UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DIPARTIMENTO DI INFORMATICA

Corso di Laurea in Informatica

Tesi di Laurea in Modelli e Metodi per la sicurezza delle applicazioni

SafeKids: Progettazione e sviluppo di un'applicazione Android per il Parental Control

Relatore:
Prof. Donato Impedovo

Laureando:
Giacomo Pagliara

ANNO ACCADEMICO 2021/22

A mia madre, a mio padre e a mio fratello, a cui devo tutto.

Abstract

Quando si parla di Parental Control si fa subito riferimento alla protezione e al monitoraggio dei bambini online da parte dei genitori. La protezione dei bambini include anche il monitoraggio della loro posizione per evitare situazioni di pericolo. Altresì, anche bambini con disturbi dello spettro autistico o persone con demenza possono imbattersi in situazioni di vagabondaggio o di pericolo. Le problematiche a cui vanno incontro sono molteplici e influiscono sulla qualità della loro vita. Per contribuire a gestire il fenomeno del vagabondaggio all'aperto di persone con o senza demenza e di bambini con o senza disturbi dello spettro autistico o per evitare che frequentino luoghi pericolosi o siano vittime di bullismo, questa tesi propone un'applicazione Android, per il monitoraggio della posizione del bambino basato sulla localizzazione in tempo reale di quest'ultimi, con la creazione di una zona sicura "safe zone".

La comunità scientifica e di ricerca ha proposto diverse tecnologie da cui ho preso spunto per la creazione della mia applicazione e soprattutto per creare il modello di rilevamento della posizione.

Dopo aver sviluppato un metodo per poter attuare il rilevamento della posizione, ho incluso questo metodo all'interno di un'applicazione Android di nome SafeKids. Al giorno d'oggi gli smartphone, che frequentemente usiamo, includono numerose tecnologie e sensori che si possono usare per facilitare il rilevamento della posizione. L'applicazione è rivolta ai genitori che vogliono controllare gli spostamenti dei propri figli. Il mio obiettivo, fin da subito, è stato quello di rendere accessibile a tutti questo strumento assistivo che aiuta parenti, tutori e chiunque affronti giornalmente determinate situazioni a causa di diverse patologie, con il fine di gestire situazioni di pericolo e garantire sicurezza.

Ringraziamenti

Con questa tesi si conclude un capitolo importante della mia vita, ricco di emozioni e soddisfazioni.

Un ringraziamento speciale va ai miei genitori, a cui devo tutto, che hanno reso possibile tutto questo, per avermi supportato sempre, per avermi guidato in ogni percorso e soprattutto per aver creduto in me fin dall'inizio, sostenendo le mie scelte e dandomi preziosi consigli che porterò sempre con me.

A mio fratello, che mi ha sempre spronato ad andare avanti nei momenti belli e nei momenti più stressanti.

A Giorgia, che mi ha sempre sostenuto e mi ha dato la forza di andare avanti anche nei momenti più bui, che ha saputo capire le mie paure e mi ha sempre incoraggiato.

Oltre alle emozioni e alle soddisfazioni, ci sono stati anche momenti più stressanti in cui ho creduto di non farcela, proprio per questo un ringraziamento va anche a me stesso per non aver mollato mai.

Ringrazio, infine, il professore Donato Impedovo, mio relatore, per avermi accompagnato in questo percorso.

Indice

Sommario

“Coloro che possono immaginare qualsiasi cosa, possono creare l’impossibile”

-Alan Turing

Capitolo 1 – Introduzione

Per poter diminuire le ansie e le preoccupazioni delle persone soprattutto dei genitori, oggi esistono numerose tecnologie assistive sia di tipo hardware che di tipo software che possono essere utilizzate. Principalmente entrerò nel dettaglio, successivamente, sulle metodologie e sulle applicazioni esistenti sul mercato per gestire il fenomeno del vagabondaggio, da cui ho preso spunto per poter sviluppare la mia applicazione Android chiamata SafeKids. In questi capitoli faccio riferimenti prima alla creazione della parola "vagabondaggio", spiegando da dove è nato il termine.

1.1 Cos'è il vagabondaggio nel Parental Control

Prima di parlare del vagabondaggio da parte dei bambini, è importante dare una panoramica sul significato della parola "vagabondaggio" e come è nata. Il vagabondaggio viene identificato come un vero e proprio sintomo di chi soffre di Alzheimer o della demenza più in generale. Consiste in una pulsione verso il vagare, lo spostarsi in direzione di qualcosa, con la speranza di arrivare in un luogo amato. Nella maggior parte dei casi questa pulsione genera grande stanchezza fisica ma anche d'animo che costringe la persona a rimandare questa pulsione al giorno successivo o nei giorni a seguire. Questi spostamenti potrebbero sembrare senza alcuno scopo, ma non è così. In realtà, ha quasi sempre uno scopo oppure è dettato da una particolare condizione ma facilmente il malato dimentica dove sta andando, che cosa si proponeva di fare, oppure non è in grado di spiegarlo.

Il wandering viene quindi definito come una sindrome di comportamento locomotorio correlato alla demenza avente una natura frequente, ripetitiva, disordinata nel tempo che si manifesta in modalità sovrapposte e casuali, alcuni dei quali sono associati a fughe, tentativi di fuga o perdersi se non accompagnati [1].

