UNIVERSITÀ
POLITECNICA
DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione

APPercorsi: un'applicazione mobile per la fruizione e creazione di percorsi escursionistici del "Parco Nazionale del Gran Sasso e dei Monti della Laga"

APPercorsi: a mobile application for the enjoyment, use and creation of hiking trails of the "National Park of Gran Sasso and the Laga Mountains"

Relatore:

Prof. EMANUELE STORTI

Tesi di Laurea di:

CAMILLA D'ANDREA

ANNO ACCADEMICO 2020/2021

Indice

Elenco delle figure

Introduzione

Come si può scegliere dove fare escursionismo nella Regione Marche? Un territorio così ampio dal quale potremmo tirar fuori tantissime buone motivazioni per visitare i famosissimi ”Parco nazionale del Monti Sibillini” a Sud, oppure ”Parco del Monte Conero”a Nord. Per un turista alla scoperta del territorio spesso la risposta è insita nella domanda: ”Perche spostarsi? Se sono a Sud visiterò il Monte Vettore nella catena montuosa dei Monti Sibillini”. Come poter dargli torto, come poter anche dar torto a chi in quei territori ci è cresciuto, il quale ha percorso lo stesso sentiero innumerevoli volte? Da qui nasce questa tesi, per dare la possibilità di esplorare qualcosa di nuovo, di rispondere alla domanda ”Dove vado oggi?”. Quel qualcosa di nuovo è rappresentato dai sentieri che non si conoscono, o perlomeno che la stragrande maggioranza dell'utenza non conosce, situati nel meraviglioso ”Parco Nazionale del Gran Sasso e Monti della Laga”, caso di studio della Tesi. Il ”Parco Nazionale del Gran Sasso e Monti della Laga” viene spesso associato alla sola Regione Abruzzo per ”Gran Sasso” contenuto nel nome, eppure esso si estende per una superficie di 141.341 ettari, difatti è la terza riserva naturale protetta più grande d'Italia per estensione territoriale, situato tra Abruzzo (nella maggior parte), Marche e Lazio. Si è pensato così, di creare uno strumento capace di rendere fruibili, consultabili e condivisibili le ricchezze presenti sul territorio, ogni scorcio, non solo agli abitanti locali ma a tutti gli amanti della montagna. Per ricchezze si intende non solo i sentieri per escursionismo ma anche tutte quelle attrazioni-punti di interesse presenti nel percorso. Questo strumento nato nell'era dove il digitale ha portato cambimenti sociali ed economici e dove ognuno ha almeno uno smartphone, non poteva che essere un Applicazione Mobile. L'applicazione mobile diffusa in modo capillare permette così di raggiungere l'obiettivo di propagare e arricchire la conoscenza sul ”Parco Nazionale del Gran Sasso e Monti della Laga” e sui suoi tesori. Affinchè però l'applicazione mobile sia diffusa in uno spettro più ampio possibile sono state fatte delle considerazioni sul sistema operativo utilizzato dalla massa, sulla semplice e buona fruibilità e molto altro, ponendosi degli obiettivi nella realizzazione del progetto stesso. Concretamente quindi si è realizzata un'applicazione

mobile che oltre a permettere la fruizione di percorsi ne garantisce l'inserimento di nuovi. Questo lavoro di Tesi è così strutturato:

- **Capitolo 1:** si pongono gli obiettivi, inquadrando l'ambito ed elencando i requisiti dell'applicazione.

- **Capitolo 2:** si analizzano gli strumenti tecnici utilizzati per il lavoro, a partire da quelli di sviluppo fino alle sorgenti dei dati

- **Capitolo 3:** la trattazione si sofferma sull'applicazione di base cioè quella sviluppata nell'ambito del progetto didattico per il corso di "Programmazione Mobile", analizzandone ogni fase. Essa è il pilastro di quello che è poi lo sviluppo per la tesi, difatti consente l'autenticazione, previa registrazione nell'app e successivamente la visualizzazione dei percorsi, degli elementi multimediali associati e di un equipaggimento consigliato per ogni percorso.

- **Capitolo 4:** viene discussa nel dettaglio la funzionalità di inserimento di un nuovo percorso da parte di un utente e la relativa fruizione per tutti gli altri utenti che hanno scaricato l'applicazione. Si parte dalla fase di progettazione fino ad arrivare a quella dell'implementazione mostrando parti di codice salienti.

- **Capitolo 5:** si discutono le conclusioni del lavoro, riassumendo quanto svolto, soffermandosi sugli insegnamenti appresi e sui possibili prossimi sviluppi dell'applicazione.

1. Obiettivo e requisiti

Il primo capitolo sarà incentrato sulla presentazione del progetto a partire dal suo scopo, poi ci sarà un approfondimento della località scelta come caso di studio per lo sviluppo dell'applicazione. Infine saranno elencati i requisiti cioè le funzionalità che presenta l'app e i vincoli che rispetta, i cosidetti requisiti non funzionali.

1.1 Obiettivo

Questo progetto ha lo scopo di realizzare un'applicazione mobile per la montagna la quale consenta di visualizzare, seguire ed inserire nuovi sentieri. Lo scenario che è stato considerato nella presente tesi è quello del "Parco Nazionale del Gran Sasso e Monti della Laga". In particolare il lavoro si incentra nell'inserimento di un nuovo percorso, dovrà essere quindi possibile il tracciamento continuo della propria posizione, successivamente inserire informazioni riguardanti quel determinato percorso e poi renderlo disponibile agli altri utenti, salvando quindi il percorso su un database remoto e non in locale. Strumento imprescindibile è quindi l'utilizzo dello smartphone dotato di sensore GPS e connessione internet, mentre è facoltativo l'utilizzo della fotocamera per scattare una foto che sarà inserita come preview di un percorso. L'obiettivo sarà quindi soddisfatto quando successivamente all'autenticazione sarà possibile per l'utente visualizzare i percorsi inseriti dagli altri escursionisti e i propri in un'unica lista, inserirne di nuovi, creando così una sorta di community.

1.2 Ambito

Si è scelto come caso di studio per la realizzazione dell'applicazione mobile il *"Parco Nazionale del Gran Sasso e Monti della Laga"*.

1.2.1 Cos'è e dove si trova il Parco Nazionale del Gran Sasso e Monti della Laga

Il Parco è situato nel cuore dell'Appennino, estendendosi su tre regioni: l'Abruzzo, il Lazio e le Marche. A sua volta include all'interno dei sui confini cinque province: L'Aquila, Teramo, Pescara, Rieti, Ascoli Piceno, occupando ben 44 territori comunali[1] . L'importanza ecologica di questo territorio è insita nel ruolo di giunzione tra la regione euro-siberiana e quella mediterranea. In un contesto così eterogeno si localizza la montagna più alta dell'Appennino in cui è possibile rinvenire l'unico ghiacciaio dell'Europa meridionale. Sono rinvenibili tantissime specie animali e vegetali, nonché una varietà di ecosistemi unici. Questa ricchezza è garantita da una posizione geografica stategica, da un'altitudine moderata e dalle differenti geologie dei suoi ambienti.

Caratteristiche e peculiarità del territorio

Il territorio prevalentemente montano è dominato dalla presenza di tre massicci montuosi di diversa conformazione litologica. I Monti della Laga costituiti da arenarie, i Monti Gemelli e la catena del Gran Sasso d'Italia da rocce calcare e dolomie. Pur essendo un territorio appenninico non mancano vette superiori ai 2000 m di quota. Di queste meritano menzione la Macera della Morte (2073 m) a Nord sui Monti della Laga, Monte Siella (2000 m) a Sud nella catena del Gran Sasso d'Italia. Le vette si estendono per oltre 50 chilometri di creste affilate. A corredo di questi ambienti si rinvengono pareti rocciose, torrioni slanciati, forre e valli. L'unica "interruzione" è rappresentata dal Valico delle Capannelle (1300 m), che, posto circa a metà strada, unisce i calcari e le dolomie del Gran Sasso con le arenarie dei Monti della Laga[2].

[1] GranSassoLagaPark. *Territorio*. 2021. URL: `http://www.gransassolagapark.it/pagina.php?id=38`.

[2] GranSassoLagaPark. *La morfologia del parco*. 2021. URL: `http://www.gransassolagapark.it/pagina.php?id=232`.

(a) Stemma del parco

(b) Mappa del parco

Figura 1.1: Mappa e stemma del Parco Nazionale

1.2.2 Itinerari e percorsi

Uno dei modi migliori per vivere un territorio, e perchè no, sentirsi a casa, è organizzare dei percorsi e degli itinerari che diano la possibilità, non solo di cogliere le bellezze paesaggistiche, di visitare luoghi e conoscere personaggi che ne hanno fatto la storia, ma anche di incontrare persone del luogo, ascoltare musiche tradizionali e gustarne i sapori tipici.

Borghi storici

Tra le attrazioni offerte dal Parco non mancano i borghi e i paesi caratteristici. Spiccano, tra la natura vergine e incontaminata, città fortezza come la celebre Calascio, con la sua rocca suggestiva meta frequentata da migliaia di turisti ogni anno. Spiccano borghi suggestivi come quello dell'antica Baronia di Carapelle, prossima alla città capoluogo di regione, L'Aquila. Non mancano attrazioni culturali come il mediceo Santo Stefano di Sessanio e Castel del Monte nota per la transumanza, simbolo di una civiltà economicamente povera ma ricca di attrazioni e cultura[3].

Itinerari escursionistici

Gli itinerari escursionistici sono molteplici, alcuni dei quali percorribili anche a cavallo o in mountain bike, per citarne alcuni: "Da Umito alle Cascate della Volpara", "Da Campotosto al Monte Cardito", "Da Ripe a Gole del Salinello"... Alcuni di questi itinerari sono presenti nell'applicazione mobile, altri devono ancora essere inseriti.

1.3 Requisiti

In questa sezione vedremo in modo particolare una piccola analisi sui requisiti funzionali e non funzionali relativi all'applicazione per la consultazione e inserimento di percorsi.

1.3.1 Requisiti funzionali

I requisiti funzionali si presentano come elenchi di funzionalità o servizi che il sistema deve fornire.

- Creazione account (Registrazione all'applicazione)
- Autenticazione
- Visualizzazione lista dei percorsi

[3] GranSassoLagaPark. *I Borghi*. 2021. URL: `http://www.gransassolagapark.it/pagina.php?id=50/`.

- Visualizzazione informazioni relative ad un percorso
- Visualizzazione mappa relativa ad un percorso
- Visualizzazione punti di interesse presenti nel percorso con relativa fruizione dei contenuti multimediali presenti
- Seguire un percorso visualizzando la propria posizione sulla mappa
- Visualizzazione lista degli equipaggiamenti consigliati per un determinato percorso
- Inserimento di un nuovo percorso (mappa e informazioni relative ad esso)

1.3.2 Requisiti non funzionali

I requisiti non funzionali rappresentano i vincoli e le proprietà/caratteristiche relative al sistema, come vincoli di natura temporale, vincoli sul processo di sviluppo e sugli standard da adottare. Quelli individuati in questo progetto sono:

- Mobile
- Tipo di App
- Piattaforma di sviluppo
- Linguaggio
- User Interface

Di seguito saranno analizzati singolarmente.

Mobile

Dalla definizione di ”mobile” in inglese, e ormai adattata anche all'italiano, ricaviamo che si riferisce a qualcosa che può essere spostato, siamo in presenza quindi di un oggetto che non ha posizione fissa. Per essere ”mobile” un oggetto ha quindi bisogno di essere traportato, perciò la definizione deve essere legata anche alle dimensioni fisiche e al peso dell'oggetto in questione. L'oggetto che interessa approfondire nello studio è il Dispositivo Mobile definito da Treccani come:

> ”Per d. m. si intende, in generale, qualsiasi dispositivo dotato di comunicazione wireless in grado di accedere alle funzioni di rete, come navigare sul web, consultare la posta elettronica e interagire con i social network (v.), per es. un telefono cellulare, uno smartphone o un altro strumento (spesso multimediale). Poiché la chiamata vocale oggi può essere mediata da un applicativo, questa funzione non è più prerogativa esclusiva dei cellulari, ma estendibile a ogni d. m. dotato di interfacce audio di input/output, come i tablet.[4]”

Ai dispositivi mobile noti si sono aggiunti negli ultimi anni gli ”Indossabili” quali smartWatch o altri dispositivi per il tracciamento dell'attività fisica o della salute in senso ampio. In Figura 1.2 vengono rappresentate le percentuali dei dispositivi digitali degli italiani.

