano essere interessanti per l'uso quotidiano. Questo lavoro di tesi rigarda proprio lo sviluppo di un'applicazione Android e con il presente elaborato si vogliono descrivere tutti gli strumenti utilizzati, le fasi di progettazione, sviluppo e test dell'applicazione, con particolare enfasi sulle tecniche utilizzate per i vari tipi di ricerca di elementi multimediali geolocalizzati che sono stati implementati nell'app. Nello specifico, i capitoli sono stati strutturati come spiegato di seguito:

- *Capitolo* 2: si chiarirà l'obiettivo della tesi e verranno analizzati i requisiti dell'applicazione, sia quelli funzionali che non;

- *Capitolo* 3: verranno introdotti tutti gli strumenti utilizzati per la realizzazione dell'applicazione, in particolare si partirà dall'ambiente di sviluppo Android Studio per poi passare a Firebase, lo spazio di memorizzazione e archiviazione dei dati. Verrà poi spiegato come utilizzare la libreria OSMdroid ed infine dove sono stati presi i dati da inserire nell'applicazione, cioè Wikiloc;

- *Capitolo* 4: inizialmente verrà introdotta la struttura tipica di un'applicazione mobile, successivamente si analizzeranno tutte le fasi necessarie per lo sviluppo di APPercorsi, a partire dalla progettazione fino all'implementazione;

- *Capitolo* 5: si descriveranno sia gli algoritmi progettati e realizzati per la funzionalità di ricerca presente in APPercorsi che i test effettuati per verificare l'effettivo funzionamento di tali algoritmi. Nella sezione finale del capitolo si analizzeranno i risultati ottenuti;

- *Capitolo* 6: dopo una riesamina dei requisiti e degli obiettivi prefissati ed una valutazione degli strumenti utilizzati si trarranno le conclusioni e si proporranno possibili sviluppi futuri.

CAPITOLO 2

Analisi dei requisiti

In questo capitolo verrà affrontata l'analisi dei requisiti necessari per la progettazione e l'implementazione dell'applicazione che si vuole realizzare. Si partirà dalla descrizione dell'obiettivo di tale applicativo, per poi elencare i requisiti funzionali e non.

2.1 | Obiettivo

Nell'area protetta del Parco Nazionale del Gran Sasso e dei Monti Della Laga sono presenti moltissimi sentieri, nei quali da millenni convivono meravigliosi tesori della natura e un rilevante patrimonio culturale [1]. Infatti, lungo i sentieri, si ha la fortuna di incontrare sia testimonianze storiche, come borghi antichi, siti archeologici, castelli, eremi e grotte, sia tesori della natura, essendo ricchi di sorgenti, cascate, praterie, altopiani, vertiginose creste e impressionanti pareti rocciose, motivo per il quale meritano di essere valorizzati. Purtroppo però, non sono tutti ben segnalati e, lungo i percorsi, non sempre si riesce a capire a che punto ci si trova o dove si trovano i punti di interesse di quel sentiero. Nasce dunque l'idea di creare un'applicazione Android, APPercorsi, in grado di seguire gli utenti lungo tutta loro escursione e di segnalare loro i luoghi caratteristici presenti sul percorso. Attraverso questo particolare applicativo si cercherà di regalare all'utente uno strumento efficiente, utile e semplice per poter visitare e godere delle bellezze del Parco Nazionale del Gran Sasso e dei Monti Della Laga.

2.1.1 Parco Nazionale del Gran Sasso e Monti della Laga

Il Parco rappresenta un vero e proprio polmone verde per il centro Italia e si estende sul terittorio di ben tre Regioni differenti: Abruzzo, Lazio e Marche. Infatti, nell'EUAP (Elenco Ufficiale delle Aree naturali Protette) risulta essere la terza Area Naturale Protetta più grande d'Italia per estensione, vantando di una superficie di quasi 150mila ettari. É proprio per la sua vasta estensione che, nel 1995, divenne Parco Nazionale Del Gran Sasso e dei Monti Della Laga, unendo i due massicci montuosi omonimi: il Gran Sasso e i Monti della Laga. Negli anni '60/'70 questo territorio sembrava essere destinato alla realizzazione di zone residenziali e complessi sciistici che avrebbero deturpato gran parte del valore naturalistico, ma grazie alle diverse associazioni ambientaliste presenti nel Parco ciò fu impedito. Oggi sembra essere stata una fortuna, in quanto, grazie al Parco si sta realizzando un turismo meno invasivo e più orientato alla valorizzazione delle ricchezze culturali, storiche e naturalistiche. Negli anni, infatti, sono stati realizzati moltissimi sentieri segnalati che includono itinerari di ogni genere, passeggiate panoramiche per i meno esperti, itinerari tematici per chiunque voglia esplorare il patrimonio culturale della zona ed escursioni più impegnative per gli amanti dell'avventura.

2.2 | Requisiti

Un requisito è detto funzionale se descrive cosa deve e cosa non deve fare un sistema software e come deve reagire agli stimoli esterni, mentre i requisiti non funzionali sono le proprietà del sistema che devono essere soddisfatte, quindi pongono vincoli su come si comporterà il sistema. In questa sezione verranno analizzati i requisiti che caratterizzeranno l'applicazione.

2.2.1 Requisiti funzionali

L'utente potrà utilizzare le seguenti funzionalità:

- *Creazione* di un account;
- *Ricerca* di elementi multimediali geolocalizzati tra tutti i percorsi:

  - Ricerca di immagini;
  - Ricerca di registrazioni audio;
  - Ricerca di testi;
- *Visualizzazione* di un percorso da seguire:
  - Visualizzazione delle informazioni descrittive del percorso;
  - Visualizzazione del sentiero da seguire ed i relativi elementi multimediali;
  - Visualizzazione dello zaino consigliato;
- *Ricerca* di elementi multimediali geolocalizzati all'interno di un percorso:
  - Ricerca di immagini;
  - Ricerca di registrazioni audio;
  - Ricerca di testi.

2.2.2 Requisiti non funzionali

- *Mobile*: APPercorsi sarà un'applicazione destinata ai dispositivi mobili;
- *Design* intuitivo e accattivante: l'applicativo dovrà essere facile da utilizzare anche per gli utenti meno esperti;
- *Velocità*: l'app dovrà essere fluida e veloce nell'esecuzione;
- *Password*: l'utente dovrà scegliere una password di almeno 8 caratteri;
- *App nativa*: APPercorsi sarà un'applicazione nativa;
- *Android*: l'applicativo sarà sviluppato per il sistema operativo Android;
- *Kotlin*: il linguaggio che verrà utilizzato per programmare l'applicazione è Kotlin;
- *Database*: tutti i dati saranno salvati all'interno di un database.

Mobile

Le applicazioni mobile sono dei programmi software progettati e sviluppati per funzionare su appositi dispositivi mobili quali smartphone, tablet, smartwatch e simili. Tali dispositivi sono caratterizzati da memorie poco capienti e bassa capacità di elaborazione e processamento, inoltre, il loro funzionamento è vincolato sui consumi energetici a causa della limitata energia fornita dalla batteria. Dal momento che tali dispositivi sono stati pensati per seguire la mobilità dell'utente, risultano essere di piccole dimensioni, manegevoli e quindi di più facile utilizzo rispetto ai tradizionali PC. Infatti, come suggerito nell'articolo [2], adesso la maggior parte del traffico web deriva da dispositivi mobile e tenderà a crescere ancora, nasce quindi la necessità di spostare una grande quantità di servizi sul web mobile. Per questo motivo, oggi, esistono applicazioni mobile in grado di soddisfare la maggior parte delle necessità degli utenti. In particolare, l'applicativo sarà sviluppato per dispositivi mobili perchè pensato per l'utilizzo in movimento.