Il vagabondaggio è comune anche tra i bambini con disturbi dello spettro autistico (ASD). In un certo senso, si potrebbe pensare a un comportamento del tutto normale per un bambino, ma la situazione cambia superati i 4 anni di età. Secondo l'articolo[1] "Autism and Wandering – The tendency of children on the spectrum to wander off impulsively is a huge safety issue for parents" in uno studio del 2011 sul vagabondaggio che ha suscitato risposte da più di 800 genitori, si diceva che circa il 50% dei bambini di età compresa tra 4 e 10 anni con un ASD avesse vagato a un certo punto, quattro volte di più rispetto ai loro fratelli non affetti.

Il vagabondaggio è significativamente più comune tra i bambini con ASD e quelli con problemi comportamentali e di sviluppo rispetto ad altri bambini [2].

[1] https://childmind.org/article/autism-plus-wandering/

1.2 Gestione del vagabondaggio nel Parental Control

Ora bisogna capire come gestire questa problematica. Esistono alcune azioni fondamentali che dovrebbero essere tenute in considerazione per poter gestire al meglio il vagabondaggio come spiegato nella pagina di "Alzheimer Italia"[2] ovvero: evitare gli atteggiamenti di sfida, la violenza, i rimproveri e cercare di distrarre il malato; non lasciarsi prendere dal panico e contattare la polizia se necessario ed evitare il ricorso ai farmaci che in alcuni casi potrebbero creare maggior irrequietezza.

Con lo sviluppo delle tecnologie, al giorno d'oggi sono tanti i dispositivi che possono essere utilizzati per poter controllare in tempo reale gli spostamenti da parte di persone che a causa di patologie in alcuni casi sono dediti al vagabondaggio. Oltre ai dispositivi hardware utilizzati per rilevare gli spostamenti, esistono metodi molto efficienti presenti sul mercato che potenziano ancor di più l'azione di rilevamento del vagabondaggio. Questi strumenti, per lo più applicazioni, utilizzano il GPS presente all'interno degli smartphone per rilevare il vagabondaggio. Alcune applicazioni permettono di rilevare ogni spostamento della persona assistita e sulla base degli spostamenti compiuti capire se si tratta di vagabondaggio oppure no, inviando una notifica sul telefono dell'assistito indicandoli che potrebbe essersi smarrito e mostrare quindi la direzione di casa propria, oppure una volta rilevato il vagabondaggio effettuare in maniera automatica una chiamata di SOS a un parente, tutor o caregiver ecc. Altri metodi/applicazioni permettono di poter impostare varie zone sicure sulla mappa, se la persona in questione dovesse uscire da queste zone sicure verrebbe inviata automaticamente sul telefono del parente, del tutor o del caregiver, una notifica di pericolo in modo tale da mettere in allerta il tutore. I metodi presenti sul mercato sono tanti, all'interno del capitolo 2 ho trattato i metodi che secondo me sono più efficienti al giorno d'oggi e da cui la mia applicazione trae spunto.

1.3 La soluzione proposta

Come ho scritto poc'anzi, prendendo spunto dai metodi più efficienti che si trovano all'interno del mercato al giorno d'oggi, con questa tesi voglio proporre la mia soluzione per monitorare la posizione attraverso un'applicazione Android, che cerca di mettere insieme tecniche e metodologie efficienti e precise, chiamata SafeKids.

La mia applicazione è composta da quattro moduli software. Una prima parte relativa all'autenticazione degli utenti con fase di registrazione e di accesso, una seconda parte di associazione fra due smartphone, una terza parte relativa al rilevamento della posizione con la creazione di una zona sicura in corrispondenza della posizione del tutore, del parente o caregiver. Grazie al GPS presente negli smartphone è stato possibile implementare questo modulo software. In caso di fuoriuscita dalla zona sicura viene inviata una notifica al tutore per gestire un eventuale situazione di pericolo. E infine il quarto modulo che si occupa della creazione, del salvataggio e della cancellazione dei contatti di emergenza che possono essere chiamati sia dal tutore che dall'assistito in caso di pericolo.

[2] https://www.alzheimer.it/wander.html

Capitolo 2 – Stato dell'arte

Lo stato dell'arte che discuto in questo capitolo riguarda lo sviluppo di soluzioni che riguardano il rilevamento del vagabondaggio all'aperto e quindi non al chiuso. Il mio obiettivo, fin da subito, è stato quello di poter aiutare genitori e tutori che sono alle prese con problemi di questo tipo. La mia applicazione, infatti, offre una soluzione di monitoraggio della posizione all'aperto di bambini con o senza disturbi dello spettro autistico e persone con demenza e non.

Prima di iniziare a sviluppare la mia applicazione, ho consultato e studiato varie riviste scientifiche che proponevano soluzioni molto efficaci per poter aiutare le persone che affrontano situazioni di vagabondaggio e soprattutto poter aiutare i propri cari.

Il primo progetto che ho analizzato, da cui è nata la voglia di creare un'applicazione per monitorare costantemente la posizione, è un sistema di monitoraggio del vagabondaggio basato sul flusso(stream).