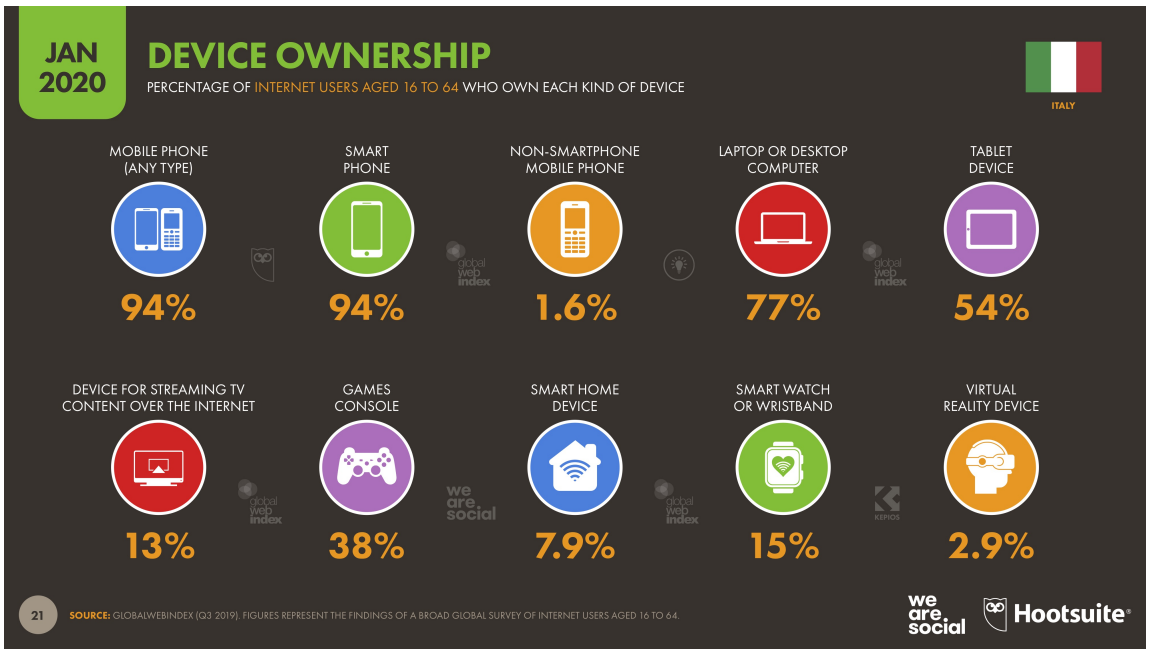


Figura 1.2: Dispositivi digitali in mano alla popolazione italiana

Per tutti questi dispositivi sono state progettate e realizzate delle specifiche applicazioni software denominate **Applicazioni Mobile**. In particolare nel caso di studio della Tesi è stata presa in esame un' applicazione mobile sviluppata per uno smartphone. Esse vengono progettate in base a specifici criteri legati all'hardware del dispositivo quali RAM,

[4]Paolo Marocco. *DISPOSITIVO MOBILE-Enciclopedia italiana*. 2015. URL: https://www.treccani.it/enciclopedia/dispositivo-mobile_%5C%28Enciclopedia-Italiana%5C%29/.

CPU, banda, batteria oppure alla presenza o meno della fotocamera o di sensori. Le app si possono classificare in 3 grandi categorie:[5]

1. App di gioco: sono tra i tipi di app più popolari difatti rappresentano un terzo di tutti i download di app e tre quarti di tutta la spesa dei consumatori.

2. App per la produttività: si concentrano sul miglioramento delle attività quotidiane semplificando varie attività come l'invio di e-mail, il monitoraggio dell'avanzamento del lavoro e molto altro.

3. App per lo stile di vita e l'intrattenimento: sempre più popolari, comprendono molti aspetti dello stile di vita personale e della socializzazione come appuntamenti, comunicazione sui social media e condivisione (e visione) di video. Alcune delle app più conosciute come Netflix, Facebook o TikTok, rientrano in questa categoria.

L'applicazione mobile sviluppata nella tesi è senza dubbio appartenente a quest' ultima categoria in quanto permette di coltivare passioni personali, condividerle, abbinarle alla cultura e ad uno stile di vita sano, camminando all'aria aperta.

Tipo di APP

Con il termine "applicazione mobile" definiamo o meglio generalizziamo un app scaricabile e fruibile da un utente sul proprio smartphone. Nella creazione di un app però bisogna sceglierne la tipologia, e questa scelta si rifletterà anche in fase di distribuzione. Esistono diversi tipi infatti di applicazioni mobile:

- **WEB APP:** sono app accessibili come siti web, senza nessuna differenza tra piattaforma, sistema di sviluppo e codice. Esse quindi non devono essere istallate, ma sarà sufficiente una connessione internet per raggiungerle.

- **APP NATIVA:** sono app sviluppate per uno specifico sistema operativo con un linguaggio di programmazione. Un applicazione sviluppata per Android scritta in Java

[5]Techopedia. *Applicazione Mobile*. 2020. URL: `https://www.techopedia.com/definition/2953/mobile-application-mobile-app`.

non funzionerà su un IPhone, come un App in Swift per iOs non funzionerà su uno smartphone Android.

- **APP IBRIDA:** rispetto alle app sviluppate in nativo sono più rapide da sviluppare e meno dispendiose. Il principale vantaggio sta nel fatto che funzionano su diverse piattaforme.

L'applicazione mobile sviluppata per il progetto didattico dal quale prende vita poi l'ampliamento della tematica come oggetto della tesi, è stata sviluppata sia come App Nativa, sia come App Ibrida con Flutter. Flutter è un framework opensource di Google per la creazione di applicazioni compilate in modo nativo per dispositivi mobili, web, desktop e incorporati da un'unica base di codice[6]. Il successivo sviluppo, cioè il lavoro di tesi incentrato nell'inserimento di un nuovo sentiero montano da parte di un utente è stato sviluppato solamente come App Nativa.

Incentrando quindi la trattazione sulle App Native, che come già detto sono App sviluppate con un linguaggio di programmazione che è principalmente usato dal dispositivo mobile e il suo sistema operativo, di seguito ne vengono mostrate alcune caratteristiche:

- Maggiormente affidabili
- Semplici, prestazioni migliori e miglior user experience
- Supportano operazioni online e offline

Tra i vantaggi di questa tipologia di App troviamo:

- Offrono un'ampia gamma di funzionalità poiché sfrutta il potenziale del dispositivo mobile.
- Hanno integrate la funzionalità di notifica push
- Consentono prestazioni del software veloci e reattive
- Offrono un'interfaccia utente (UI) che si adatta meglio all'utilizzo sistema operativo

[6]Flutter. *Homepage Flutter*. 2021. URL: `https://flutter.dev/`.

e negli svantaggi:

- Le app native richiedono codici diversi per ogni sistema operativo adottato, poiché ogni dispositivo deve avere la sua versione specifica dell'app. Per esempio, il codice tra un sistema operativo Android e un sistema operativo iOS dovrebbe essere diverso
- Richiedono un costo maggiore, e sono richiesti più sviluppatori per creare e gestire il codice per ogni piattaforma
- E' richiesto più tempo nello sviluppo dovuto alla creazione differenziata per i diversi sistemi operativi.

Piattaforma

L'applicazione mobile è stata sviluppata per il sistema operativo Android, specificamente con versione del S.O. Android 11, in quanto come già espresso, si richiede massima diffusione dell'app per rendere i sentieri il più condivisibili possibile. Le motivazioni sono molteplici, primo tra i quali la dominanza di Android come sistema operativo negli smartphone degli italiani. Il sistema operativo mobile di Google è installato in decine di migliaia di modelli di smartphone ed Apple non può competere. I dati rilevati nel trimestre Gennaio-Marzo 2018 da Kantar Worldpanel ComTech sui sistemi operativi per smartphone rivelano che la concorrenza all'interno del mondo Android ha continuato a intensificarsi mentre la quota di iOS (Apple) è rimasta stabile nei cinque mercati principali in Europa e negli Stati Uniti, in contrasto con la continua crescita cinese. La diffusione del sistema operativo Android va dal 59% negli Stati Uniti, al 70% dell'Italia, l'80% della Cina e l'87% del Brasile. Solo in Giappone gli utenti di iOs battono quelli di Android (57% contro 41%)[7] .
Nella Figura 1.3 è rappresentata la mappa dell'istituto di analisi Kantar che monitora il mercato dei dispositivi mobili. Il dato riguara il trimestre Marzo-Giugno 2021, i quali dati vanno a confermare il trend 2018 con una predominanza Android in Italia.

[7]Luca Tremolada. *Android vs iOs: chi controlla il mercato dei sistemi operativi mobili?* 2018. URL: https://www.infodata.ilsole24ore.com/2018/07/18/android-vs-ios-controlla-mercato-dei-sistemi-operativi-mobili/.

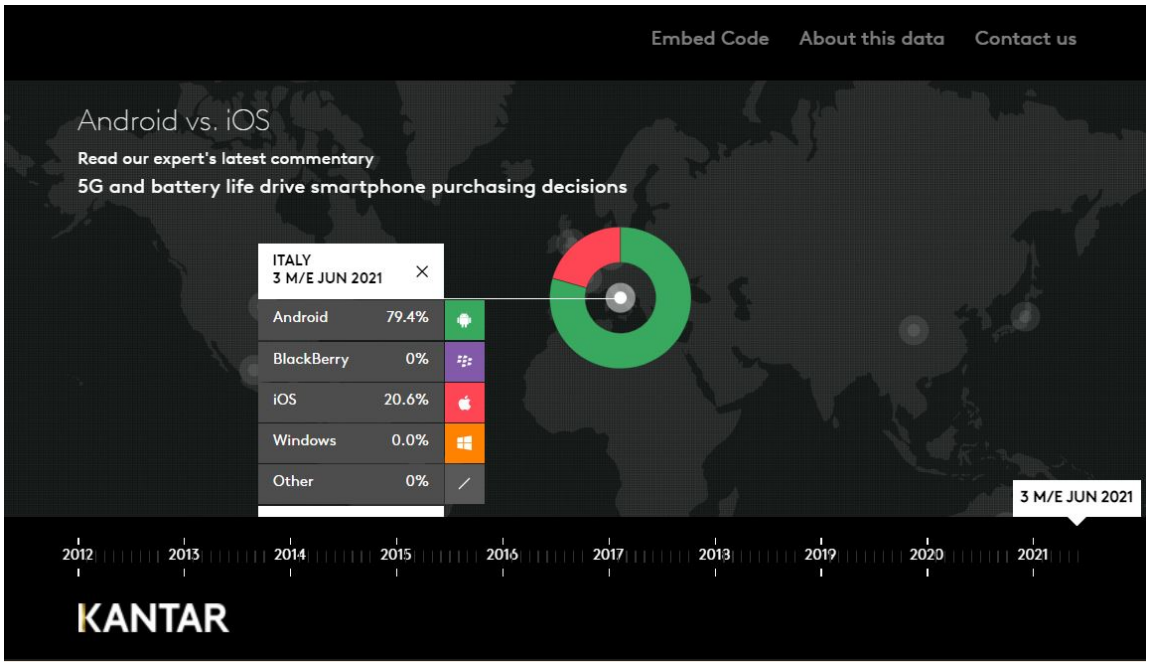


Figura 1.3: Mappa dell'istituto di analisi Kantar

Linguaggio

Le applicazioni Android come già visto possono essere scritte in linguaggi diversi come ad esempio Java o Kotlin. Quest ultimo è stato utilizzato nello sviluppo del progetto. Kotlin è un moderno linguaggio di programmazione, basato sul Java, tipizzato e utilizzato da oltre il 60%[8] degli sviluppatori Android professionisti. E' un linguaggio che soddifa gli sviluppatori anche per la sicurezza del codice, in quanto con @Nullable e @NonNull inclusi nel suo sistema di tipi, Kotlin aiuta a evitare NullPointerExceptions. Le app Android che utilizzano Kotlin hanno il 20%[9] di probabilità in meno di bloccarsi.

[8]Kotlin. *Sviluppa App Android con Kotlin*. 2021. URL: `http://www.developer.android.com/kotlin`.

[9]Kotlin. *Sviluppa App Android con Kotlin*. 2021. URL: `http://www.developer.android.com/kotlin`.

User Interface

Al 2019 esistevano circa 2 milioni di applicazioni mobile, numero destinato ad aumentare[10]. La continua diffusione e il miglioramento delle applicazioni mobile hanno innalzato le aspettative dell'utente al quale quindi si deve offrire sempre una User Experience ottimale. La User Experience negli anni si è legata molto alla User Interface che deve essere il piu possibile semplice, intuitiva e che non faccia sprecare tempo, con pochi click bisogna arrivare al risultato. Creare una UI comune per gli utenti Android mediante ad esempio Floating Action Button, Navigation Drawer o altre componenti, è essenziale affinchè anche gli utenti non esperti riescano ad utilizzare l'applicazione con facilità e in completa autonomia.

[10]Simone Cosimi. *Esistono 2 milioni di app, perché ne usiamo quattro?* 2019. URL: `https://www.esquire.com/it/lifestyle/tecnologia/a26997659/app-piu-usate/`.

2. Strumenti utilizzati

2.1 Strumenti tecnici

Nel seguente capitolo saranno elencati gli strumenti utilizzati per lo sviluppo del lavoro di Tesi, quali: l'ambiente di sviluppo, il database e la sorgente dei dati.