Android

L'applicativo si rivolge agli utenti che possiedono dispositivi mobili con sistema operativo Android. I due sistemi operativi che attualmente dominano sul mercato sono Android ed iOS. Il primo è di proprietà di Google ed è il più diffuso nel mondo mobile rappresentando una fetta di mercato superiore al 62%, mentre il secondo è sviluppato da Apple in esclusiva per i propri dispositivi mobile. Android fu lanciato per la prima volta nel 2008, è basato su kernel Linux e segue il modello di sviluppo open source, quindi offre all'utente una vasta libertà di utilizzo ed una grande flessibilità declinata soprattutto nei termini di progettazione e rilascio di software ad esso dedicati [3]. Tale sistema operativo si è evoluto rapidamente, fino all'ultima versione rilasciata: Android 11.

Tipi di app

Di seguito sono descritte tutte le tipologie di applicazioni.

1. **Native**: vengono sviluppate specificamente per un sistema operativo;

2. **Web**: funzionano come un sito web ma senza la necessità di scaricare l'applicazione sul dispositivo;

3. **Ibride**: sono dette anche multipiattaforma, combina le caratteristiche delle due tipologie di app precendenti.

Come già detto APPercorsi è un'app nativa, sviluppare questo tipo di applicazioni porta diversi vantaggi quali velocità, affidabilità, maggiore reattività e risoluzione superiore che assicurano una *user experience* migliore. Inoltre permettono l'accesso all'hardware e al software installato nel device come per esempio il file system, i sensori ed i servizi.

Kotlin

Le App Native vengono sviluppate con un linguaggio di programmazione differente da un sistema operativo all'altro: iOS utilizza in gran parte il linguaggio Objective-C mentre Android usa Java e Kotlin [4]. Kotlin è un moderno linguaggio di programmazione sviluppato da JetBrains nel 2011 ed è stato adattato ed integrato nell'ambiente di sviluppo Android Studio già nel 2017 [5]. I due linguaggi sono molto affini e Kotlin è interoperabile al 100% con Java ma, essendo più giovane, ha una community molto più piccola. Dal 2019 Kotlin risulta essere il linguaggio di programmazione consigliato da Google [6] per lo sviluppo di applicazioni mobile, questo perchè è un linguaggio più user-friendly, per esempio non è necessario esplicitare il tipo di una variabile e il compilatore non ha sempre bisogno di un cast esplicito. Kotlin quindi, richiede meno codice da scrivere rispetto al Java ma questo non risulta essere a spese della sicurezza, infatti, è stato introdotto il `Null-Safety`. Nell'ambito della programmazione mobile è stato una vera e propria rivoluzione poichè il `NullPointerException`, o `NPE`, risultava essere uno dei principali problemi con Java, causando solitamente il crash dell'app. L'`NPE` è una RuntimeException e si genera quando si cerca di accedere ad un oggetto che ha un riferimento a `null`. Kotlin evita i verificarsi di tale eccezione permettendo di distinguere tra due tipi di variabili:

- Nullable references che accettano valori nulli;

- Non-null references che non possono contenere riferimenti nulli.

ciò garantisce un accesso sicuro alle variabili segnalando eventuali errori a compile-time. L'utilizzo di questo nuovo e conciso linguaggio di programmazione mira ad alleggerire il lavoro degli sviluppatori, i quali si troverebbero a realizzare il software attraverso l'impiego di una quantità inferiore di codice e con meno complicazioni durante il processo di sviluppo.

CAPITOLO 3

Strumenti utilizzati

In questo capitolo verranno analizzati gli strumenti e le tecnologie che sono state utilizate per la realizzazione di APPercorsi. Nello specifico:

- *Android Studio* come IDE per la programmazione;
- *Firebase* per la memorizzazione e gestione dei dati;
- *OSMdroid* per la realizzazione e gestione delle mappe e dei percorsi;
- *Wikiloc* come sorgente dei dati, quali punti di interesse e tracciati.

3.1 | Android Studio

Esistono degli strumenti e dei particolari software in grado di supportare gli sviluppatori durante la fase di programmazione dei propri applicativi. In primis vengono utilizzati degli ambienti di sviluppo integrati, detti anche IDE, che supportano i programmatori durante la fase di debugging del codice, spesso segnalando errori di sintassi in fase di scrittura. La maggior parte degli IDE presenta i seguenti tool integrati:

- un editor di testo (text editor), con cui scrivere e modificare il codice;
- un evidenziatore di sintassi (syntax highlighter), che consente di presentare i diversi elementi sintattici del codice (ad esempio proprietà, tag e attributi) in colori differenti per facilitare la scrittura e il controllo del codice;
- degli strumenti di automazione della build, per velocizzare le attività semplici o ripetitive nella fase di creazione di una build del software;

- un debugger, per testare il codice e identificare i bug;
- un compilatore (compiler), che traduce il codice sorgente in codice oggetto;
- un interprete (interpreter), che esegue direttamente il codice sorgente senza bisogno di tradurlo.

Oggi uno degli IDE più utilizzati dagli sviluppatori è Android Studio. È un software gratuito su licenza Apache 2.0 ed il download è disponibile per la maggior parte dei sistemi operativi in circolazione, come Windows, Linux e macOS. Con la sua uscita Google ha voluto sostituire il precedente Eclipse mettendo a disposizione degli sviluppatori un ambiente di sviluppo sensibilmente migliorato sia dal punto di vista delle funzionalità che dell'efficienza nel sviluppare gli applicativi. inoltre il sistema garantisce una maggiore facilità di utilizzo, grazie ad una interfaccia più pulita e ottimizzata anche per gli utenti meno esperti. Le caratteristiche che rendono l'editor di Android Studio semplice, innovativo ed intuitivo, sono:

- la tab dei documenti;
- la barra di stato;
- l'area testuale;
- la barra laterale di validazione;
- strumenti per la cattura di prestazioni, usabilità, compatibilità di versione e altri problemi;
- funzionalità ProGuard e app-signing;
- Supporto per Google Cloud Platform e semplicit'a di integrazione con Google Cloud Messaging e App Engine.

Un altro elemento fondamentale presente negli IDE è l'emulatore. Esso permette di simulare i dispositivi Android direttamente sul computer in modo da poter testare l'applicazione su molteplici terminali e livelli API Android senza dover disporre di ogni dispositivo fisico. Testare l'app nell'emulatore spesso risulta essere più

veloce e semplice rispetto a farlo su un dispositivo fisico. Ad esempio, è possibile trasferire i dati più velocemente rispetto a un dispositivo collegato tramite USB.

3.2 | Firebase

Quando si sviluppa un'applicazione c'è la necessità di memorizzare i dati e archiviare i file per accedervi da qualsiasi luogo e in qualsiasi dispositivo. Nel caso di APPercorsi è stato scelto Firebase per soddisfare tale necessità. Firebase è un servizio open source offerto da Google per semplificare lo sviluppo di applicazioni mobili e web. Sfrutta l'infrastruttura di Google e il suo cloud per fornire una suite di strumenti per scrivere, analizzare e gestire applicazioni cross-platform. Consente agli sviluppatori di concentrarsi solo sul Front-end, permettendo di evitare tutta la gestione della parte di Back-end, aiutandoli inoltre nella gesione del server, delle chiavi API e nell'archiviazione dei dati. Il tutto è molto intuitivo così da facilitarne l'utilizzo ai meno esperti. Di seguito solo descritti alcuni dei servizi di maggior importanza offerti da Firebase, che sono quelli utilizzati per APPercorsi.