2.1 Sistema di monitoraggio del vagabondaggio basato sul flusso(stream)

All'interno dello studio condotto da W. Sansrimahachai [3], viene proposto un sistema che applica una tecnica di elaborazione del flusso insieme alla tecnologia GPS per identificare la posizione e rilevare il comportamento errante degli anziani in tempo reale.

La Figura 1 mostra una chiara panoramica del sistema proposto, in particolare mostra l'architettura del sistema di monitoraggio basato sul flusso. Un middleware orientato ai messaggi (MOM) svolge un ruolo centrale in questa architettura. Fornisce un'infrastruttura che supporta l'invio e la ricezione di flussi di dati tra il server di elaborazione dei flussi e i relativi client. Prima di tutto, il flusso di elaborazione del sistema prende i dati di geolocalizzazione che vengono immessi nel sistema di monitoraggio del wandering da un GPS presente all'interno di un telefono cellulare o qualsiasi dispositivo dotato di geolocalizzazione. Ogni evento di flusso viene ricevuto da MOM e subito indirizzato al servizio per il rilevamento in tempo reale del wandering (WDS). Il WDS, ovvero il sistema di rilevamento del wandering, viene eseguito continuamento in modo tale da individuare eventuali situazioni di vagabondaggio. Quando il WDS rileva una situazione di vagabondaggio invia una notifica a un'applicazione client di gestione del vagabondaggio basata sul web utilizzata dagli operatori sanitari, parenti, tutor ecc. Quest'ultimi, che gestiscono l'applicazione client, possono fare delle interrogazioni o più precisamente "query" per ottenere informazioni da un archivio di eventi che conterrà ad esempio i vari spostamenti e quindi lo storico degli spostamenti effettuati dall'assistito.

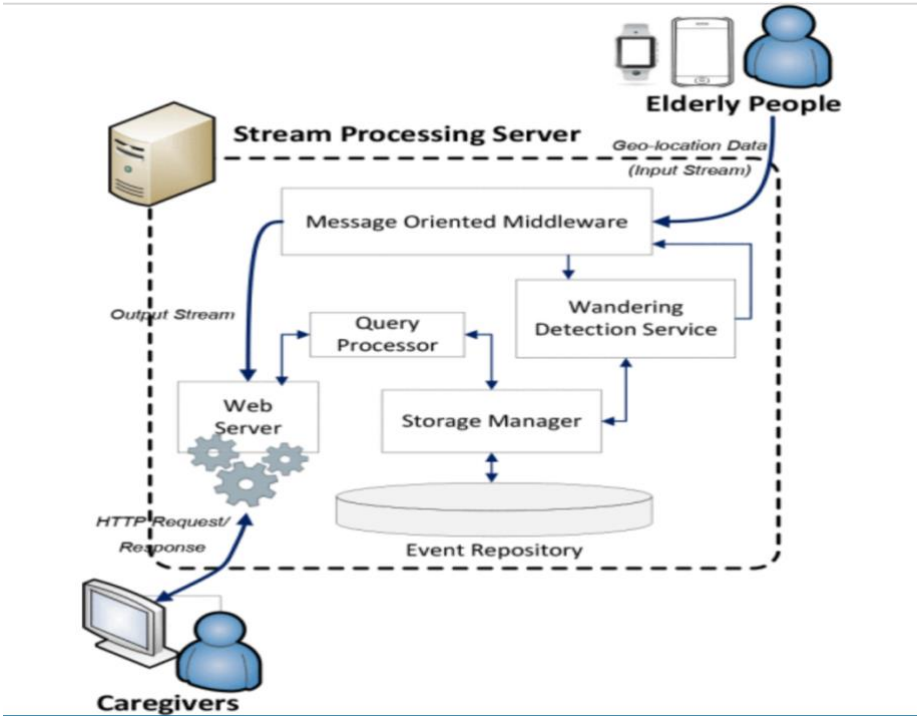


Figura 1-L'architettura del sistema proposto per ili rilevamento del wandering [3]

Il WDS corrisponde al metodo per rilevare il vagabondaggio in tempo reale nell'ambiente esterno. Sulla base del modello di Martino-Saltzman [4] vengono identificati i loop in una traccia GPS in modo tale da poter rilevare il comportamento di vagabondaggio basato sulle traiettorie di mobilità degli anziani [4] [3].

Il WDS [3], viene suddiviso in tre fasi principali: la preelaborazione dei dati GPS, l'identificazione del loop e la generazione di notifiche. Nella prima fase, per ogni punto GPS viene calcolata la distanza tra sé stesso e il suo predecessore. La distanza dal passaggio precedente viene utilizzata per filtrare i punti GPS rumorosi utilizzando l'algoritmo di soglia della distanza. Questo algoritmo scarta ogni punto GPS che non sia almeno a un valore *x* di distanza dall'ultimo punto non scartato. Fatto questo, i punti GPS, che sono stati filtrati, vengono aggiunti a una finestra di distanza scorrevole. Quest'ultima consente al metodo proposto di acquisire una serie di punti GPS per una data distanza D (una distanza minima di viaggio degli anziani utilizzata per calcolare il rilevamento del vagabondaggio) e creare una traccia GPS per un'ulteriore elaborazione. Infatti, la traccia GPS, costituita da un insieme di punti GPS generata dalla finestra di distanza viene quindi semplificata per rendere più precisa la fase di elaborazione, ovvero l'identificazione del loop. Nella seconda fase, il rilevamento del wandering viene affidato all'algoritmo di intersezione del segmento di linea. Questo algoritmo prende ogni traccia GPS dalla fase precedente, ovvero la fase di preelaborazione dei dati, come input per identificare il loop nella traccia GPS. Infine, nella fase finale, se vengono rilevati uno o più loop, il metodo proposto di rilevamento del vagabondaggio invierà automaticamente una notifica istantanea agli assistenti o ai familiari degli assistiti affinché vengano prese azioni di tutela.