2.1.1 Android Studio IDE

L'ambiente di sviluppo utilizzato nello sviluppo del lavoro è Android Studio , IDE ufficiale per lo sviluppo di app Android, basato su IntelliJ IDEA[1]. Oltre al potente editor di codice e agli strumenti di sviluppo di IntelliJ, Android Studio offre altre funzionalità molto utili, come:

- Un sistema di build basato su Gradle
- Un emulatore ricco di funzionalità e personalizzzabile
- Un ambiente unificato in cui poter sviluppare per tutti i dispositivi Android
- Modelli di codice e integrazione con GitHub
- Strumenti e framework di test estesi (come Espresso)
- Supporto C++ e NDK

2.1.2 Firebase

Firebase è una piattaforma per la creazione di applicazioni per dispositivi mobili e web sviluppata da Google. Essa consente, una volta fatta l'operazione preliminare di creazione del progetto sulla piattaforma e collegamento tra il progetto Firebase e l'applicazione su

[1]Developers Android. *Meet Android Studio*. 2021. URL: `http://www.developer.android.com/studio/intro`.

Android Studio, di gestire l'autenticazione degli utenti all'interno dell'applicazione, l'accesso e la memorizzazione dei dati. Per una trattazione più approfondita, consultare il link ufficiale di Firebase: `https://firebase.google.com/`

Authentication

La sezione "Authentication" della console Firebase del progetto, permette di gestire gli utenti che si sono registrati nell'applicazione. Ogni utente è unico, cioè il campo "email" deve essere univoco quindi non sarà consentita la registrazione di più utenti con la stessa mail. Le operazioni che si possono effettuare all'interno della console di Firebase su ogni utente sono:

- Reimposta Password
- Disabilita Account
- Elimina Account

Firebase Authentication consente di utilizzare metodi di accesso come quelli classici cioè tramite Email e Password, utilizzato in questo lavoro, sia quelli che avvengono tramite piattaforme diverse, quali ad esempio Google o Facebook. Nella Figura 2.1 è rappresentata la console di gestione.

Per utilizzare "Authentication" all'interno del progetto dobbiamo andare ad inserire la dipendenza nel file Gradle:

```
implementation 'com.google.firebase:firebase-auth-ktx'
```

Una volta effettuati questi passaggi è possibile utilizzare nel codice i costrutti disponibili nella documentazione di Firebase, quali quelli di creazione di un nuovo utente oppure di login di un utente. La documentazione è disponibile la seguente link: `https://firebase.google.com/docs/auth/android/start`.

Realtime Database

Il database rappresenta il cuore dell'applicazione, senza esso non sarebbe possibile memorizzare e consultare i dati. Realtime Database è un database non relazionale (NoSQL)

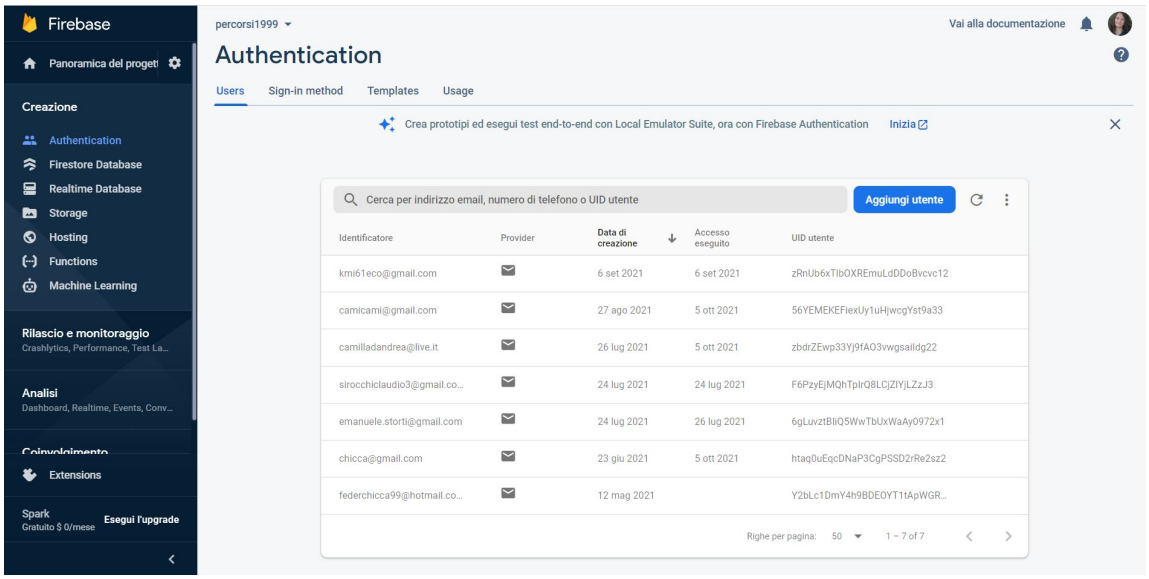


Figura 2.1: Authentication in Console Firebase

quindi non sono presenti entità e tabelle, bensì tutti i dati in Firebase Realtime Database vengono archiviati come oggetti JSON. Può essere pensata come un'associazione chiave valore. Nel caso del progetto, Realtime Database è stato utilizzato per la memorizzazione dei dati relativi ad ogni singolo percorso. Come mostrato nella Figura 2.2, vi è la radice dell'albero "Percorsi", al quale interno contiene i singoli percorsi. All'interno di ogni

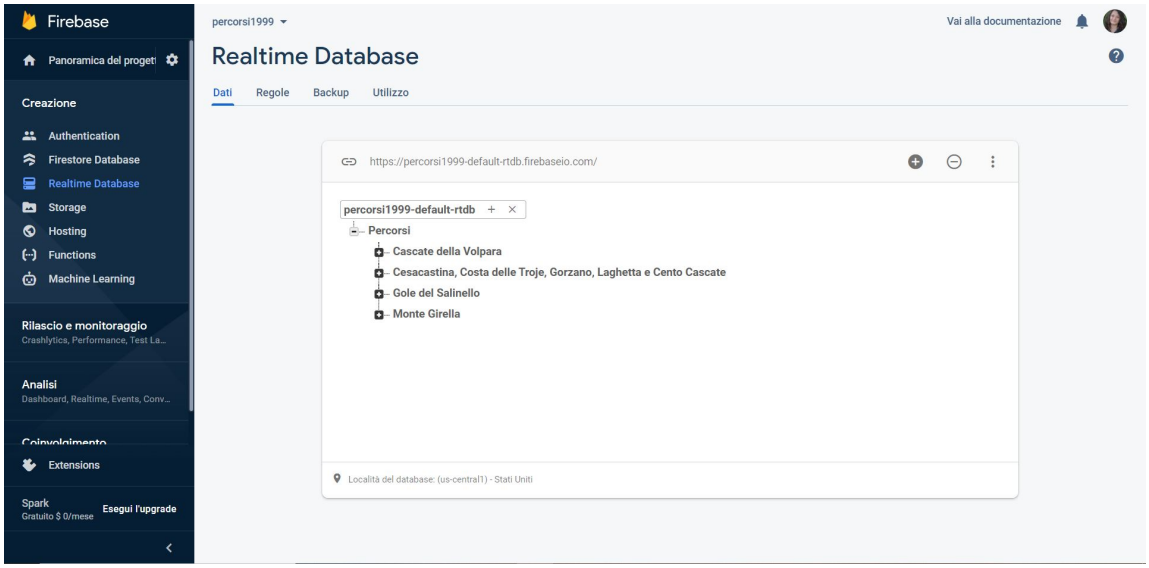


Figura 2.2: Struttura database

percorso poi vi sono i campi, il quale valore ne evidenzia le singole caratteristiche di ogni

percorso come in Figura 2.3.

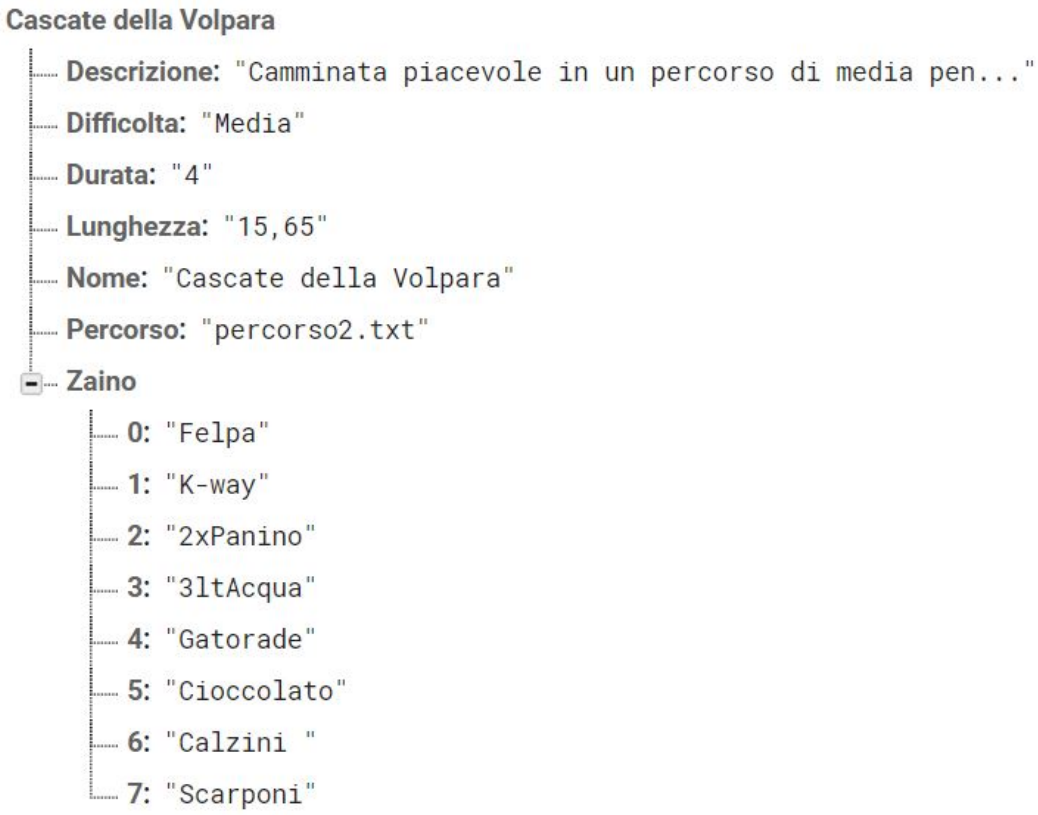


Figura 2.3: Struttura di ogni percorso

Per andare a manipolare i dati all'interno del codice in Android Studio è necessario aggiungere la seguente dipendenza nel file Gradle:

```
implementation 'com.google.firebase:firebase-database-ktx'
```

Dopodichè è possibile utilizzare le apposite funzioni definite nelle librerie per leggere o aggiungere valori al database. Per una trattazione più ampia si consiglia la visione della documentazione reperibile al sito web: `https://firebase.google.com/docs/database/android/start`

Storage

Lo Storage accessibile dalla console di Firebase rappresenta lo spazio di archiviazione dell'applicazione mobile, nel quale è possibile salvare file con diverse estensioni. Nello sviluppo dell'applicazione in esame, è stato utilizzato per archiviare:

- FILE DI TESTO: un file dall'estensione ".txt" per ogni percorso, il quale nome coincide con il nome dello stesso. Tale nome completo di estensione sarà il valore del campo "Percorso" all'interno del singolo percorso memorizzato in Realtime Database. Il file di testo contiene una lista di "latitudine, longitudine", i quali vanno a rappresentare una serie di punti che uniti costituiranno il tracciato relativo al percorso, visibile all'interno della mappa nell'applicazione.

- IMMAGINI: un file dall'estensione ".jpeg" per ogni percorso, il quale nome coincide con il nome dello stesso. Il nome, completo di estensione, sarà utilizzato per il caricamento di una preview dell'immagine nella lista di tutti i percorsi consultabile sull'applicazione. Per tale funzionalità è stata adoperata la libreria Glide la quale sarà trattata nel prossimo paragrafo. Sono presenti anche altre immagine utilizzate nei punti di interesse.

- FILE JSON: è un unico file all'interno del database denominato "POI.json" il quale all'interno contiene i riferimenti per i punti di interesse presenti in ogni percorso.

- REGISTRAZIONI: registrazioni vocali con estensione ".mp4" utilizzate come audio nei punti di interesse, collegate ad esso tramite il file "POI.json"

Per utilizzare lo storage all'interno del codice è stata aggiunta la seguente dipendenza nel file Gradle dell'applicazione:

```
implementation 'com.google.firebase:firebase-storage-ktx'
```

Con tale implementazione, viene resa possibile la lettura e la scrittura di file, mediante ad esempio l'utilizzo di stream di byte. Per una conoscenza più approfondita è possibile consultare la documentazione ufficiale al seguente link: `https://firebase.google.com/docs/storage/android/start`.