3.2.1 RealtimeDatabase

È un database con strutture NoSQL. La maggior parte dei database, solitamente, richiede di effettuare chiamate HTTP per ottenere e sincronizzare i dati. Non è il caso di Firebase, collegando l'applicazione a tale servizio ci si connette tramite WebSocket, il che rende ogni azione molto più veloce.

3.2.2 Storage

In Firebase possono essere archiviate diverse tipologie di file: immagini, documenti di testo e file in formato JSON. È inoltre disponibile un sistema di regole di sicurezza nel quale è possibile gestire i privilegi di scrittura e lettura al fine di proteggere tale archivio.

3.2.3 Autenticazione

Offre diversi sistemi di autenticazione integrati come Google, Twitter, Facebook e molti altri, ma solitamente si utilizza quello basato su Email e Password, come

nel caso dell'applicazione sviluppata.

3.3 | OSMdroid

Osmdroid [7] è una libreria che consente la visualizzazione delle mappe on-line sia di OpenStreetMap [8], una mappa del mondo creata dagli utenti stessi ed utilizzabile grazie alla licenza open source, sia di altri provider. Inoltre, offre agli utenti la possibilità di interagire tramite azioni di panning e zooming e permette agli sviluppatori la gestione e la sovrapposizione di più overlay, i quali possono identificare luoghi di interesse e tracciati. Una volta creato il proprio progetto Android, per poter utilizzare tale libreria e tutte le sue funzionalità sono necessarie le azioni preliminari [10] elencate di seguito:

- aggiungere le seguenti **dipendenze** sull file *build.gradle (Module:app)*:

```
dependencies {
    implementation 'org.osmdroid:osmdroid-android:<VERSION>'
    implementation 'com.github.MKergall:osmbonuspack:<VERSION
   >'
    }

```

Listing 3.1: Dipendenze nel file build.gradle (Module:app).

- impostare come *<VERSION>* l'ultima rilasciata, nel caso di APPercorsi è stata utilizzata la 6.1.11 per la prima dipendenza e la 6.7.0 per la seconda, e sincronizzare;

- settare le seguenti **autorizzazioni** sul file AndroidManifest.xml:

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

Listing 3.2: Permessi necessari.