Un'altra parte importante del sistema di monitoraggio del wandering proposto, è l'applicazione client di gestione del wandering basata sul Web che consente agli utenti (parenti, operatori sanitari,

tutor e caregiver) di tracciare in tempo reale la posizione degli assistiti. Oltre a questo l'applicazione web proposta offre altre funzionalità. Ho deciso di analizzare solo tre funzionalità su cinque proposte da Sansrimahachai, perché ritenute più in linea con i miei obiettivi di tesi. La prima funzionalità offerta è visualizzare la posizione corrente di ciascun assistito sulla mappa in tempo reale. La seconda funzionalità offerta permette di definire una zona sicura o una zona pericolosa, come si vede nella Figura 2, per gli assistiti sulla mappa e invia notifiche istantanee nel caso in cui ogni assistito entri o esca dalle zone definite. L'ultima funzionalità delle tre proposte è rilevare un'occorrenza di comportamento errante in tempo reale in base al metodo di rilevamento del wandering descritto in precedenza e visualizzare i dettagli del comportamento errante per ciascun assistito.

I risultati sperimentali hanno dimostrato che il sistema proposto consente di affrontare il problema di rilevamento del wandering in tempo reale con prestazioni accettabili e spese generali ragionevoli. La latenza media, circa 1 ms per paziente/assistito in più, è stata osservata come l'impatto del rilevamento del vagabondaggio sulle prestazioni del sistema che è relativamente basso [3].

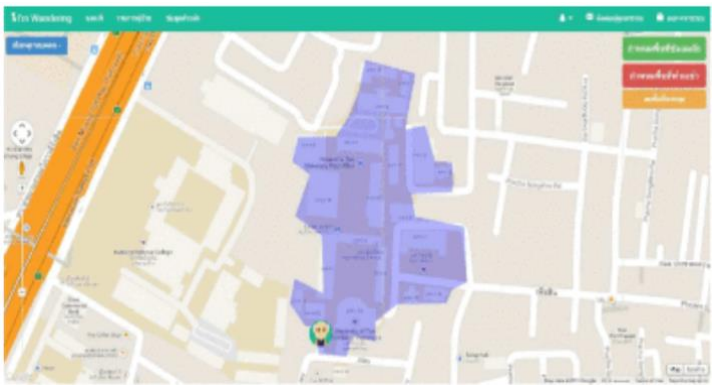


Figura 2 - Funzione per impostare la zona sicura o pericolosa [3]

2.2 Sistemi di monitoraggio del vagabondaggio basati su zone sicure o geofence

In questa parte analizzo i vari sistemi di rilevamento del wandering basati su zone sicure o geofence (recinti geografici) da cui ho preso spunto per sviluppare il mio progetto di tesi.

L'idea che sta alla base è quella di creare recinti geografici virtuali che corrispondono a vere e proprie zone sicure, che permettono ai parenti, agli operatori sanitari ecc. di poter controllare se i propri assistiti escono o meno da questa o queste zone sicure. Principalmente queste "zone sicure" corrispondono a dei veri e propri cerchi dove il raggio di quest'ultimi può essere impostato dai tutori oppure viene già impostato di default, sulla residenza dell'assistito.

Il primo progetto che discuto in questa sezione è stato fondamentale per arricchire il mio obiettivo iniziale di tesi e mi ha dato ulteriori spunti importanti per rendere efficiente lo sviluppo e la modifica della mia applicazione.

2.2.1 Tecnologia di geofencing nel monitoraggio degli assistiti

All'interno dello studio condotto da E. R. Pratama et al. [5], viene proposto una tecnologia di Geofencing per tracciare e monitorare gli assistiti. Con questa tecnologia, nel sistema proposto, vengono creati confini nell'area dove si trova l'assistito e utilizzando la tecnologia GPS, il segnale del dispositivo cellulare dell'assistito può essere tracciato o monitorato. Nel momento in cui l'assistito esce dalla zona sicura di geofencing viene inviata una notifica al tutore e al tempo stesso viene calcolata la distanza più vicina tra assistito e tutore per poter intervenire in tempo.