2.1.3 Glide

Glide è un framework open source veloce ed efficiente per la gestione dei media e il caricamento delle immagini per Android. L'obiettivo principale di Glide è rendere lo scorrimento di qualsiasi tipo di elenco di immagini il più fluido e veloce possibile, ma

Glide è anche efficace per quasi tutti i casi in cui è necessario recuperare, ridimensionare e visualizzare un'immagine remota[2]. Al fine di integrarla con l'ambiente di sviluppo e successivamente utilizzarlo è stato necessario aggiungere le seguenti dipendenze:

```
implementation 'com.github.bumptech.glide:glide:4.11.0'
annotationProcessor 'com.github.bumptech.glide:compiler:4.11.0'
```

2.1.4 OSMdroid

OSMdroid è una libreria di Android che fornisce gli strumenti per interagire con OpenStreetMap. Essa non solo consente di gestire le mappe ma anche di eseguire un tracciamento su di esse oppure di aggiungere delle forme di disegno quali dei marker. Per integrare OSMdroid con l'ambiente di sviluppo è necessario aggiungere prima delle dipendenze:

```
    implementation 'org.osmdroid:osmdroid-android:6.1.11'
    implementation 'com.github.MKergall:osmbonuspack:6.7.0'
```

Open Street Map

Open Street Map fornisce dati geografici in maniera *open data*, difatti i suoi dati sono liberamente distribuibili e copiabili con licenza Open Data Commons Open Database License (ODbL) dalla OpenStreetMap Foundation (OSMF), cioè attribuendoli a OSM. Essi vengono spesso utilizzati in GPS o altri dispositivi proprio per la peculiarità di essere open data al contrario di Google Maps. Open Street Map è composta da una community di mapper, professionisti GIS, ingegneri[3] che contribuiscono e mantengono i dati e le mappe costantemente aggiornati.

2.2 Dati

I dati presenti nell' app, quali: le informazioni, le mappe e le foto dei singoli percorsi sono stati estrapolati da Wikiloc.

[2]Github. *About Glide*. 2021. URL: https://bumptech.github.io/glide/.

[3]Open Street Map. *About*. 2021. URL: https://www.openstreetmap.org/about.

2.2.1 Wikiloc

Wikiloc è una community di persone le quali condividono i migliori percorsi all'aria aperta per ciclismo, escursionismo e molte altre attività. Essa è disponibile sia come App per iOS/Android, sia come sito web. Nel presente caso di studio si è effettuata una ricerca sui percorsi presenti nel "Parco Nazionale del Gran Sasso e dei Monti della Laga". Per iniziare a popolare l'applicazione, sono stati scelti alcuni percorsi tra i quali, a titolo esemplificativo e non esaustivo:

- **Monte Girella** → URL:`https://it.wikiloc.com/percorsi-escursionismo/monte-girella-76868606`
- **Gole del Salinello** → URL: `https://it.wikiloc.com/percorsi-escursionismo/gole-salinello-74725372#morePhotos`
- **Cascate della Volpara** → URL: `https://it.wikiloc.com/percorsi-escursionismo/acquasanta-umito-cascate-della-prata-e-volpara-52958730#morePhotos`

Ogni link all'apertura presenterà il percorso come mostrato nella Figura 2.4

Da ognuno di loro sono stati estrapolati i seguenti dati:

- **informazioni**: presenti al lato destro della visualizzazione della pagina web relativa al percorso.
- **foto**: si è scelta un'immagine tra quelle caricate dall'utente per essere mostrata nella RecyclerView la quale permette la visualizzazione di tutti i percorsi.
- **mappa**: scaricabile tramite il pulsante "Scarica" presente in alto a destra sullo schermo. Una volta effettuata la scelta si è reindirizzati alla seguente pagina visibile in Figura 2.5 (previa autenticazione), che permette di decidere in che formato si richiede il download. Il download è stato effettuato con le opzioni di default "Formato GPX, semplificato a 500 punti traccia".

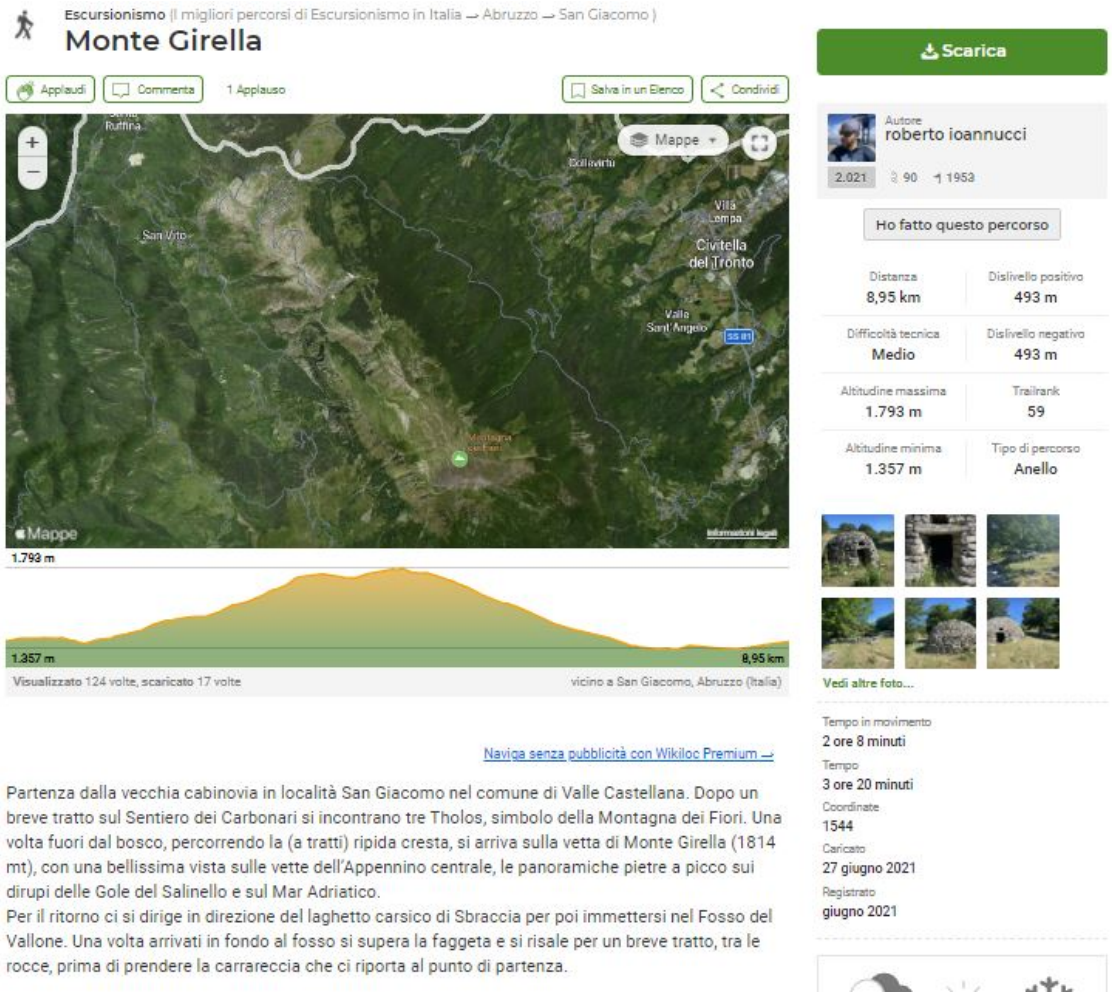


Figura 2.4: Visualizzazione percorso su Wikiloc

2.2.2 Manipolazione dati

Come già detto, la mappa che mostra il percorso crea un tracciato da un file di testo contenente ”latitudine, longitudine”, quindi una volta scaricato il percorso in formato GPX è necessario:

- Convertire il file di formato GPX in formato TXT, cambiando l'estensione
- Aprire il TXT e estrapolare i punti del percorso racchiusi nel tag <trkseg> come mostrato nella Figura 2.6 memorizzandoli in un nuovo txt, dove ogni riga sarà composta da ”latitudine, longitudine” come rappresentato in Figura 2.7. Questo nuovo file di testo sarà denominato con lo stesso nome del percorso.

Figura 2.5: Schermata per il download del percorso

Discorso analogo avviene per i punti di interessi che saranno mostrati nel percorso, vengono estratti dal file .gpx convertito in un file .txt e vengono immessi in un file JSON costruito come mostrato in Figura 2.8. I dati vengono così manipolati al file di utilizzarli all'interno dell'applicazione perchè come sarà descritto in seguito il tracciato rappresentato sulla mappa è dato dall'unione di una lista di punti quali "latitudine, longitudine".

Figura 2.6: Estrarre i punti dai tag

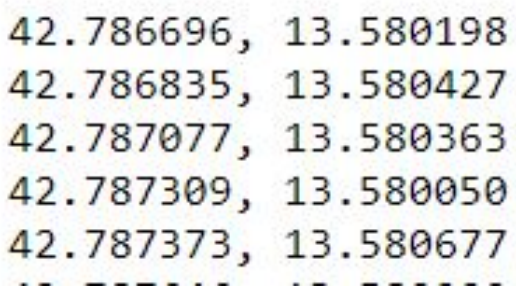


Figura 2.7: Costruzione nuovo file di testo

```
{
   "Media":[
      {
         "DataType":"audio",
         "Title":"Parcheggio",
         "Path":"Cascate della Volpara",
         "Desc":"Parcheggio ad Umito",
         "Lat":"42.737513",
         "Long":"13.406964",
         "Reg":"Registrazione11.m4a",
         "Tag":[
            "parking",
            "parcheggio",
            "parcheggi",
            "auto",
            "automobile"
         ]
      },
      {
         "DataType":"audio",
         "Title":"Indicazioni",
         "Path":"Cascate della Volpara",
         "Desc":"Prime indicazione di sentiero",
         "Lat":"42.737158",
         "Long":"13.405498",
         "Reg":"Registrazione12.m4a",
         "Tag":[
            "inizio"
         ]
      },
```


Figura 2.8: File JSON

3. Applicazione di base

In questo capitolo sarà discussa l'applicazione per il progetto di base cioè quello che è stato sviluppato nel corso di "Programmazione Mobile", il quale poi è stato precursore del lavoro di tesi. L'applicazione di base consente la registrazione e l'autenticazione all'interno dell'app, nel quale poi è possibile consultare una lista dei percorsi e per ognuno di essi le informazioni relative, la mappa con il tracciato e punti di interessi e in conclusione l'equipaggiamento consigliato.

3.1 Progettazione ed implementazione

La progettazione è una delle fasi più importanti nel progetto, essa precede le fasi di realizzazione e testing finale della qualità. Al suo interno vengono definiti: l'obiettivo, i requisiti dell'applicazione, gli strumenti da utilizzare (tutti elementi discussi in precedenza), i casi d'uso e l'interfaccia grafica. Ne segue poi l'implementazione cioè la realizzazione vera e propria dell'applicazione, in questo elaborato saranno mostrati dei frammenti di codice che ne hanno permesso lo sviluppo.

3.1.1 Casi d'uso

L'utente come già discusso dopo aver effettuato l'autenticazione può visualizzare la lista dei percorsi e poi ogni percorso piu in dettaglio. Le azioni che lui può svolgere sono visibili tramite l'apposito diagramma in Figura 3.1.

3.1.2 Mockup

I mockup sono uno strumento utile per creare una rappresentazione statica del prodotto finale con il più alto livello di dettaglio e fedeltà possibile. Si visualizzano, ora, i mockup realizzati; nello specifico quelli relativi a:

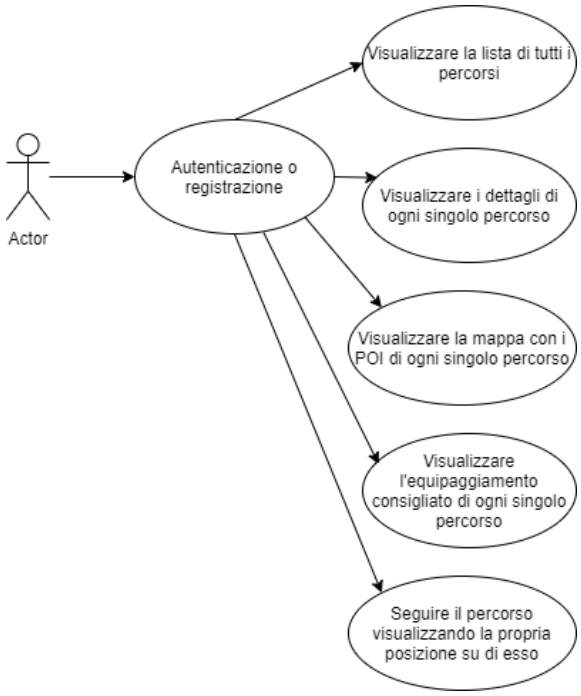


Figura 3.1: Diagramma casi d'uso

- **FirstScreen:** è la schermata di benvenuto, viene mostrata come prima schermata all'apertura dell'applicazione, visibile in Figura 3.2.
- **Login:** è la schermata nella quale viene concesso il login all'utente, previa scelta nella FirstScreen di effettuare il Login, visibile in Figura 3.3.
- **Registrazione:** è la schermata nella quale viene concessa la registrazione all'utente, previa scelta nella FirstScreen di effettuarla, visibile in Figura 3.3.
- **Main:** è la schermata al quale l'utente accede previa autenticazione, dove può effettuare delle scelte sul come proseguire, come mostrato in Figura 3.4.
- **Lista percorsi:** è la schermata alla quale l'utente accede una volta effettuata la scelta "Visualizza Percorsi" (unica scelta possibile nell'app di base oltre il "LOGOUT"). In essa è rappresentata la lista con i percorsi disponibili, è possibile visionarla in Figura 3.5.
- **Dettagli Percorso:** è la schermata che dettaglia tramite delle informazioni il singolo percorso, rappresentata in Figura 3.6.