- inserire la mappa all'interno del **layout** desiderato come mostrato nel codice di seguito:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/
    res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <org.osmdroid.views.MapView android:id="@+id/map"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent" />
</LinearLayout>
```

Listing 3.3: Layout con la mappa.

Una volta effettuate tali operazioni è possibile utilizzare tutte le funzionalità che tale libreria offre, come per esempio per gestire la mappa o i Marker, che verranno utilizzati per identificare i punti di interesse.

3.4 | Sorgente dei dati

Come sorgente dei dati è stato scelto Wikiloc poichè vanta una community molto ampia, che supera i 9 milioni di utenti, i quali condividono una grande mole di percorsi outdoor, immagini e molto altro. Ciò permette di ottenere qualsiasi informazione si necessiti. Più in generale, Wikiloc è uno spazio dove è possibile scoprire e condividere i migliori percorsi all'aria aperta per ciclismo, escursionismo e molte altre attività [9]. Nello specifico, è stato utilizzato per:

1. Consultare i dettagli e le specifiche dei percorsi da inserire nell'applicazione;
2. Estrapolare i punti geolocalizzati utili a comporre i tracciati;
3. Estrapolare tutte le informazioni relative ai punti di interesse da visualizzare in ogni percorso;
4. Ottenere delle immagini.

Per i punti 2 e 3 è stato utile scaricare dal sito di Wikiloc il file in formato GPX relativo ad ogni percorso e successivamente estrapolare la latitudine e la longitudine sia dei punti che compongono i percorsi sia dei punti di interesse.

CAPITOLO 4

APPercorsi

4.1 | Struttura di un'applicazione

Un'app è un inseme di una o più schermate che interagiscono tra loro. Ogni applicazione Android, indipendentemente dalla sua finalità, affida le sue funzionalità ai seguenti componenti.

- **Activity**: sono le interfacce utente, costituiscono il flusso in cui l'utente si inoltra per sfruttare le funzionalità messe a disposizione. Le activity possono contenere Views per mostrare dei dati sullo schermo o dei Fragments (sono delle Activity più piccole);
- **Service**: svolge il ruolo opposto delle Activity, infatti è un lavoro che viene svolto completamente in background, senza l'interazione con l'utente;
- **Content Provider**: ha lo scopo di condividere i dati tra le applicazioni, è utile per consentire alle app di accedere a tali dati senza dover utilizzare elementi terzi come i database;
- **BroadcastReceiver**: reagisce all'invio di messaggi di sistema, ad esempio l'arrivo di una chiamata o di un SMS. Solitamente non utilizzano l'interfaccia grafica sebbene possano inoltrare notifiche alla barra di stato per avvisare l'utente;
- **Resources**: l'insieme dei contenuti statici o di cui un'app ha bisogno per funzionare e contengono per esempio immagini, stringhe da visualizzare nelle interfacce, risorse grafiche e file di layout;

- **AndroidManifest**: è un file in formato XML che raccoglie informazioni necessarie per il corretto funzionamento dell'app. Fornisce un nome al package Java dell'applicazione (che è un identificativo univoco della stessa), descrive le componenti dell'app, dichiara i permessi, le librerie e il livello API minimo necessario per interagirci;

- **Intents**: sono un meccanismo per permettere alle Activity di comunicare tra loro.

Quando viene avviata un'applicazione Android crea il proprio processo Linux. Oltre a questo, il sistema crea un thread di esecuzione per l'applicazione chiamato thread principale o thread dell'interfaccia utente, anche detto thread del gestore. Quest'ultimo è responsabile della gestione degli eventi di tutta l'app ed, eventualmente, anche quelli di altre app. Esso è di fondamentale importanza perchè si occupa della gestione delle callback che scandiscono il ciclo di vita dei vari componenti, di disegnare l'interfaccia grafica e di rispondere all'interazione dell'utente. In particolare è l'unico responsabile dell'aggiornamento dell'interfaccia: per questo motivo viene solitamente chiamato UI Thread.

4.2 | Progettazione

Affinchè un'applicazione sia di successo essa deve garantire una buona *user experience*, ciò vuol dire sviluppare un'app semplice da utilizzare, con un'interfaccia intuitiva ed un design accattivante, ma che sia anche fluida e reattiva, affinchè l'utente non perda l'interesse durante l'utilizzo della stessa. Per poter garantire tutto ciò è fondamentale la fase di progettazione dell'app. In questo capitolo analizzeremo le componenti principali di un'applicativo e come è stata progettata APPercorsi.

4.2.1 Attori e casi d'uso

Prendendo in considerazione quanto si è detto nell'analisi dei requisiti, discussa nel capitolo precedente, è stato possibile delineare la struttura logica dell'applicazione attraverso la definizione del flusso di funzionalità a disposizione dell'utente. A tal

fine è stato utile costruire i diagrammi dei casi d'uso con i relativi attori, i quali permettono di descrivere in maniera chiara e non ambigua i "modi" in cui il sistema può essere utilizzato, cioè le funzionalità che il sistema mette a disposizione dei suoi utilizzatori. Gli attori sono coloro che eseguono i casi d'uso. Nello specifico, gli attori che operano nell'applicativo sviluppato sono i seguenti:

- *Nuovo utente*, colui che non ha mai effettuato un accesso;
- *Utente curioso*, una volta autenticatosi nell'app può visualizzare la lista dei percorsi e tutte le informazioni disponibili su di essi, inoltre può cercare dei contenuti multimediali geolocalizzati;
- *Utente escursionista*, una volta autenticatosi nell'app può visualizzare la lista dei percorsi e tutte le informazioni disponibili su di essi, inoltre può cercare dei contenuti multimediali geolocalizzati, seguire il percorso mediante la mappa e visionare lo zaino consigliato;
- *Amministratore*, è l'attore che gestisce l'applicativo ed in particolare tutto ciò che riguarda il database.

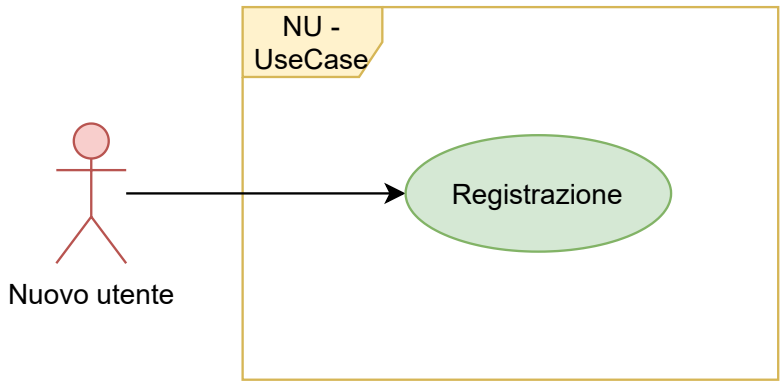


Figura 4.1: Use case diagram di un nuovo utente.

4.2.2 Mockup

Successivamente alle fasi appena descritte sono stati realizzati i mockup dell'applicazione. Quest'ultimi sono di grande importanza, in quanto, l'interfaccia grafica è l'unica parte dell'applicazione che sarà visibile dall'utente finale. Di seguito sono mostrati i 9 mockup realizzati durante la progettazione dell'app:

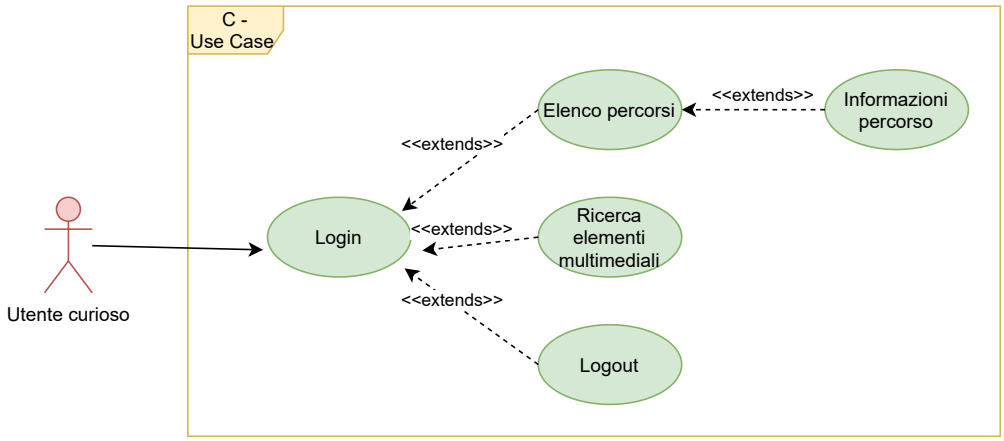


Figura 4.2: Use case diagram di un utente curioso.

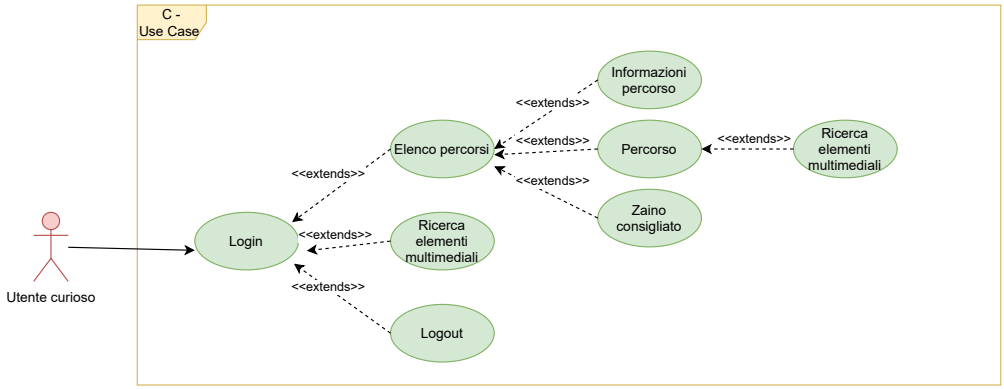


Figura 4.3: Use case diagram di un utente escursionista.

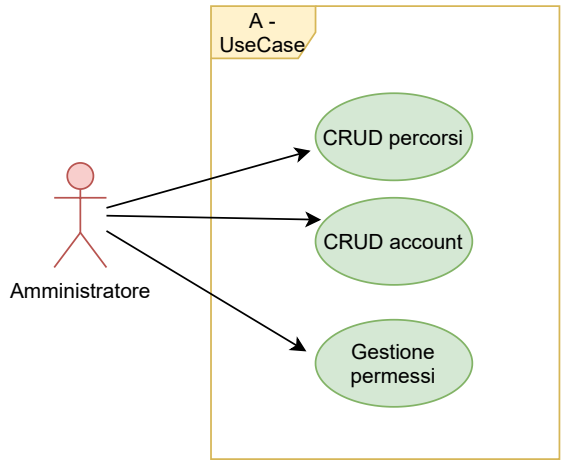


Figura 4.4: Use case diagram dell'admin.

- First Activity, la prima schermata che viene visualizzata quando si avvia l'app, rappresentata in Figura 4.5;

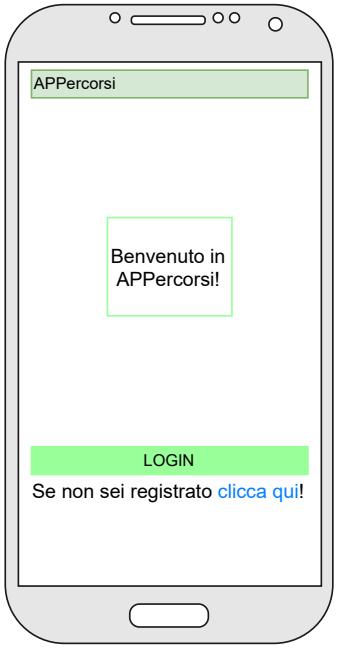


Figura 4.5: First Activity.

- Activity che permette la registrazione dei nuovi utenti in Figura 4.6 (a);
- Login Activity per gli utenti che hanno già un account in Figura 4.6 (b);
- Main Activity, o Activity Principale dell'applicazione, da qui in base a quale dei 4 bottoni verdi visionabili in Figura 4.7 viene scelto, è possibile accedere ad una delle funzionalità principali di APPercorsi. Inoltre, in Figura 4.7 è rappresentato anche il bottone *Aggiungi percorso* ma non verrà trattata la sua funzionalità in quanto non è stato oggetto di questa tesi bensì del lavoro di tesi dal titolo *APPercorsi: un'applicazione mobile per la fruizione e creazione di percorsi escursionistici del "Parco Nazionale del Gran Sasso e dei Monti della Laga"* di Camilla D'Andrea, una studentessa del Corso di Laurea in Ingegneria Informatica e dell'Automazione dell'Università Politecnica delle Marche;