La Figura 3 mostra l'architettura del sistema proposto. Quest'ultimo utilizza la tecnologia Geofencing come area per monitorare il movimento dei pazienti in modo virtuale, dove la posizione è il punto centrale dell'area Geofencing che viene utilizzata come area di sorveglianza e utilizza la tecnologia GPS come strumento per tracciare la posizione dell'assistito nell'area Geofencing. Il supervisore riceverà una notifica automatica inviata al server dall'FCM. Il sistema proposto utilizza un database in tempo reale fornito da Firebase come server cloud. I dati di localizzazione provenienti dal dispositivo GPS applicato all'assistito vengono aggiornati ogni 5 secondi e inseriti nel database in tempo reale; i dati di localizzazione vengono utilizzati come posizione del paziente sul sistema e allo stesso tempo vengono visualizzati anche gli spostamenti. La notifica si attiva automaticamente quando l'assistito supera un'area di Geofencing predeterminata; quando il segnale GPS dell'assistito attraversa l'area di Geofencing ma non la supera, il server FCM invia un messaggio o notifica a scopo informativo al tutore che sceglierà se e come intervenire. Il server FCM invia la notifica in base al token o ai token dei dispositivi presenti nel server. Infine, Il sistema proposto calcola la distanza più vicina tra tutor e assistito utilizzando la formula di Haversine, che può determinare la distanza tra due oggetti.

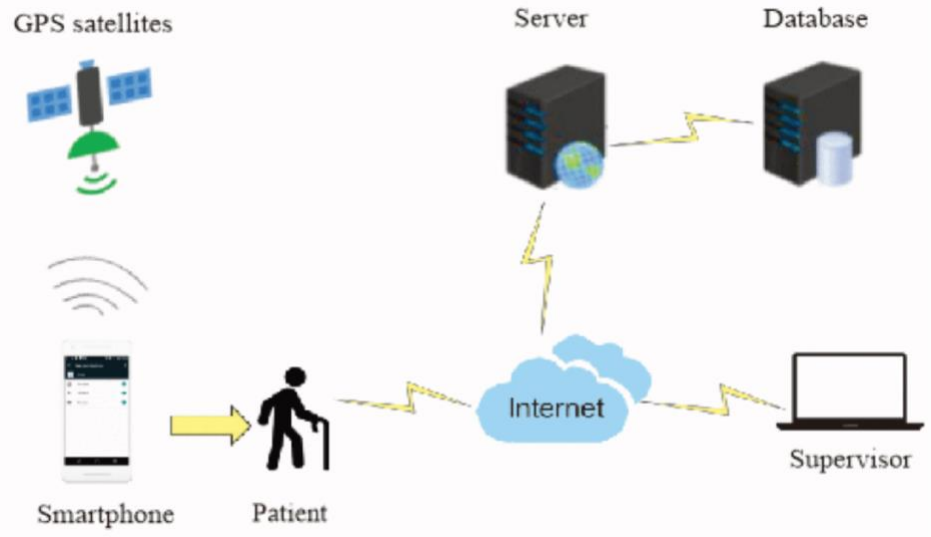


Figura 3 - Architettura del sistema proposto [5]

All’interno del sistema proposto, la rappresentazione dell’area di Geofencing nella mappa è fatta sotto forma di cerchio, dove la rappresentazione della forma del cerchio è un oggetto sulla mappa che ha coordinate di latitudine e longitudine. Per rilevare la zona di Geofencing viene presa in considerazione la posizione dell’assistito, infatti la zona viene impostata sulla posizione dell’assistito. L’area del Geofencing può essere modificata per essere più ampia o più piccola e il colore può essere cambiato.

Firebase Cloud Messagging (FCM) viene utilizzato come si vede in Figura 4, all’interno del sistema proposto, per inviare una notifica o un avviso, al tutore quando il paziente esce dall’area di Geofencing. La notifica viene inviata tramite i token dei dispositivi dei tutor presenti nel database di Firebase.

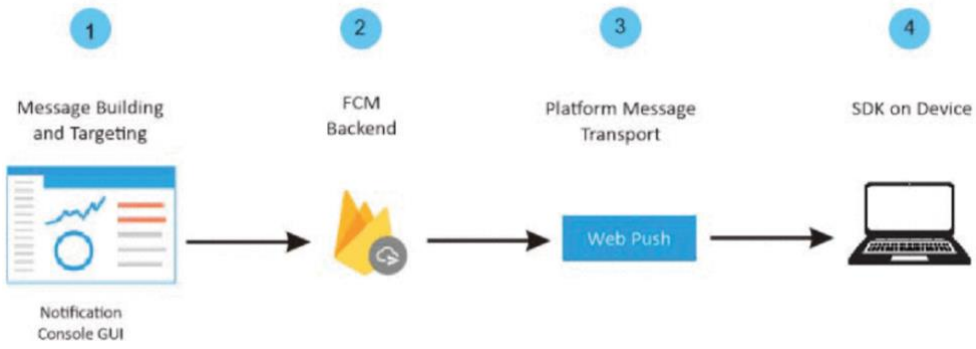


Figura 4 - Architettura FCM [5]

Sempre all’interno del sistema proposto, il metodo di Haversine viene utilizzato per calcolare la distanza più vicina tra due posizioni dell’oggetto prese in base a latitudine e longitudine come variabili di input. Nel sistema proposto i due oggetti sono la posizione dell’assistito e la posizione del tutor.

La formula di Haversine, così come spiegato da M. Cs. Arfiani et al. [6], serve a determinare la distanza tra due punti calcolando il grado di curvatura della terra [6].

Nel sistema proposto [5], il calcolo utilizzando Haversine può essere eseguito solo su due oggetti, quindi il calcolo viene eseguito in più fasi per eseguire calcoli per ciascun tutore. La Figura 5 mostra il calcolo della distanza tra l’assistito e il tutore.