Figura 3.2: Mockup FirstScreen

- **Mappa:** è la schermata che mostra il tracciato con i punti di interesse, rappresentata in Figura 3.6.

- **Zaino:** è la schermata nel quale viene mostrato l'equipaggiamento consigliato per quel percorso, rappresentata in Figura 3.6.

3.1.3 Struttura progetto

All'interno del progetto si trovano le activity e i fragment, file contenenti il codice sorgente con estensione .kt, i quali gestiscono la parte logica. Sono visibili in Figura 3.7. Ad affiancare le risorse di codice si trovano tutte le risorse di "stile", come layout XML, stringhe dell'interfaccia utente, colori dei temi e immagini bitmap, visibili in Figura 3.8.

Altri file importanti all' interno di un progetto sono:

- **AndroidManifest.xml** nel quale vengono esplicitate proprietà importanti del progetto quali nome, icona, permessi, activity e pagina iniziale.

- **Build.gradle** file del gestore delle dipendenze Gradle, il quale ci permette di gestire le stesse.

(a) Mockup schermata Login

(b) Mockup schermata Registrazione

Figura 3.3: Mockup dell'autenticazione

3.1.4 Permessi

Impostare delle autorizzazioni all'applicazione consente di supportare la privacy degli utenti proteggendo l'accesso a quanto segue:

- **Dati limitati**, come lo stato del sistema e le informazioni di contatto di un utente.
- **Azioni limitate**, come la connessione ad un altro dispositivo e la registrazione dell'audio.

I permessi in base al loro "grado di invasione" nella privacy dell'utente sono classificati in permessi **"Normal", "Signature" o "Dangerous"**. Nelle autorizzazioni classificate come "Normal" che consentono l'accesso a dati e azioni che si estendono oltre la sandbox dell'app senza richiedere l'autorizzazione all'utente rientrano permessi come il bluetooth, il Wi-Fi, internet, accesso alla memoria, fotocamera e altri. I permessi "Signature" invece sono permessi dove: se l'app dichiara un'autorizzazione per la firma definita da un'altra app e se le due app sono firmate dallo stesso certificato, il sistema concede l'autorizzazione alla prima app al momento dell'installazione. In caso contrario, non sarà possibile conce-

Figura 3.4: Mockup Main

dere l'autorizzazione alla prima app. Ultima categoria sono i permessi "Dangerous", dei quali viene chiesta l'autorizzazione all'utente a runtime in quanto accedono a delle informazioni sensibili, un esempio nell'applicazione sviluppata è stato il permesso riguardante la propria posizione. Quelli dichiarati all'interno del Manifest di APPercorsi sono:

```
1 <uses-permission android:name="android.permission.
  ACCESS_FINE_LOCATION" />
2   <uses-permission android:name="android.permission.INTERNET" />
3   <uses-permission android:name="android.permission.
  ACCESS_NETWORK_STATE" />
4   <uses-permission android:name="android.permission.
  ACCESS_COARSE_LOCATION" />
5   <uses-permission android:name="android.permission.
  READ_EXTERNAL_STORAGE" />
6   <uses-permission android:name="android.permission.
  WRITE_EXTERNAL_STORAGE" />
7   <uses-permission android:name="android.permission.
  WRITE_INTERNAL_STORAGE" />
```

Figura 3.5: Mockup lista dei percorsi

3.1.5 Autenticazione

Per l'autenticazione è stato utilizzato Firebase Authentication come illustrato nel paragrafo 2.1.2. Il login e la registrazione sono stati possibili quindi utilizzando funzioni della relativa libreria importata. Attenendosi il più fedelmente possibile a quanto definito in fase di progettazione tramite l'uso dei mockup, le schermate dell'applicazione relative al login e alla registrazione sono presentate in Figura 3.9.

Il codice che implementa il login, presente quindi in LoginActivity.kt, il quale si attiva quando l'utente preme il bottone relativo al Login e dopo aver estrapolato dalle TextInputEditText gli input dell'utente, quali username e password, è il seguente:

```
1 auth.signInWithEmailAndPassword(mail.text.toString(), pass.text.
    toString())
2                     .addOnCompleteListener(this) { task ->
3                         if (task.isSuccessful) {
4                             val user = auth.currentUser
5                             updateUI(user)
6                         } else {
7                             updateUI(null)
8                         }
```

(a) Mockup Dettagli Percorso (b) Mockup Mappa (c) Mockup Zaino

Figura 3.6: Mockup singolo percorso

Esso richiama la funzione UpdateUI che è definita come di seguito:

```
    private fun updateUI ( currentUser: FirebaseUser?) {
        if (currentUser !=null) {
            startActivity(Intent(this,MainActivity::class.java))

        } else {
            Toast.makeText(baseContext, "Login fallito",
                Toast.LENGTH_SHORT).show()
        }
    }
```

Essa quindi in caso di utente autenticato provvede a chiamare la MainActivity.kt, con l'uso dell'intent esplicito.

Per la registrazione è stato utilizzato il metodo "create" nella RegisterActivity, come di seguito mostrato:

```
auth.createUserWithEmailAndPassword(usr.text.toString(), pass.text.
    toString())
                    .addOnCompleteListener(this) { task ->
```

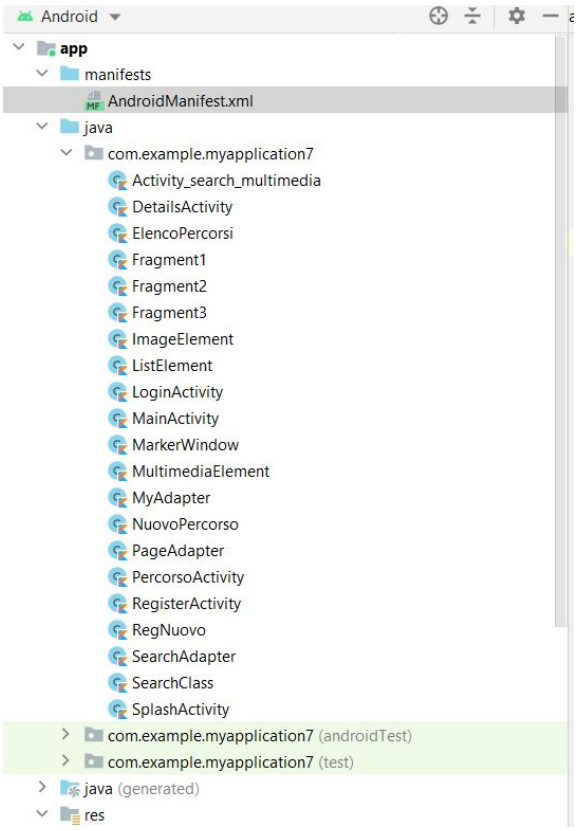


Figura 3.7: Le activity e i fragment nella vista dell' applicazione da Android Studio

```
3                       if (task.isSuccessful) {
4                           val user = auth.currentUser
5                           startActivity(Intent(this, LoginActivity
    ::class.java))
6                           finish()
7                       } else {
8                           Toast.makeText(
9                               baseContext, "Authentication failed
    .",
10                              Toast.LENGTH_SHORT
11                          ).show() }  }
```

3.1.6 Homepage

Dopo aver effettuato l'autenticazione all'interno di APPercorsi, si viene reindirizzati alla homepage dell'App, rappresentata dal MainActivity.kt. Essa si presenta come in Figura

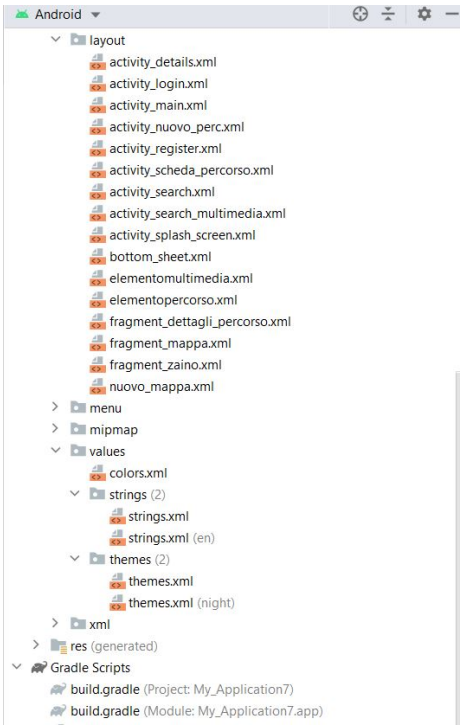


Figura 3.8: Risorse di progetto dalla vista di Android Studio

3.10. Come è visibile dalla pagina sono presenti quattro Floating Action Button, i quali rispettivamente permettono :

1. l'inserimento di un nuovo percorso da parte dell'utente → lavoro di tesi svolto e approfondito nel capitolo successivo

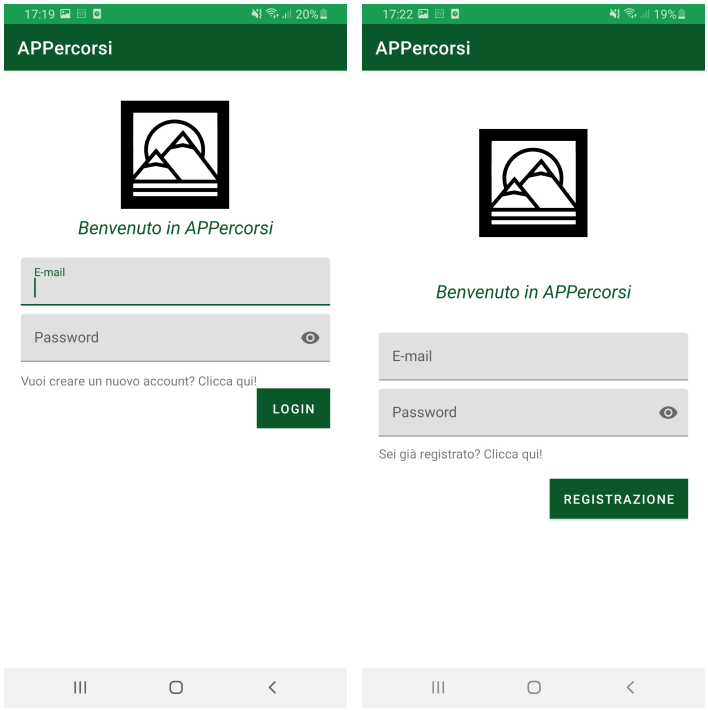
2. la ricerca di un contenuto multimediale → lavoro di tesi di Federica Ripani

3. la visualizzazione della lista dei percorsi → viene attivato un intent contenente la RecyclerView

4. il Logout dall'applicazione → viene utilizzato il metodo *SignOut()* di Firebase Authentication richiamando la LoginActivity

3.1.7 Lista dei percorsi

Per visualizzare la lista dei percorsi si utilizza la RecyclerView. La RecyclerView è una view group che consente di creare un lista di dimensione non fissata e di elementi (item)



(a) Schermata di Login (b) Schermata di registrazione

Figura 3.9: Screenshoot schermate di autenticazione

che si aggiornano dinamicamente.[1] Come suggerisce il nome, RecyclerView ricicla questi singoli elementi. Quando un elemento scorre fuori dallo schermo, RecyclerView non ne distrugge la visualizzazione; al contrario, RecyclerView riutilizza la vista per i nuovi elementi che sono stati visualizzati sullo schermo. Questo riutilizzo migliora notevolmente le prestazioni, migliorando la reattività dell'app e riducendo il consumo energetico. Ogni RecyclerView ha bisogno di un Adapter il quale consente di associare alla lista il layout di ogni singolo item e i suoi dati. In questo progetto la RecyclerView viene definita all' interno di ElencoPercorsi.kt e viene anche collegato il relativo adapter definito in MyAdapter.kt. MyAdapter.kt estende la classe RecyclerView.Adapter<MyAdapter.MyViewHolder>() e ha come parametro "private val routeList : ArrayList<ListElement>", dove ListElement è una data class contente i dati da visualizzare. Nell'adapter sono stati fatti gli override dei tre metodi:

[1] Developers Android. *Create dynamic lists with RecyclerView*. 2021. URL: `https://developer.android.com/guide/topics/ui/layout/recyclerview`.