- Activity contenente l'elenco dei percorsi in Figura 4.8;

(a) **(b)**

Figura 4.6: Autenticazione e Registrazione

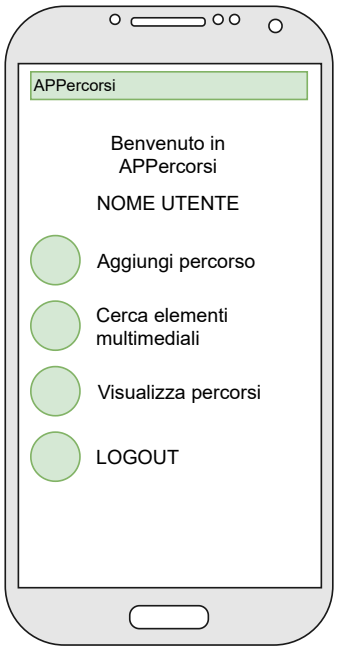


Figura 4.7: Main Activity.

- 3 Fragment relativi ai dettagli, la mappa e lo zaino consigliato di un percorso in Figura 4.9;
- Activity per la ricerca di contenuti multimediali in Figura 4.10;

Figura 4.8: Elenco percorsi.

(a) **(b)** **(c)**

Figura 4.9: Fragment dettagli, mappa e zaino.

4.2.3 Database

L'ultima fase della progettazione ha riguardato il database, in particolare sono stati scelti gli attributi che avrebbero descritto al meglio i percorsi da inserire

Figura 4.10: Ricerca contenuti multimediali.

nell'applicazione. Come mostrato in Figura 4.11 ogni percorso avrà 8 attributi. L'ultimo dei quali, lo *zaino*, sarà composto da una lista di 8 elementi. Più nel dettaglio, i seguenti campi sono utili per mostrare all'utente le specifiche tecniche e descrittive del percorso:

- *descrizione*;
- *difficoltà*;
- *lunghezza*;
- *nome*.

Nel caso specifico dello *zaino*, invece, è stato scelto di mettere a disposizione dell'utente una lista di equipaggiamenti consigliati che potrebbero risultare utili durante l'escursione, in modo da offrirgli l'esperienza migliore possibile.

4.3 | Implementazione

Nel capitolo precedente sono stati introdotti gli strumenti necessari ma non è stato illustrato come sono stati utilizzati per implementare l'app. In questa sezione verrà chiarito come è stata sviluppata l'applicazione grazie a tali strumenti.

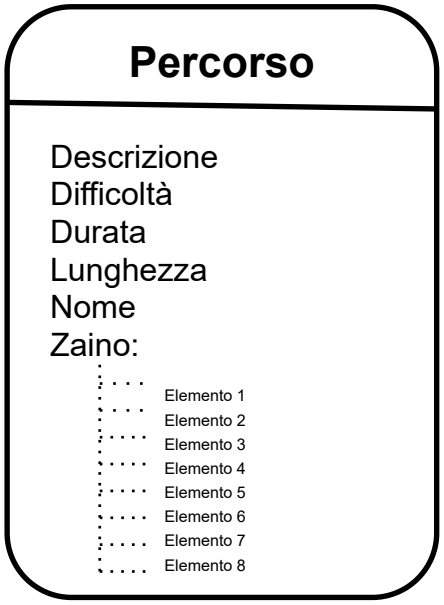


Figura 4.11: Attributi di percorso.

4.3.1 Struttura progetto Android Studio

La struttura del progetto su Android Studio è visionabile nella Figura 4.12. Il progetto presenta una cartella denominata *Manifests* contenente l'omonimo file in formato xml, la cartella *Java* che presenta tutte le classi e la *Res* che racchiude tutta la componente grafica dell'app dai file di layout alle stringhe.

Figura 4.12: Struttura del progetto.

4.3.2 Activity

Come illustrato nel capitolo precedente le Activity sono l'unica componente dell'applicazione con cui interagisce l'utente quindi sono state realizzate per essere il più possibile intuitive. Per esempio, in ognuna di esse è stata data la possibilità di tornare all'Activity precedente in modo semplice ed agevole grazie al metodo `onSupportNavigateUp`. Per implementare tutte le funzionalità presenti in APPercorsi sono state sviluppate 7 Activity. Di seguito verrano descritte le caratteristiche specifiche di ciascuna.

First Activity

È la prima activity che viene lanciata nel momento in cui l'utente apre l'applicazione. Come mostrato in Figura 4.13 qui l'utente oltre a visualizzare il logo di APPercorsi può effettuare il login per accedere a tutte le funzionalità dell'applicazione o, nel caso non lo avesse già fatto, può decidere di registrarsi.

Figura 4.13: First Activity.

Activity di registrazione

Qualsiasi utente che voglia usufruire dell'app deve prima registrarsi. Come mostrato in Figura 4.14 per farlo ha bisogna di inserire:

- *indirizzo di posta elettronica*, rappresenta l'identificativo univoco di ciascun utente. Infatti affinchè la registrazione termini con successo è necessario che nel database non sia presente un altro utente con lo stesso indirizzo;
- *password*, è la chiave di accesso al proprio account e viene crittografata prima di essere salvata sul dataabase.

Se un utente già registrato dovesse esser finito in questa Activity, grazie alla *Text View* cliccabile *Sei già registrato? Clicca qui!*, ha la possibilità di essere reindirizzato direttamente all'Activity corretta per effettuare il Login.

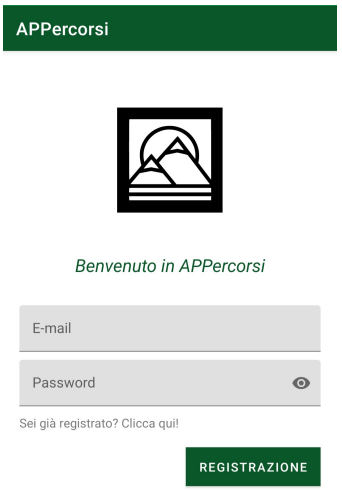


Figura 4.14: Registrazione.

Login Activity

Una volta che l'utente si è registrato con successo, ogni qualvolta voglia utilizzare l'app potrà autenticarsi inserirendo l'indirizzo di posta elettronica e la password inseriti in fase di registrazione. Se l'utente non avesse un account e dovesse essere finito in questa Activity per errore, grazie alla *Text View* cliccabile *Vuoi creare un nuovo account? Clicca qui!* viene reindirizzato all'Activity di registrazione. È possibile visionare un esempio in Figura 4.15.

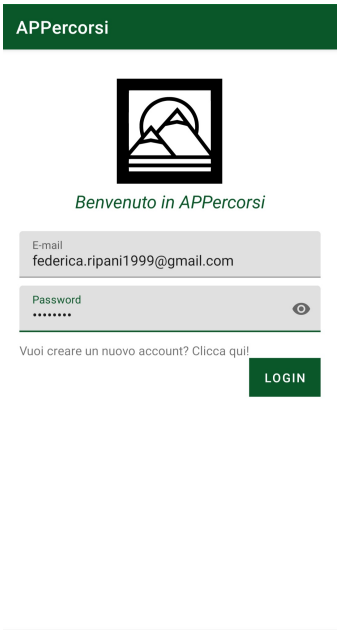


Figura 4.15: Login Activity.

Main Activity

Come suggerito dal nome è l'activity principale dell'applicativo. Come mostrato in Figura 4.13 presenta 4 Floating Action Button, detti FAB, ed in base a quale viene cliccato l'utente viene reindirizzato ad una funzionalità differente.

- *Aggiungi percorso*, permette di aggiungere un nuovo percorso ma come illustrato in precedenza non è stato oggetto di questa tesi;
- *Cerca contenuti multimediali*, viene data la possibilità di cercare un contenuto multimediale specifico tra tutti quelli presenti nella collezione di percorsi di cui dispone APPercorsi. I meccanismi di ricerca saranno discussi nel capitolo successivo;
- *Visualizza percorsi*, mostra la lista di tutti percorsi;
- *Logout*, l'utente effettua il logout e viene reindirizzato nella Login Activity.

Activity con l'elenco dei percorsi

In questa activity viene visualizzato l'elenco dei percorsi. Per implementarla è stata realizzata una RecyclerView [11] che solitamente viene utilizzata quando si

Figura 4.16: Main Activity.

ha un elenco a scorrimento. Il programmatore decide i dati che saranno visualizzati per ogni elemento della RecyclerView e che aspetto avrà ognuno di essi. Come suggerisce il nome, la RecyclerView ricicla i singoli elementi, quando uno di essi esce fuori dallo schermo non ne distrugge la visualizzazione bensì lo riutilizza per ospitare il nuovo elemento che entrerà nello schermo. Adottare questo tipo di soluzione porta diversi vantaggi sia per la reattività dell'app che per il consumo energetico. Un altro aspetto interessante della ReyclerView è la sua modularità che lo rende facilmente modificabile e riutilizzabile in più parti nel codice, infatti per realizzarla è necessaria la collaborazione di più componenti:

- *DataSource* è l'insieme dei dati che popolano la RecyclerView e solitamente sono sotto forma di una lista;
- *ViewHolder* fornisce il layout di base da popolare con i dati presenti nel DataSource;
- *Layout Manager* è responsabile del posizionamento delle view all'interno della RecyclerView;

- *Adapter* estrae i dati dal DataSource per popolare il ViewHolder ed inviarli al Layout Manager. Quando viene definito estende la classe RecyclerView.Adapter e deve sovrascrivere tre funzioni:
  - *getItemCount* per ottenere il numero totale di item presenti nella lista;
  - *onCreateViewHolder* crea un layout per gli items;
  - *onBindViewHolder* per riciclare il layout dell'item aggiornando i dati mostrati all'interno.

Come è possbile notare dalla Figura 4.17 per ogni percorso viene mostrata un'immagine, il nome, la durata e la lunghezza. Inoltre, per rendere cliccabili gli elementi della RecyclerView è stato inserito un *Listener* nel metodo *onBindViewHolder* presente nell'*Adapter*. Nello specifico è stato inserito il `setOnClickListener`, grazie al quale, nel momento in cui l'utente clicca su un elemento dell'elenco viene reindirizzato nell'Activity che mostra i dettagli del percorso ad esso relativo.

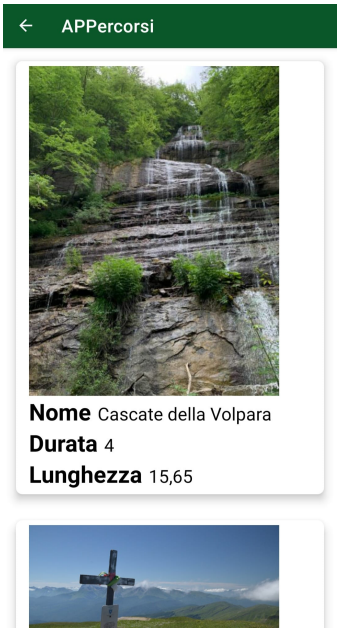