I risultati sperimentali mostrano come, utilizzando la formula di Haversine per determinare la distanza minima tra l’assistito e il tutore, si riesce a velocizzare la gestione di un’eventuale situazione di pericolo da parte del tutore. Altrettanto, FCM è essenziale nel sistema perché il tutore può ricevere rapidamente notifiche quando l’assistito esce dall’area di Geofencing. Nel sistema proposto, c’è un punto debole in termini di accuratezza nel rilevare la posizione dell’assistito, perché il dispositivo mobile nel rilevare il segnale di GPS non ha un livello di accuratezza elevato. Se l’assistito non dovesse portare con sé il dispositivo mobile, il tutore dovrebbe controllare dove si trova l’assistito quando il dispositivo mobile dell’assistito non si muove per circa 30 minuti. È uno

svantaggio perché il sistema non monitorerà i movimenti dell'assistito. Infine, anche se l'uso dei dispositivi mobili ha dei punti deboli, ci sono dei vantaggi, in cui l'utilizzo di un dispositivo mobile può risparmiare sulle spese [5].

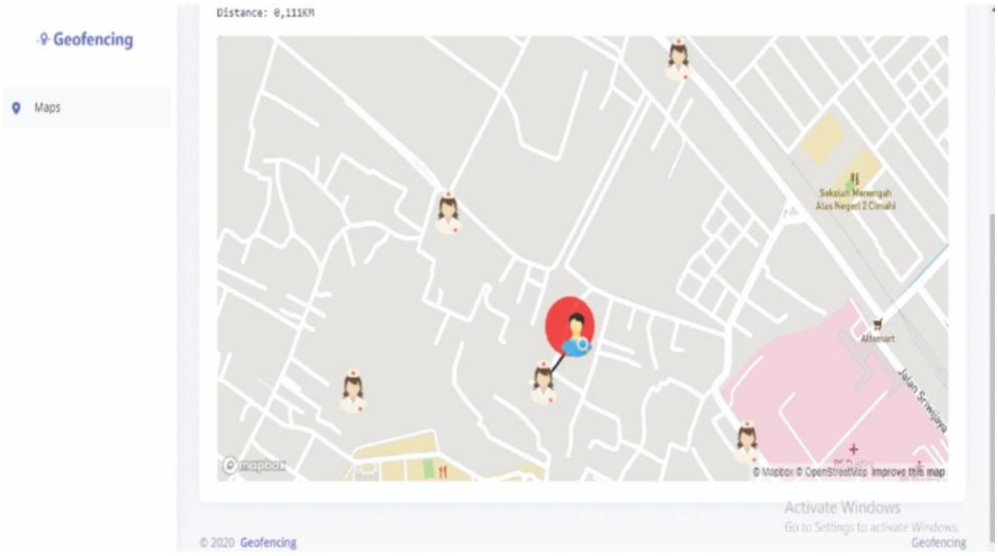


Figura 5 - Distanza tra assistito e tutore [5]

2.2.2 Find My Kids

Una delle applicazioni più famosa e più concorrente sul mercato, è sicuramente Find My Kids.

Destinata principalmente alle famiglie, Find My Kids è una delle applicazioni più rilevanti presenti ad oggi sul mercato per gestire e contribuire al fenomeno del wandering tra bambini con o senza disturbi dello spettro autistico.

L'articolo *"Find My Kids, il tracker GPS per le famiglie"* [7], presente all'interno del blog/sito web applicazioni.it[3] offre una panoramica dettagliata sull'applicazione Find My Kids.

All'interno dell'articolo, viene spiegato come l'applicazione Find My Kids arriva in aiuto alle famiglie fungendo da tracker GPS, offrendo la possibilità di attuare anche un sistema di parental control per la sicurezza dei bambini. La posizione di quest'ultimi e i loro movimenti vengono costantemente monitorati e tracciati in tempo reale, riducendo ansia e preoccupazioni dei genitori.

Per evitare un controllo maniacale che potrebbe scaturire utilizzando l'applicazione, Find My Kids può non voler essere usata dai figli, infatti quest'ultimi devono dare il loro esplicito consenso.

L'applicazione Find My Kids, come spiegato nell'articolo, prevede vari strumenti utili a verificare lo stato di sicurezza dei figli. Sono presenti otto funzionalità, ma nel mio lavoro di tesi ho preso spunto solo da alcune di queste, ritenute da me più importanti per il mio obiettivo. In particolare, la

[3] https://www.applicazioni.it/

prima funzionalità e la più importante è la localizzazione GPS che permette di visualizzare la posizione del bambino e i suoi spostamenti nel corso della giornata. La seconda funzionalità è la possibilità di ascoltare i luoghi circostanti, ovvero ascoltare cosa succede attorno al bambino per capire se sta bene e chi è con lui. Funzionalità molto utile perché permette di capire se il bambino è in buone mani oppure se si trova in una situazione di pericolo a causa di atti di bullismo o altro, oppure anche per capire se viene seguito attentamente. La terza funzionalità, offre la possibilità di inviare un allarme costituito da notifica e suono sul dispositivo del bambino, in modo tale da attirare la sua attenzione nel momento in cui dimentica il telefono nello zaino oppure non risponde alle chiamate. Infine, la quarta funzionalità consiste in un segnale di SOS inviato dal bambino al telefono del genitore. Questo allarme viene inviato dal bambino quando si trova in una situazione di pericolo e non riesce ad effettuare una chiamata. Infatti premendo il segnale di SOS verrà inviata e verrà segnata sulla mappa del genitore la posizione del bambino. Inoltre, partirà una registrazione per poter registrare tutto quello che accade attorno al bambino.