Figura 3.10: Schermata relativa alla homepage

- **getItemCount()** il quale restituisce il numero di item della lista

```
1    override fun getItemCount(): Int {
2        return routeList.size}
3
```

- **onCreateViewHolder()** nel quale viene definito il layout di ogni item

```
1   override fun onCreateViewHolder(parent: ViewGroup, viewType:
     Int): MyViewHolder {
2
3        val itemView = LayoutInflater.from(parent.context).
   inflate(R.layout.elementopercorso,
4            parent,false)
5        contexto = parent.context
6        return MyViewHolder(itemView) }
7
```

- **onBindViewHolder()** nel quale vengono definiti i dati da mostrare in ogni item, qui si utilizza il Glide introdotto nel paragrafo 2.1.3 per la visualizzazione di una piccola

immagine rappresentante il percorso.

```
override fun onBindViewHolder(holder: MyViewHolder, position:
  Int) {
       val currentitem = routeList[position]
       holder.Nome.text = currentitem.Nome
       holder.Durata.text = currentitem.Durata
       holder.Lunghezza.text = currentitem.Lunghezza
       var imagename= currentitem.Nome
       var imageRef = Firebase.storage.reference
       val  dbref = imageRef.child("$imagename.jpeg")
      dbref.downloadUrl.addOnSuccessListener {
           Glide.with(contexto)
               .load(it)
               .into(holder.image) }
       holder.rowlayout.setOnClickListener { v: View ->
           val intent = Intent(contexto, PercorsoActivity::
  class.java)
           intent.putExtra("Nome",routeList[position].Nome)
           intent.putExtra("Diff", currentitem.Difficolta)
               contexto.startActivity(intent) }
   }

```

La lista dei percorsi è presentata come mostrato in Figura 3.11.

3.1.8 Singolo percorso

Nell' istante in cui si seleziona un item raffigurante il singolo percorso, si attiverà un ViewPager con una Tab superiore la quale consente la navigazione tra i tre fragment associati; ciò è definito in PercorsoActivity.kt . Il passaggio dei dati tra un'activity ad un'altra è avvenuto tramite il metodo *.putExtra()* applicato all'intent, il quale ha consentito il passaggio del nome del percorso, valore chiave per estrapolare i dati relativi al singolo percorso da Firebase. I fragment associati ad ogni percorso hanno funzionalità differenti e sono illustrati singolarmente di seguito.

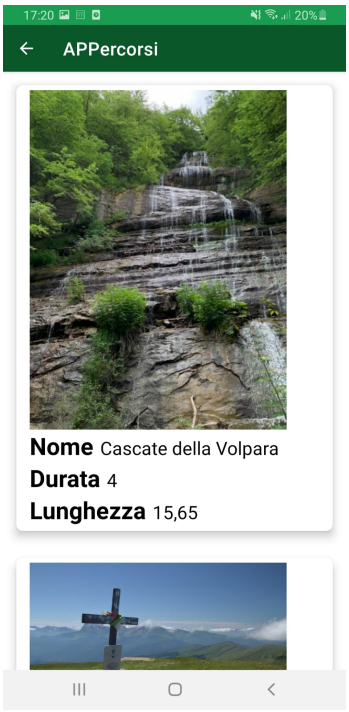


Figura 3.11: Schermata relativa alla RecyclerView

Dettagli Percorso

E' il primo fragment il quale rappresenta i dettagli del percorso, si presenta come in Figura 3.12. I dettagli sono estrapolati dai campi di ogni singolo percorso da Realtime Database. I campi interessati sono:

- Nome
- Difficoltà
- Durata
- Lunghezza
- Descrizione

All'interno del progetto il codice relativo è presente nel Fragment1.kt, qui si recuperano prima le View Text nel quale si inseriscono i dettagli, poi si estrapolano i dati tramite le query dal database, successivamente le informazioni estratte si trasformano in dei JSONObject. Una volta fatto ciò, questi JSONObject, vengono traformati in stringhe e poi

Figura 3.12: Schermata relativa al fragment ”Dettagli Percorso”

possono essere visualizzati all'interno del fragment. Nel seguente frammento di codice viene rappresentata l'estrazione dei dati (la query) e la traformazione in JSONObject:

```
1   mDatabase.child("Percorsi").child(x).get().addOnSuccessListener
    {
2                   val resultString = it.value
3                   val resultJson =  JSONObject(Gson().toJson(
    resultString))
4                   ParseData(resultJson)
5                   Log.i("TEST", ("Got value $resultJson"))
6
7               }.addOnFailureListener {
8                   Log.e("TEST", "Error getting data", it  }
9
```

Dove *mDatabase* è un campo che rappresenta il riferimento al database, *x* invece rappresenta il nome del percorso passato come argomento dall'activity chiamante al fragment.

Mappa

La mappa rappresenta il secondo fragment accessibile dal Tab posto in alto, in cui è possibile visualizzare la mappa con il percorso selezionato ed i punti di interesse, con la posizione dell'utente sovrapposta. Questa sarà visualizzata come mostrato nella Figura 3.13.

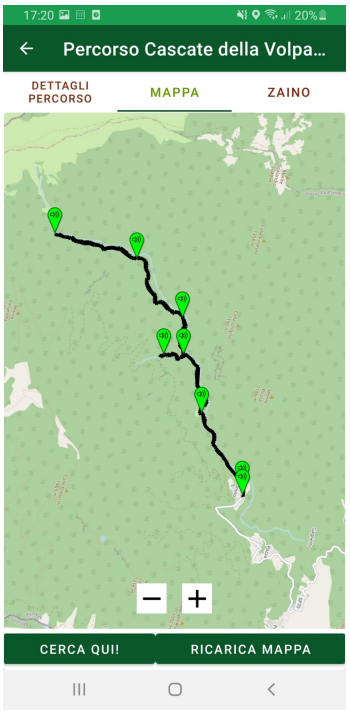


Figura 3.13: Schermata relativa alla mappa

Per mostrare la mappa è stata utilizzata la libreria OSMdroid, introdotta nel paragrafo 2.1.4. Seguendo la guida reperibile al link `https://github.com/osmdroid/osmdroid/wiki/How-to-use-the-osmdroid-library-(Kotlin)` la mappa è stata introdotta con il seguente frammento di codice, previo controllo dei permessi per localizzare le propria posizione su di essa:

```
1
2  map = vista.findViewById<MapView>(R.id.map)
3        map.isClickable = true
4        map.setBuiltInZoomControls(true)
5        map.setMultiTouchControls(true)
6        map.setTileSource(TileSourceFactory.MAPNIK)
```

```
        val mapController = map.controller
        mapController.setZoom(18)
          val mMyLocationOverlay = MyLocationNewOverlay(
GpsMyLocationProvider(context), map)
        mMyLocationOverlay.enableMyLocation()
        mMyLocationOverlay.enableFollowLocation()
        mMyLocationOverlay.isDrawAccuracyEnabled = true
        mMyLocationOverlay.runOnFirstFix {
            FragActivity?.runOnUiThread {
                mapController.setCenter(mMyLocationOverlay.
myLocation)
                mapController.animateTo(mMyLocationOverlay.
myLocation)}
        }
        map.overlays.add(mMyLocationOverlay)
```

Il passo successivo alla realizzazione della mappa è stata la visualizzazione del percorso su di essa, questo è stato possibile con l'utilizzo di una *PoliLine*, cioè una lista di punti graficamente unita. Il percorso risiede su un file di testo contenente una lista di "latitudine, longitudine", all'interno dello storage e collegato al singolo percorso tramite il campo *Percorso* in Realtime Database. Perciò si procede recuperando prima il file di testo collegato, successivamente si va a creare un file locale come copia dello stesso, che sarà eliminato dopo avere creato la Polines.

```
val localFile = File.createTempFile(x, ".txt")
            seriesRef.getFile(localFile).addOnSuccessListener {
                // Local temp file has been created
                fragAct?.let { fragmentActivity ->
                    try {
                        val lines: List<String> = localFile.
readLines()
                        val poly : Polyline = Polyline()
                        val linea1 = lines[0]
                        val size = (lines.size -1)
                        for ( i in 0..size) {
                            val point = lines[i].split(",")
```

```
                            val latitude  = point[0].toDouble()
                            val longitude = point[1].toDouble()
                            val PointOfPoliline = GeoPoint(latitude,
    longitude)
                            poly.addPoint(PointOfPoliline)
                        }
                        map.overlays.add(poly)
                        localFile.delete()
                    }
```

Inoltre sul percorso sono presenti dei POI (Point Of Interest), i quali sono memorizzati su un file JSON all'interno dello storage e rappresentati all'interno della mappa mediante l'utilizzo dei marker. Il codice relativo al caricamento è il seguente:

```
val localFile = File.createTempFile("POI", ".json")
      ref.getFile(localFile).addOnSuccessListener {
          // Local temp file has been created
          val resultString = localFile.readText()
          val resultJson = JSONObject(resultString)
          //prendo tutti i multimedia
          allPOI = resultJson.getJSONArray("Media")
          //filtro per percorso, seleziono quelli relativi
          filterPerc = search.mediaFilter(allPOI, gt)
          //per tutti i punti di interesse trovati si creano i
 markers
          for(i in 0..((filterPerc.length())-1)) {
              var m = Marker(map)
              var image = String()
              var jsonObj = filterPerc.getJSONObject(i)
```

Su ogni marker è poi gestito il tap su di essi. Essendo degli elementi multimediali, se si tratta di un audio verrà attivata la riproduzione, mentre se si tratta di un'immagine si aprirà una finestra per la visualizzazione. Questi elementi multimediali risiedono fisicamente sullo Storage di Firebase. Per una trattazione approfondita dei marker con una ricerca per descrizione, audio e/o immagine su di essi, si rimanda alla Tesi *"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" di Federica Ripani.

Zaino

E' il terzo fragment il quale rappresenta lo zaino del percorso, cioè l'equipaggiamento consigliato per esso. Si presenta come in Figura 3.14.

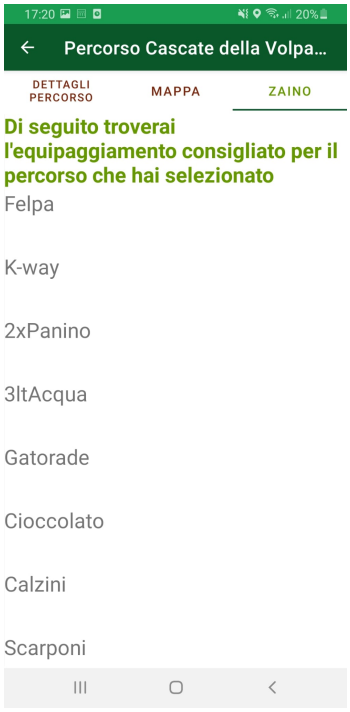


Figura 3.14: Schermata relativa allo zaino

L'equipaggiamento viene estrapolato dal campo "Zaino" di ogni percorso. All'interno vi sono gli 8 elementi che saranno mostrati. L'estrazione e la visualizzazione di essi viene gestita in Fragment3.kt e avviene proprio come nel Fragment1.kt, dapprima l'estrazione dal database, la trasformazione in JSONobject e poi la stampa nella TextView con il metodo *.toString* applicato al JSONObject.

4. Funzionalità di inserimento

La trattazione di questo capitolo è incentrata sull'argomento della Tesi cioè la creazione, da parte di un utente, di un nuovo itinerario escursionistico che possa essere condiviso con gli altri utenti. Fino ad ora difatti l'applicazione consentiva la sola visualizzazione dei percorsi completi di dettaglio, mappa ed equipaggiamento. La lista dei percorsi era però di default, cioè limitata a quelli inseriti manualmente dal programmatore in fase di sviluppo dell'applicazione. Era quindi un'applicazione non destinata ad una crescita ed un'espansione, bensì un'app che sarebbe stata gran presto superata perchè in montagna proprio come nell'universo della tecnologia ci sono sempre nuove scoperte.

4.1 Progettazione

E' stata la prima fase del lavoro, prima dello sviluppo e della stesura di questo elaborato. Qui si è capito come utilizzare gli strumenti per compiere un lavoro che soddisfi al meglio gli obiettivi, analizzando gli stessi.

4.1.1 Casi d'uso

Oltre alle azioni elencate nel capitolo precedente l'utente ne può svolgere un'altra fondamentale, il caso di studio di questa tesi. Nel diagramma mostrato nella Figura 4.1, l'azione è evidenziata in rosso.

4.1.2 Progettazione logica

Cosa vuol dire esattamente memorizzare un percorso per un utente? Esso consta di due parti fondamentali:

- **Registrazione del tracciato**: cioè memorizzare il percorso che si sta seguendo. E' quello che effettivamente sarà visibile sulla "Mappa" di ogni sentiero.

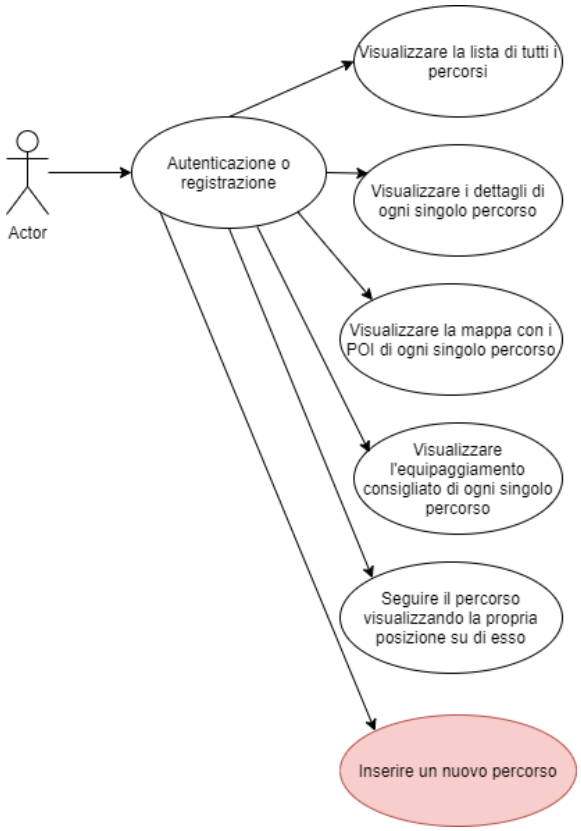


Figura 4.1: Diagramma casi d'uso

- **Inserimento delle informazioni:** cioè inserimento delle informazioni relative al percorso tracciato visibili in "Dettagli percorso" e l'equipaggiamento consigliato in "Zaino".