Figura 4.17: RecyclerView con i percorsi.

Activity dettagli del percorso

Come già discusso nella sezione della progettazione, si è deciso di rendere disponibili i dettagli, la mappa e lo zaino consigliato per ogni percorso. Quindi nella

fase di implementazione è stato scelto di utilizzare un *ViewPager* con tre differenti fragment in modo da rendere l'*Activity* più intuitiva per l'utente.

1. *Fragment Dettagli*: ospita le seguenti informazioni specifiche per ogni percorso:

   - Nome;
   - Difficoltà;
   - Durata espressa in ore;
   - Lunghezza espressa in chilometri;
   - Descrizione dettagliata del percorso;

   Poichè la descrizione solitamente è molto estesa, nel fragment è stato necessario inserire una *ScrollView*;

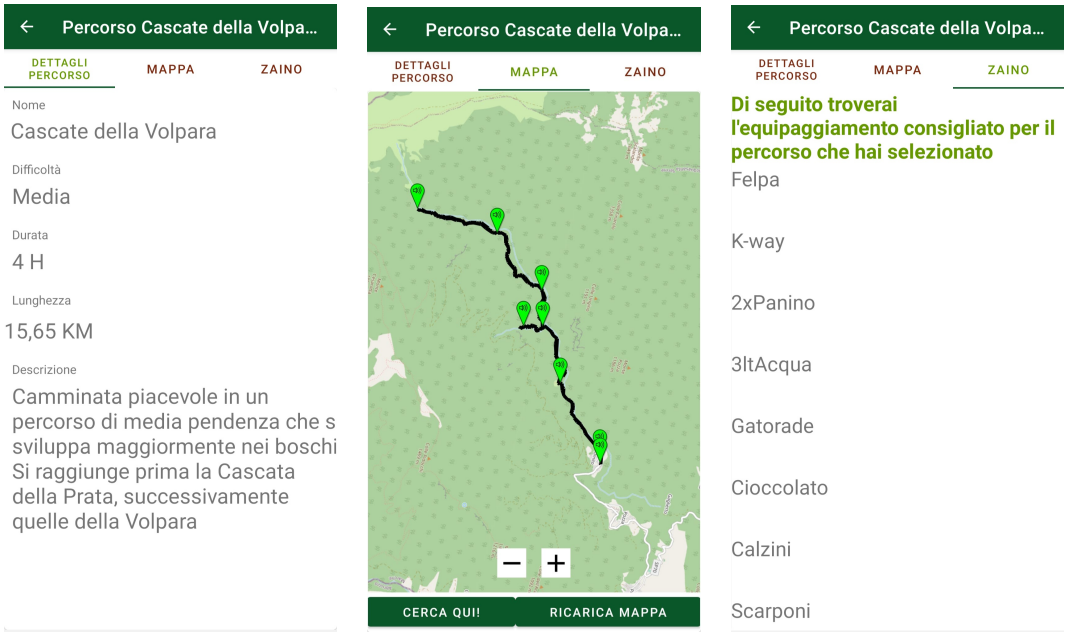
2. *Fragment Mappa*: è il fragment contenente la mappa con il percorso ed i sui punti di interesse, detti POI, ed un *BottomSheet* mostrato in FIgura 4.19 per la ricerca di audio, testi ed immagini tra i punti di interesse. Per gi algoritmi di ricerca si faccia riferimento al capitolo successivo;

3. *Fragment Zaino*: è una lista di 8 equipaggiamenti che si consiglia di portare con sè durante l'escursione.

Il risultato finale di tale *View Pager* è mostrato in Figura 4.18.

Activity di Ricerca dei Contenuti Multimediali

Grazie a questa Activity qualsiasi utente sia interessato ad un POI in particolare può cercarlo direttamente qui. La ricerca è estesa ai punti di interesse di ogni percorso e di qualunque tipologia quindi sia immagini, sia audio che testi. Come si nota dalla Figura 4.20, l'utente può cercare la parola desiderata tramite una *Search View* e qualora la parola in input soddisfi i criteri i ricerca verrà mostrata una *RecyclerView* contenente tutti i multimedia trovati, in caso contrario comparirà un *Toast* con scritto: *Nessun multimedia trovato!*. I criteri di ricerca utilizzati saranno illustrati nel capitolo successivo.



(a) **(b)** **(c)**

Figura 4.18: View Pager ed i relativi Fragment.

Figura 4.19: BottomSheet per la ricerca di contenuti multimediali nel percorso.

4.3.3 Database

Dopo aver scelto i percorsi da rendere fruibili sull'applicazione, sono stati raccolti i dati a loro relativi su a Wikiloc e sono stati utilizzati per popolare il dataset su Firebase. In Realtime Database sono stati inseriti tutti i percorsi e le loro

Figura 4.20: Ricerca di elementi multimediali.

informazioni. La struttura realizzata è come quella mostrata in Figura 4.21, dove ogni percorso contiene gli attributi già elencati nella progettazione del database e mostrati in Figura 4.11. Per ogni percorso sono stati archiviati diversi file sullo

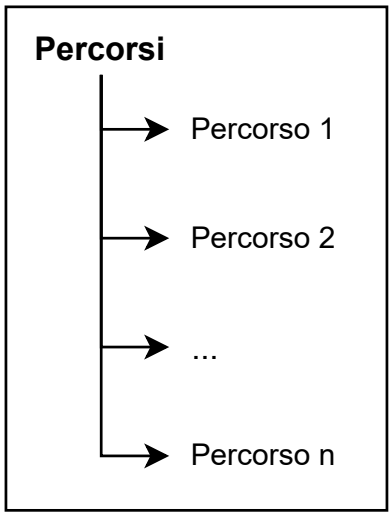


Figura 4.21: Struttura Realtime Database.

Storage di Firebase:

- una foto in formato JPG identificativa di ogni percorso;

- una foto in formato JPG per ogni elemento multimediale che dovesse contenere un'immagine;
- una registrazione per ogni elemento multimediale che dovesse contenere un audio;
- un file di testo contenente tutti i punti necessari per comporre il tracciato del percorso;

Inoltre, sempre sullo Storage, è stato salvato un file denominato POI, necessario a tutti i percorsi. Tale file è in formato JSON e contiene tutti i dati relativi agli elementi multimediali di ciascun percorso presente su Realtime Database.

JSON

Acronimo di JavaScript Object Notation è un formato molto utilizzato in quanto risulta essere semplice da leggere e scrivere per l'uomo e facile da analizzare e generare per le macchine. Solitamente i dati presenti sui file in formato JSON sono sotto forma di coppia chiave-valore.

CAPITOLO 5

Algoritmi di ricerca

Come già anticipato nei capitoli precedenti è possibile effettuare la ricerca dei contenuti multimediali in due Activity differenti:

- Activity di Ricerca dei Contenuti Multimediali;
- Activity dettagli del percorso, più in particolare nel Fragment della mappa;

In entrambe vengono utilizzati gli stessi criteri di ricerca ma con l'unica differenza che nel secondo caso, prima di procedere alla ricerca, si effettua un filtraggio dei contenuti multimediali, in particolare la ricerca avviene solo su quelli presenti nel percorso in cui ci si trova. Nel momento in cui l'utente inserisce una parola, la ricerca viene effettuata tra tutti i multimedia, quindi sia immagini che audio che testi. Nello specifico la parola verrà cercata nel campo *tag* e nel campo *descrizione* di ogni elemento multimediale. Per permettere tutto ciò sono stati sviluppati due diversi algoritmi, uno studiato ad-hoc per APPercorsi e l'altro basato sul concetto della Levenshtein distance. In una fase successiva sono stati testati entrambi gli algoritmi per valutarne i vantaggi e gli svantaggi di ciascuno.

5.1 | Algoritmo Longer Prefix Similarity

Il primo algoritmo è stato progettato appositamente per permettere agli utenti di cercare una parola che fosse simile, e quindi non necessariamente identica, ad una presente nel dataset. L'algoritmo effettua le seguenti operazioni:

- Prende in input la parola inserita dall'utente e quella da confrontare;

- Calcola il prefisso più lungo comune ad entrambe le parole;
- Calcola la percentuale di lunghezza di tale prefisso rispetto a quella delle due parole in input.

Se tale percentuale è maggiore dell'80% il confronto tra le due parole verrà considerato come un buon match.

5.2 | Algoritmo Levenshtein Distance

Il secondo algoritmo è basato sul concetto della *Levenshtein distance* [12], che è uno dei metodi più utilizzati per verificare la somiglianza tra due stringhe. Tale metodo calcola l'*edit distance*, cioè il minimo spostamento di caratteri necessario per rendere due stringhe identiche. Gli spostamenti possono essere sia sostituzioni, sia inserimenti che cancellazioni. Solitamente però l'utilizzo di questa tecnica richiede un elevato tempo di esecuzione [13], tuttavia nel caso della ricerca di singole parole, o piccoli testi, non si riscontra questo tipo di problema quindi è stato possibile realizzare l'algoritmo utilizzando il metodo della *Levenshtein distance* riadattandolo però al caso in esame. In particolare, utilizzando tale algoritmo per confrontare la stringa inserita dall'utente con i tag e la descrizione di ciascun elemento multimediale, il risultato prodotto sarà una *Levenshtein distance* compresa tra 0 e 1. Il calcolo di tale distanza utilizzando il metodo originale avrebbe avuto i seguenti limiti:

- come *limite inferiore* la differenza delle lunghezze delle due stringhe;
- come *limite superiore* la lunghezza della stringa più lunga.

Quindi la scelta della soglia per determinare il successo o meno del match tra le due stringhe messe a confronto sarebbe dipesa dalla lunghezza delle stesse, perciò per rendere il più generale possibili sia l'algoritmo di ricerca che la soglia, tale tecnica è stata riadattata nel seguente modo:

- prende in input la stringa inserita dall'utente e quella da confrontare;
- calcola le lunghezze delle due stringhe;

- assegna alla variabile `longerLength` la dimensione della stringa più lunga;
- assegna alla variabile `shorterLength` la lunghezza della stringa più corta;
- se il calcolo della `longerLength` da come risultato 0, la *Levenshtein distance* viene considerata pari a 1, cioè il massimo e l'algoritmo termina;
- altrimenti calcola l'*edit distance* tra la stringa più corta e quella più lunga tramite il metodo `editDistance(longer, shorter)`;
- calcola la *Levenshtein distance* con la seguente formula:

```
(longerLength - editDistance(longer, shorter)) / (shorterLength)
```

Se tale valore è maggiore di 0.8 il confronto tra le due stringhe verrà considerato come un buon match.

5.3 | Test e risultati

Per verificare il corretto funzionamento degli algorimi sono stati realizzati tre test. Per ognuno è stata simulata la ricerca di 11 parole da parte di un utente che utilizza l'app e successivamente è stata realizzata una tabella contente i risultati ottenuti, in ciascuna di esse sono state inserite le seguenti informazioni:

- VP sono i veri positivi, cioè i match riconosciuti;
- VN rappresentano i veri negativi, cioè non-match scartati correttamente;
- FP esprimono i falsi positivi, cioè i non-match scambiati per match erratamente;
- FN sono i falsi negativi, cioè i match scartati erratamente;
- Precision rappresenta il numero di elementi recuperati correttamente durante la ricerca rispetto al totale degli elementi recuperati ed è stata calcolata tramite la formula:

```
VP / (VP + FP)
```

Test 1	VP	VN	FP	FN	Precision	Recall
Acqua	7	19	0	0	100%	100%
Cascata	4	22	0	0	100%	100%
Corno	1	25	0	0	100%	100%
Indicazioni	2	24	0	0	100%	100%
Mon	4	20	2	0	67%	100%
Monte	3	22	1	0	75%	100%
Parch	1	25	0	0	100%	100%
Punto	1	25	0	0	100%	100%
Segnalazione	2	24	0	0	100%	100%
Seguire	1	24	1	0	50%	100%
Vista	2	24	0	0	100%	100%
TOTALE	27	254	4	0	87.1%	100%

Tabella 5.1: Risultati ottenuti con il Test 1. Si verificano 4 falsi positivi.

- Recall rappresenta il numero di elementi recuperati durante la ricerca rispetto al numero totale di elementi da recuperare ed è stata calcolata tramite la formula:

```
VP / (VP + FN)
```

5.3.1 Test 1

Nel primo test è stato utilizzato l'Algoritmo Longer Prefix Similarity per effettuare la ricerca della parola inserita dall'utente nei tag e l'Algoritmo Levenshtein Distance per effettuarla nelle descrizioni. I risultati ottenuti sono quelli mostrati in Tabella.

5.3.2 Test 2

Per questo test è stato utilizzato l'Algoritmo Longer Prefix SImilarity sia per la ricerca nei tag che nelle descrizioni. I risultati ottenuti sono quelli mostrati in Tabella 5.2.

5.3.3 Test 3

Infine è stato testato l'Algoritmo Levenshtein Distance per effettuare la ricerca della parola inserita dall'utente sia nei tag che nelle descrizioni. I risultati ottenuti

Test 2	VP	VN	FP	FN	Precision	Recall
Acqua	7	19	0	0	100%	100%
Cascata	4	22	0	0	100%	100%
Corno	1	25	0	0	100%	100%
Indicazioni	2	24	0	0	100%	100%
Mon	4	22	0	0	100%	100%
Monte	3	23	0	0	100%	100%
Parch	1	24	1	0	50%	100%
Punto	1	25	0	0	100%	100%
Segnalazione	2	24	0	0	100%	100%
Seguire	1	25	0	0	100%	100%
Vista	2	24	0	0	100%	100%
TOTALE	28	257	1	0	96.5%	100%

Tabella 5.2: Risultati ottenuti con il Test 2. Si verificano tutti match corretti a meno di un falso positivo.

sono quelli mostrati in Tabella 5.3.

5.3.4 Risultati

Dalle tre Tabelle 5.1, 5.2 e 5.3 è possibile notare che tutti i test effettuati hanno prodotto dei buoni risultati, la Recall è sempre del 100% e la Precison totale non scende mai al di sotto dell'80%. Come ci si poteva aspettare, utilizzando esclusivamente il primo algoritmo per la ricerca sia nei tag che nelle descrizioni si ottengono dei risultati leggermente migliori (si verifica solo un falso positivo). Questo accade perchè tale algoritmo è stato studiato e realizzato appositamente per questo tipo di ricerca, mentre l'Algoritmo Levenshtein Distance è stato realizzato riadattando una tecnica di ricerca che solitamente viene utilizzata per confrontare testi e non singole parole.

Test 3	VP	VN	FP	FN	Precision	Recall
Acqua	7	19	0	0	100%	100%
Cascata	4	22	0	0	100%	100%
Corno	1	25	0	0	100%	100%
Indicazioni	2	23	1	0	67%	100%
Mon	4	20	2	0	67%	100%
Monte	3	22	1	0	75%	100%
Parch	1	25	0	0	100%	100%
Punto	1	25	0	0	100%	100%
Segnalazione	2	24	0	0	100%	100%
Seguire	1	24	1	0	50%	100%
Vista	2	24	0	0	100%	100%
TOTALE	28	253	5	0	84.8%	100%

Tabella 5.3: Risultati ottenuti con il Test 3. Si verificano 5 falsi positivi.

CAPITOLO 6

Conclusioni

L'obiettivo di questa tesi è stato quello di discutere le fasi di progettazione e realizzazione di un'applicazione mobile che permettesse la fruizione di percorsi escursionistici presenti nel Parco Nazionale del Gran Sasso e Monti della Laga e che offrisse inoltre agli utenti la possibilità di cercare contenuti multimediali quali audio, immagini e testi. Nel primo capito si è discusso dei requisiti minimi dell'applicazione e nel complesso l'app li ha rispettati. Durante la realizzazione di APPercorsi l'utilizzo della libreria OSMdroid ha permesso lo sviluppo dell'applicazione in modo molto più agevole, facilitando l'operazione di aggiunta della mappa, creazione di un percorso da seguire e la segnalazione dei Marker che identificano i contenuti multimediali. Allo stesso modo, anche l'utilizzo di Android Studio e il linguaggio Kotlin hanno giocato un ruolo fondamentale, in quanto hanno reso lo sviluppo dell'applicazione un'ottima occasione per accrescere le proprie competenze personali. Nell'ultimo capitolo sono stati analizzati gli algoritmi di ricerca sviluppati. Da tale analisi si può concludere che l'algoritmo sviluppato appositamente per soddisfare i requisiti di APPercorsi è leggermente migliore ed è più adatto alla ricerca di singole parole mentre il secondo algoritmo è adatto alla ricerca di stringhe composte da più parole, ciononostante entrambi hanno prodotto dei buoni risultati e risultano perfettamente funzionali per l'applicazione.

6.1 | Sviluppi futuri

L'applicazione sviluppata ha sicuramente ampi margini di miglioramento, tuttavia rappresenta un'ottima base di partenza per un'eventuale applicazione ufficiale

del Parco Nazionale del Gran Sasso e Monti della Laga. Tra i requisiti discussi nel primo capitolo alcuni riguardavano la possibilità di ottenere un'applicazione intuitiva, chiara nell'organizzazione dei contenuti e semplice da utilizzare per gli utenti finali, un possibile sviluppo futuro potrebbe riguardare sicuramente il miglioramento di questo aspetto, in modo da ottenere una user experience ottimale. Per rendere la funzionalità di ricerca ancora più accurata, un interessante sviluppo futuro potrebbe riguardare l'utilizzo di altri algoritmi di information retrieval, ad esempio basati su Latent semantic analysis, oppure approcci basati su knowledge representation e tecnologie semantiche.

Bibliografia

[1] http://www.gransassolagapark.it/itinerari.php.

[2] C. Fang, J. Liu e Z. Lei, " *Fine-Grained HTTP Web Traffic Analysis Based on Large-Scale Mobile Datasets*", in IEEE Access, vol. 4, pp. 4364-4373, 2016, doi: 10.1109/ACCESS.2016.2597538.

[3] Raffaella Papa, Tecnoandroid, https://www.tecnoandroid.it/2019/01/14/android-cose-come-fatto-e-come-funziona-il-sistema-operativo-di-google-460374, 14 Gennaio 2019.

[4] Mr APP's, http://www.mr-apps.com/it/blog/la-differenza-tra-app-native-app-ibride-e-web-app, 31 October 2018.

[5] Mike Cleron, Android Developers Blog, https://android-developers.googleblog.com/2017/05/android-announces-support-for-kotlin.html, 17 May 2017.

[6] Chet Haase, Android Developers Blog, https://android-developers.googleblog.com/2019/05/google-io-2019-empowering-developers-to-build-experiences-on-Android-Play.html, 07 May 2019.

[7] https://github.com/osmdroid/osmdroid.

[8] https://www.openstreetmap.org.

[9] https://it.wikiloc.com.

[10] https://github.com/osmdroid/osmdroid/wiki/How-to-use-the-osmdroid-library-(Kotlin).

[11] Google Developers, https://developer.android.com/guide/topics/ui/layout/recyclerview.

[12] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions Insertions and Reversals", Soviet Physics Doklady, vol. 10, pp. 707, Feb. 1966.

[13] K. Balhaf, M. A. Alsmirat, M. Al-Ayyoub, Y. Jararweh e M. A. Shehab, "Accelerating Levenshtein and Damerau edit distance algorithms using GPU with unified memory", 2017 8th International Conference on Information and Communication Systems (ICICS), 2017, pp. 7-11, doi: 10.1109/IACS.2017.7921937.