La Figura 6 mostra le fasi da attuare per configurare l'applicazione Find My Kids. La prima fase consiste nello scaricare l'applicazione dai vari store di riferimento, successivamente si può scegliere quale dispositivo del bambino connettere (telefono o orologio GPS). Se il bambino ha un cellulare ovvero un dispositivo mobile dotato di GPS, l'applicazione funziona in modalità localizzatore GPS. Se si vuole avere funzionalità in più e una maggiore efficienza dell'applicazione dovrà essere scaricata l'applicazione complementare chiamata "Pingo: chat with parents" che permette di restare in contatto con il bambino e utilizzare la funzionalità per il segnale di SOS. Per evitare di scaricare quest'applicazione complementare si dovrebbe acquistare un orologio GPS. L'ultima fase consiste nell'autorizzare il controllo del bambino all'interno dell'applicazione sul suo dispositivo [7].

Figura 6 - Find My Kids, come iniziare ad usare l'app[4] [7]

Prima di iniziare a sviluppare la mia applicazione e prima di iniziare a redigere la mia tesi, ho avuto la possibilità di testare l'applicazione Find My Kids.

Prima di tutto, l'applicazione permette, come ho scritto poc'anzi, di rilevare in tempo reale la posizione del bambino, o di qualsiasi persona, collegando il telefono del bambino con il telefono del genitore tramite un codice univoco fornito dall'applicazione stessa. Dal telefono del bambino

[4] https://findmykids.org/it/

dovranno essere impostate una serie di impostazioni fondamentali affinché possa avere senso utilizzare l'applicazione (ad esempio: attivare la geolocalizzazione, attivare l'accesso continuo al microfono, ecc.), queste impostazioni devono essere impostate prima di partire con l'utilizzo dell'applicazione, un esempio di zona sicura viene mostrato in Figura 7. Infatti, per persone affette da demenza o Alzheimer dovrebbe essere impostata l'intera zona di residenza dell'assistito in modo tale da monitorare quest'ultimo in caso di eventuali situazioni di pericolo. Si può inserire il luogo della propria residenza, ovvero una sorta di area sicura tramite un cerchio regolabile in modo tale da designare l'intera area. Compiendo vari test, ho notato che quando l'assistito si sposta la sua icona inizia a muoversi in tempo reale. La precisione della localizzazione non è perfetta, ciò dipende anche dal sistema GPS implementato nei vari dispositivi e anche dalla connessione internet che si ha. L'applicazione, non appena l'assistito esce dall'area sicura, invia una notifica al telefono del tutore con scritto *"Tizio ha lasciato luogo CASA",* CASA in questo caso è la zona sicura precedentemente impostata. Possono essere create più zone sicure, in questo caso il genitore può scegliere, ad esempio, di impostare anche la scuola del bambino come zona sicura. Quando il bambino, ad esempio, entra nella zona sicura SCUOLA, la notifica sul telefono del genitore è del tipo *"Tizio è arrivato a SCUOLA",* un esempio di notifica viene mostrato in Figura 7. Un aspetto particolare presente all'interno dell'applicazione è fornire l'informazione dei km/h ovvero a che velocità l'assistito si sta spostando e quindi differenziare se si sta spostando a piedi o con macchina o qualsiasi altro mezzo.

Per ascoltare cosa succede nell'ambiente circostante, l'applicazione mette a disposizione 4 minuti totali di ascolto, espandibili acquistando la versione Plus dell'applicazione. Durante il test da me condotto, ho notato una qualità audio davvero ottima di ascolto.

Quando l'assistito non risponde al telefono si può inviare un segnale sonoro di SOS per attirare l'attenzione del soggetto interessato. Durante il test, ho notato che il segnale sonoro è molto forte e consente di richiamare l'attenzione dell'assistito in maniera rapida ed efficace.

In conclusione, l'applicazione Find My Kids è un buon prodotto se l'obiettivo è controllare i propri assistiti ed evitare situazioni di vagabondaggio. L'accuratezza dell'applicazione è molto alta, soprattutto nell'utilizzo delle funzionalità proposte. Il difetto, che ho potuto notare conducendo i miei test, è la mancanza di una funzionalità background che permetta di controllare l'assistito anche quando, non accorgendosene, dovesse chiudere definitivamente l'applicazione. Per evitare una situazione del genere si potrebbe inviare una notifica sul telefono dell'assistito per informargli di questa situazione.