Identificate queste due azioni fondamentali, si è pensato che l'una debba essere il susseguirsi dell'altra come evidenziato nel workflow visibile in Figura 4.2. L'utente è stato reso inoltre più consapevole delle azioni che andrà ad intraprendere tramite l'utilizzo di apposite caselle di dialogo che si attiveranno quando vuole effettuare un'operazione.

4.1.3 Mockup

Come già detto i mockup permettono di rappresentare l'idea di come sarà l'app. La schermata visibile in Figura 4.3 è la prima che si avvia nel momento in cui l'utente sceglierà dalla MainActivity di creare un nuovo percorso.

Una volta che è termina la registrazione di punti del percorso sarà possibile andarne a memorizzare tutte le informazioni relative tramite un'apposita activity strutturata come in figura 4.4.

4.2 Implementazione

Come detto nell'introduzione la condivisione è un principio cardine, per questo è stato utilizzato un meccanismo di memorizzazione condivisa quale è il database. Altri utilizzi importanti che sono stati fatti nell'implementazione sono state certamente le mappe ma anche gli strumenti di layout, quali l'alert dialogs e di funzionalità proprie del sistema operativo per il rilevamento continuo della posizione.

4.2.1 Struttura

La struttura del progetto rispetto a quella iniziale si è allargata introducendo due classi Kotlin in più (visibili in Figura 3.7):

- **RegNuovo.kt** $\rightarrow$ activity che si occupa del tracciamento del percorso quindi la memorizzazione dell'aggiornamento continuo della posizione
- **NuovoPercorso.kt** $\rightarrow$ activity che si occupa dell'inserimento delle informazioni e della memorizzazione finale sul database

Le interfacce di tali activity sono definite da due corrispondenti file di layout: *nuovo_mappa* e *activity_nuovo_perc*. Inoltre sono state aggiunte altre stringhe sia in italiano che in inglese per i messaggi visualizzati all'interno degli Alert Dialogs.

4.2.2 Alert Dialogs

Le Alert Dialogs, tradotte in Italiano con "Finestre di dialogo", sono delle piccole finestre che si attivano come un pop-up sullo schermo, le quali chiedono un'azione da parte dell'utente per proseguire. Queste azioni possono essere sia delle decisioni da prendere, sia delle informazioni da inserire. Le Alert Dialogs in particolare hanno delle proprietà che si possono definire quali:

- Titolo (non obbligatorio): cioè il titolo della finestra che apparirà
- Messaggio: cioè il contenuto, può essere anche un layout personalizzato
- Pulsanti di azione (da uno a tre): ognuno potrà far riferimento ad un'operazione da eseguire. Nell'applicazione in questione sono stati utilizzati i classici due pulsanti di accettazione o rifiuto, "Si" o "No".

In APPercorsi le finestre di dialogo sono state utilizzate due volte. La prima a seguito del click sul pulsante "Aggiungi Percorso" presente nella MainActivity.kt, per informare l'utente che in caso di affermazione positiva sarebbe iniziata la registrazione del percorso, che consiste nella memorizzazione ad ogni intervallo predefinito della sua posizione. Il primo alert è mostrato in Figura 4.5. In caso di scelta positiva si procede, altrimenti si resta sulla schermata di partenza. La seconda volta in cui è stata utilizzata la finestra di dialogo è quando l'utente dopo aver registrato il percorso cioè il suo avanzamento sulla mappa decide di terminarlo per andarne ad inserire le informazioni relative. Essendo questa un'operazione irreversibile l'utilizzo della finestra è fondamentale. La visualizzazione è rappresentata in Figura 4.6.

4.2.3 Tracciamento

Il tracciamento del percorso come già detto fa riferimento all'activity RegNuovo.kt. Esso si basa sul principio dell'estrapolazione continua della latitudine e longitudine dalla posizione dell'utente. Per permettere ciò, è necessario che lo smartphone sia dotato di sensore GPS e vi sia il consenso dell'utento al permesso della localizzazione. L'utente quindi che vuole inserire un nuovo percorso dovrà attivare la funzionalità "Aggiungi Percorso" la quale mostrerà la mappa con sovrapposta costantemente la propria posizione. Una volta terminato il tragitto, l'utente per finire la registrazione dovrà selezionare il bottone di "Fine Tracciamento". La schermata visualizzata in tale fase è mostrata nella Figura 4.7.

Da qui verranno estratti tutti i punti che andranno a comporre il file di testo, collegato ad ogni percorso che permette nel Fragment2.kt la visualizzazione dello stesso mediante l'unione dei punti. La mappa visualizzata dall'utente viene creata con lo stesso sistema del Fragment2, con l'utilizzo della libreria OSMdroid. Per la registrazione continua della

propria posizione è stata utilizzata la funzionalità di cambio posizione disponibile proprio all'interno del sistema operativo. Per una trattazione più approfondita si consiglia di consultare la documentazione all'indirizzo `https://developer.android.com/training/location`.

Viene mostrato un frammento di codice per l'estrapolazione della latitudine e longitudine. Queste informazioni saranno memorizzate all'interno di un ArrayList.

```
fusedLocationClient = LocationServices.
 getFusedLocationProviderClient(this)
 fusedLocationClient.lastLocation.addOnSuccessListener { location
 : Location? ->
          if (location != null) {
              mCurrentPosition=location
              Log.d("TEST", "last location" + mCurrentPosition)
          }
      }
      createLocationRequest()


      locationCallback = object : LocationCallback() {
          override fun onLocationResult(locationResult:
 LocationResult?) {
              locationResult ?: return
              Log.d("TEST", "results" + locationResult)
              for (location in locationResult.locations){
                  var attuale: String =
                      "${location.latitude},${location.longitude}"
  + "\n"
                  Log.d("TEST", "pos attuale" + attuale)

                  array.add(attuale)
              }
          }
      }
      startLocationUpdates()
```

Nella funzione createLocationRequest() si definiscono tutti i parametri per il prelievo della posizione quali l'intervallo e l'accuratezza. La startLocationUpdates() è definita come di seguito:

```
1   private fun startLocationUpdates() {
2        fusedLocationClient.requestLocationUpdates(locationRequest,
3            locationCallback,
4            Looper.getMainLooper())
5    }
```

dove locationRequest è una variabile globale, inizializzata da CreateLocationRequest().

Al momento del click su "Fine tracciamento" verrà mostrato l'Alert Dialogs il quale chiede se l'utente è certo di voler continuare, in tal caso viene chiamata un'altra activity cioè NuovoPercorso.kt tramite un intent al quale si allegano tramite il metodo *.putExtra()* delle informazioni, quali l'array contenente tutti i punti estrapolati dal tracciamento. Fatto ciò si interrompe anche la funzione dell'aggiornamento della posizione. E'definito nel codice come segue:

```
1   var fine = findViewById<View>(R.id.button) as Button
2        fine.setOnClickListener{
3
4            val alertDialog: AlertDialog? = this.let {
5                val builder = AlertDialog.Builder(it)
6                builder.setMessage(R.string.dialog_fire_missiles)
7                builder.apply {
8                    setPositiveButton(R.string.si,
9                        DialogInterface.OnClickListener { dialog, id
     ->
10                           val vis = Intent(this@RegNuovo,
    NuovoPercorso::class.java)
11                           fusedLocationClient.
    removeLocationUpdates(locationCallback)
12                           vis.putExtra("Percorso",array)
13                           startActivity(vis)
14                       })
15                   setNegativeButton(R.string.no,
```

```
                        DialogInterface.OnClickListener { dialog, id
    ->
                            null })
                }
                 builder.create()
                 builder.show()
            }
        }
```

4.2.4 Inserimento informazioni

L'inserimento informazioni e la memorizzazione finale sul database avvengono a seguito della registrazione del tracciamento. Esse sono implementate nell'activity NuovoPercorso.kt. Qui viene gestita la view corrispondente, visibile in Figura 4.8, la quale mostra i campi che l'utente deve riempire obbligatoriamente, un pulsante per aggiungere la foto del percorso che verrà visualizzata come immagine di preview nella lista dei percorsi e infine un pulsante per il salvataggio del percorso all'interno del database. All'interno del codice, prima vengono definiti i riferimenti al layout, poi sono specificate le azioni che si attivano sul click sui bottoni, le quali richiamano delle funzioni. Il primo bottone che si incontra è "Inserisci Foto", nel codice è così definito all'interno del metodo OnCreate():

```
var addPhoto = findViewById<View>(R.id.btn_scatta) as Button
addPhoto.setOnClickListener() {
  val takePictureIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
  try {
      startActivityForResult(takePictureIntent,
   REQUEST_IMAGE_CAPTURE)
  } catch (e: ActivityNotFoundException) {
      Toast.makeText(this, "Non posso aggiungere", Toast.
   LENGTH_SHORT).show()
  }
}
```

ed è definita anche l'override della OnActiviyResult fuori dal metodo onCreate() in questo modo:

```
override fun onActivityResult(requestCode: Int, resultCode: Int,
    data: Intent?) {
super.onActivityResult(requestCode, resultCode, data)
if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK)
     {
  val imageBitmap = data?.extras?.get("data") as Bitmap
        imageView3.setImageBitmap(imageBitmap)}
}
```

Successivamente è stato definito il bottone ”Aggiungi”, il quale permette la memorizzazione sul database. Al click sul bottone, viene dapprima effettuato il controllo che tutte le caselle di testo siano non vuote, una volta che sono riempite vengono estratte le informazioni del percorso dalle TextInputEditText, poi vengono chiamate in quest'ordine le funzioni:

- saveFile(name, datipassati as ArrayList¡String¿)
- saveStorage(name)
- saveRealtime(name, lung, dur, diff, desc, txtname, za1, za2, za3, za4, za5, za6, za7, za8)

e infine si è rimandati all'activity iniziale tramite:

```
        val inizio = Intent(this, MainActivity::class.java)
        startActivity(inizio)
```

Analizzando le singole funzioni:

saveFile(name, datipassati as ArrayList<String>) è così definita:

```
    private fun saveFile(name: String, datipassati:ArrayList<String
   >){
    val sb = StringBuilder()
    if (datipassati != null) {
        for (s in datipassati) {
            sb.append(s)
        }
        Log.d("TEST", "SB" + "$sb")
```

```
8     }
9     val stream = ByteArrayInputStream(sb.toString().toByteArray(
      charset("UTF-8")))
10    val storageRef = Firebase.storage.reference.child("""$name.txt
      """)
11    var uploadTask = storageRef.putStream(stream)
12    }
```

Vengono estratti dall'ArrayList<String> i punti, poi trasformati in uno stream di byte, al fine di salvarli sullo Storage. Il nome del file sarà pari al nome del percorso con l'estensione "txt".

saveStorage(name: String) viene così definita:

```
1  private fun saveStorage(name: String) {
2    var storage = FirebaseStorage.getInstance()
3    val storageRef = Firebase.storage.reference.child("""$name.jpeg
       """)
4    // Get the data from an ImageView as bytes
5    imageView3.isDrawingCacheEnabled = true
6    imageView3.buildDrawingCache()
7    val bitmap = (imageView3.drawable as BitmapDrawable).bitmap
8    val baos = ByteArrayOutputStream()
9    bitmap.compress(Bitmap.CompressFormat.JPEG, 100, baos)
10   val data = baos.toByteArray()
11   var uploadTask = storageRef.putBytes(data)
12 }
```

Qui l'immagine viene estratta dall'ImageView della schermata dove è riportata la foto precedentemente scattata con "Aggiungi Foto", trasformata in un bitmap, compressa e memorizzata sul db come uno stream di byte. L'immagine nel database viene registrata con il nome del percorso e un estensione "jpeg".

Ultima funzione chiamata dopo aver memorizzato i file sullo storage, è: **saveRealtime(name, lung, dur, diff, desc, txtname, za1, za2, za3, za4, za5, za6, za7, za8)**, definita come:

```
1  private fun saveRealtime(name: String, lung: String, dur: String,
      diff: String, desc:String,txt: String, za1:String , za2:String ,
```

```
    za3:String , za4:String ,za5:String ,za6:String ,za7:String,
   za8:String){
  val dbref1  = FirebaseDatabase.getInstance().getReference("
   Percorsi")
    val  dbref = dbref1.child(name)
dbref.child("Descrizione").setValue(desc)
dbref.child("Difficolta").setValue(diff)
dbref.child("Durata").setValue(dur)
dbref.child("Lunghezza").setValue(lung)
dbref.child("Nome").setValue(name)
dbref.child("Percorso").setValue(txt)
dbref.child("Zaino").child("0").setValue(za1)
dbref.child("Zaino").child("1").setValue(za2)
dbref.child("Zaino").child("2").setValue(za3)
dbref.child("Zaino").child("3").setValue(za4)
dbref.child("Zaino").child("4").setValue(za5)
dbref.child("Zaino").child("5").setValue(za6)
dbref.child("Zaino").child("6").setValue(za7)
dbref.child("Zaino").child("7").setValue(za8)
}
```

Tutte le informazioni passate come parametro alla funzione sono state precedemente estratte, come già detto, tranne ”txt”. Esso corrisponde a ”txtname” che è stato definito come:

```
val txtname="$name"+".txt"
```

dove name è il nome del percorso. Questo valore è inserito sul campo ”Percorso” e permette quindi di collegare il file di testo che rappresenta il tracciato, al percorso relativo. Dopo aver eseguito tutti i passaggi finalmente il percorso è salvato sul database insieme agli elementi che lo caratterizzano. Da adesso in poi esso sarà quindi fruibile dagli altri utenti nella lista dei percorsi, è realizzato così il principio della condivisione.

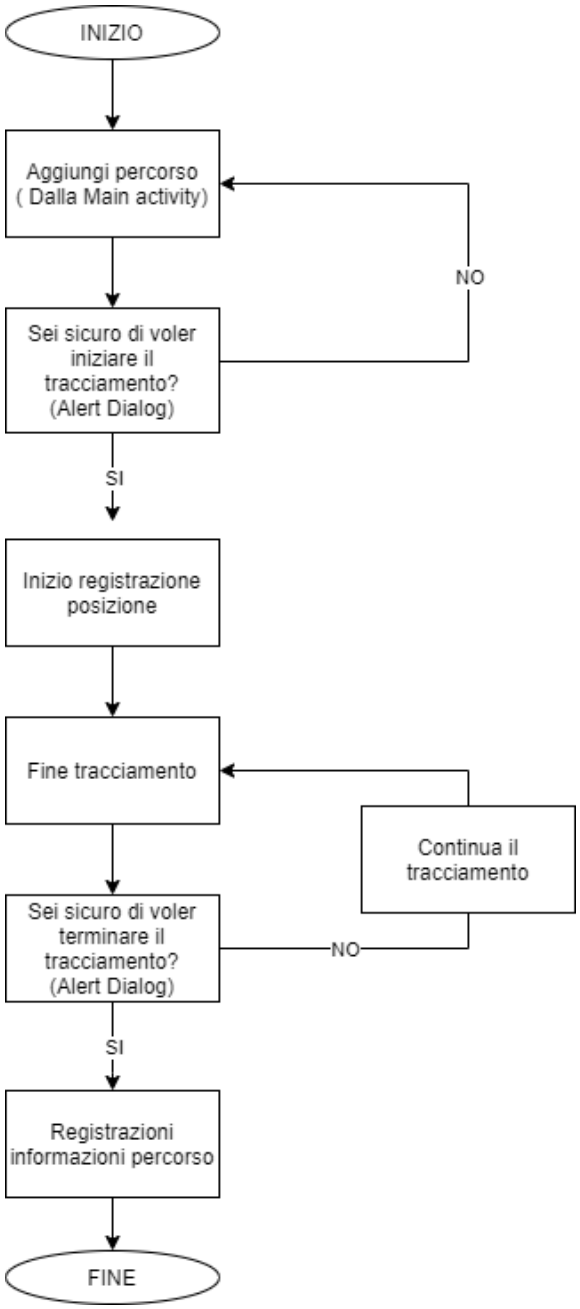


Figura 4.2: Workflow

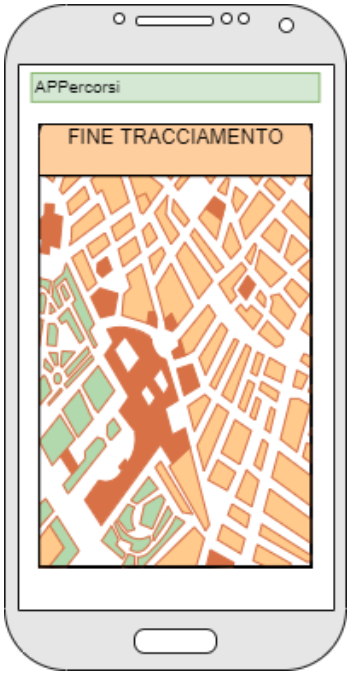


Figura 4.3: Mockup di registrazione del percorso

Figura 4.4: Mockup di registrazione informazioni del percorso

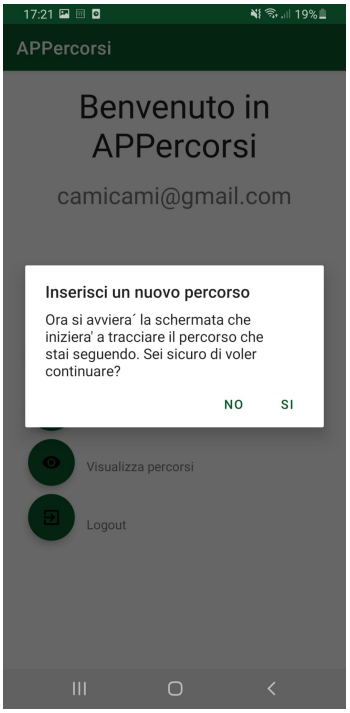


Figura 4.5: Primo Alert Dialog

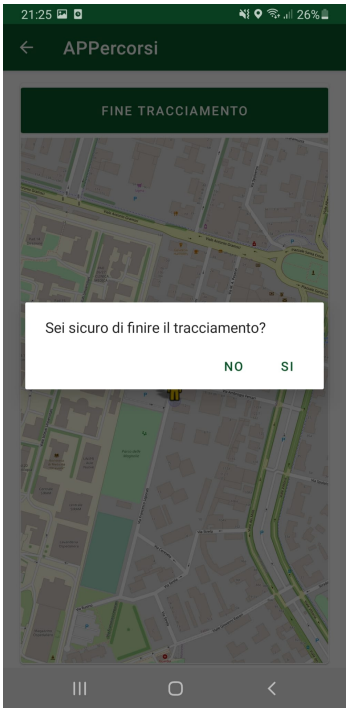


Figura 4.6: Secondo Alert Dialog

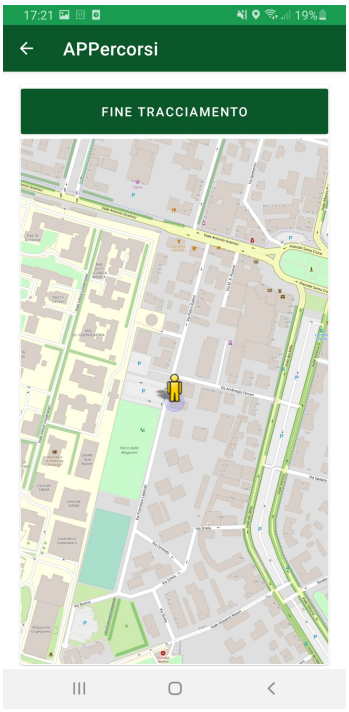
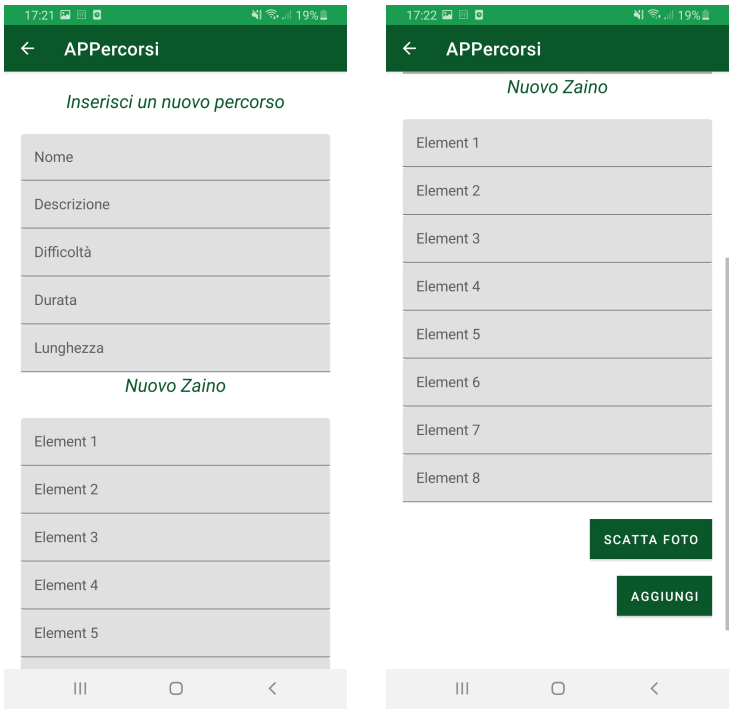


Figura 4.7: Schermata per il tracciamento



(a) Prima parte (b) Seconda parte

Figura 4.8: Schermata per l'aggiunta delle informazioni e il salvataggio del percorso

5. Conclusioni

In questa tesi abbiamo discusso del lavoro che ha permesso di creare un'applicazione in grado di aiutare coloro che sono interessati a seguire i percorsi del *Parco Nazionale del Gran Sasso e Monti della Laga* permettendone anche l'aggiunta di nuovi. Grazie ai punti di interesse sia multimediali che non, risulta essere uno strumento informativo per chiunque voglia esplorare o conoscere qualcosa in più della zona; mentre l'inserimento di nuovi sentieri è stato realizzato per ampliarne la conoscenza permettendo a tutti di usufruirne. Sono state analizzate tutte le fasi del lavoro dalla progettazione allo sviluppo, dopo un'attenta analisi sulle motivazioni che ci han spinto fin qui. L'implementazione è stata realizzata utilizzando le ultime tecnologie quali Kotlin, un linguaggio moderno per lo sviluppo di applicazioni mobile e Firebase, database open-source ad oggi fra i più utilizzati nel mondo web. Ciò ha reso il lavoro stimolante per il programmatore il quale ha dovuto informarsi e documentarsi sia sulle tecnologie che sul caso di studio. Il lavoro svolto finora presenta molte funzionalità sviluppate ma certamente non al massimo delle sue potenzialità, quindi è un'utile base per sviluppi futuri da parte di appassionati o programmatori. Alcune delle possibili future implementazioni sono ad esempio la possibilità per un utente di inserire dei marker nella creazione di un nuovo percorso, rendendo così multimediali anche i percorsi che si vanno ad inserire, oppure la possibilità per l'utente di scegliere se salvare il percorso rendendolo visibile a tutti, o solamente in locale per lui stesso. Si auspica che l'app, oltre ad essere stata una valida esperienza di accrescimento delle competenze informatiche sia nell'ambito della Programmazione Mobile che in quello del generico Problem Solving, miri anche alla diffusione e conoscenza del *Parco Nazionale del Gran Sasso e Monti della Laga* in un contesto il più ampio possibile.

Bibliografia

[1] Developers Android. *Create dynamic lists with RecyclerView*. 2021. URL: https://developer.android.com/guide/topics/ui/layout/recyclerview.

[2] Developers Android. *Meet Android Studio*. 2021. URL: http://www.developer.android.com/studio/intro.

[3] Simone Cosimi. *Esistono 2 milioni di app, perché ne usiamo quattro?* 2019. URL: https://www.esquire.com/it/lifestyle/tecnologia/a26997659/app-piu-usate/.

[4] Flutter. *Homepage Flutter*. 2021. URL: https://flutter.dev/.

[5] Github. *About Glide*. 2021. URL: https://bumptech.github.io/glide/.

[6] GranSassoLagaPark. *I Borghi*. 2021. URL: http://www.gransassolagapark.it/pagina.php?id=50/.

[7] GranSassoLagaPark. *La morfologia del parco*. 2021. URL: http://www.gransassolagapark.it/pagina.php?id=232.

[8] GranSassoLagaPark. *Territorio*. 2021. URL: http://www.gransassolagapark.it/pagina.php?id=38.

[9] Kotlin. *Sviluppa App Android con Kotlin*. 2021. URL: http://www.developer.android.com/kotlin.

[10] Open Street Map. *About*. 2021. URL: https://www.openstreetmap.org/about.

[11] Paolo Marocco. *DISPOSITIVO MOBILE-Enciclopedia italiana*. 2015. URL: https://www.treccani.it/enciclopedia/dispositivo-mobile_%5C%28Enciclopedia-Italiana%5C%29/.

[12] Techopedia. *Applicazione Mobile*. 2020. URL: https://www.techopedia.com/definition/2953/mobile-application-mobile-app.

[13] Luca Tremolada. *Android vs iOs: chi controlla il mercato dei sistemi operativi mobili?* 2018. URL: https://www.infodata.ilsole24ore.com/2018/07/18/android-vs-ios-controlla-mercato-dei-sistemi-operativi-mobili/.