Figure 7 - Monitoraggio degli spostamenti e notifiche[4]

2.2.3 iWander

iWander, come spiegato da F. Sposaro et al. [8] è un'applicazione progettata per alleviare in parte lo stress, l'onere finanziario e offrire un monitoraggio remoto più semplice ai tutori, utilizzando il social network dell'utente come servizio di monitoraggio. Al giorno d'oggi con lo sviluppo delle tecnologie, la maggior parte delle persone posseggono uno smartphone che porta con sé all'aperto, l'applicazione sfrutta questa potenzialità fondamentale. È un'applicazione che funziona su qualsiasi dispositivo Android. L'applicazione proposta, viene eseguita in background e raccoglie dati dai sensori del dispositivo, come il GPS, l'ora del giorno, le condizioni atmosferiche, lo stato di demenza e il feedback dell'utente. Questi dati vengono valutati in seguito attraverso una rete bayesiana per determinare la probabilità che la persona stia vagando. L'applicazione proposta, intraprende automaticamente azioni che aiutano a guidare il paziente verso un luogo sicuro, ad avvisare gli operatori e soprattutto a fornire la posizione attuale del paziente oppure a chiamare le forze dell'ordine.

L'applicazione proposta, parte sempre da un principio chiave ovvero designare sulla mappa zone sicure, che possono essere luoghi domestici o esterni, dove l'assistito è al sicuro dai potenziali danni del vagabondaggio. Inizialmente, l'applicazione definisce una prima zona sicura che combacia con il luogo di residenza dell'assistito. La zona sicura viene rappresentata attraverso un cerchio la cui dimensione può essere regolata, in genere l'applicazione modifica automaticamente la zona sicura quando l'assistito si trova all'interno di una determinata zona sulla mappa per lunghi periodi di tempo.

Quando l'assistito esce al di fuori delle zone sicure l'applicazione viene attivata informando il tutore oppure aiutando l'assistito a trovare la direzione giusta e la probabilità di vagabondaggio viene determinata utilizzando tecniche di rete bayesiana. Più sono rilevanti le informazioni maggiore sarà il grado di certezza della rete bayesiana in questione. Proprio per questo, le variabili per la rete bayesiana sono scelte con cura perché sono direttamente proporzionali alla stima della probabilità. All'interno della rete bayesiana vengono immesse, oltre all'età e al livello di demenza dell'assistito, anche l'ora del giorno, le condizioni metereologiche attuali e il meteo. Tutte queste variabili sono importanti per l'applicazione proposta perché aiutano a classificare in maniera efficiente il comportamento come normale o anomalo. Il comportamento normale corrisponde al

fatto che l'assistito non si trova in una situazione di pericolo dovuta al vagabondaggio mentre un comportamento anomalo suggerisce il rischio di pericolo dovuto al vagabondaggio.

All'interno dell'applicazione proposta, la macchina a vettore di supporto riceve i dati della rete bayesiana e filtra l'attività normale che dovrebbe seguire l'assistito.

Successivamente, i dati filtrati dalla macchina a vettore di supporto vengono passati a una funzione di regressione non lineare che classificherà il comportamento come normale quindi senza rischio di vagabondaggio oppure anormale e quindi situazione di pericolo dovuta al vagabondaggio.

La Figura 8, mostra le relazioni tra le variabili descritte poc'anzi e il modo in cui quest'ultime influenzano la probabilità di vagabondaggio. Ciascuna variabile contribuisce a dare il proprio "peso" all'interno della rete bayesiana.

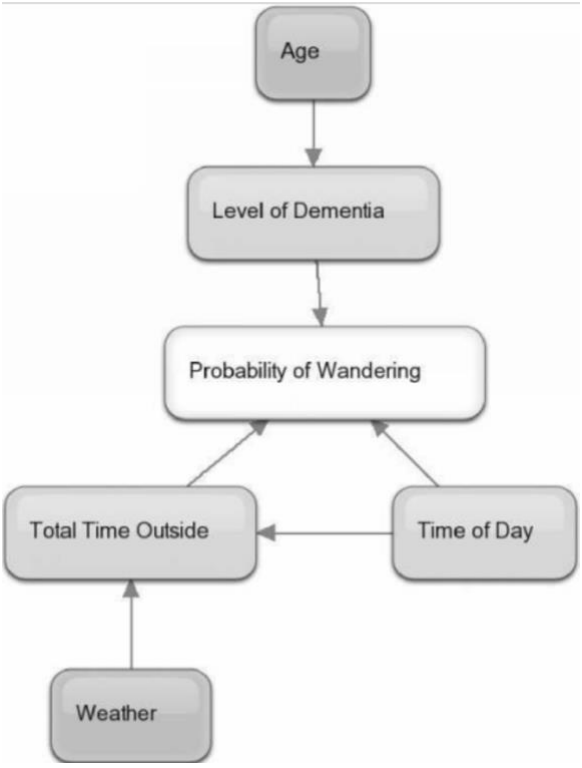


Figura 8 - Relazioni delle variabili della rete bayesiana [8]

Nell'applicazione proposta, se il risultato finale dovesse suggerire il possibile vagabondaggio dell'assistito, una notifica sul cellulare dell'assistito richiede loro di fornire un feedback per vedere se stanno bene. Se a questo feedback l'assistito risponde in maniera positiva vuol dire che quest'ultimo non si trova in una situazione di vagabondaggio. In caso contrario, invece, se l'assistito non dovesse rispondere, l'applicazione deduce che l'assistito potrebbe essersi perso e viene fornito a quest'ultimo le indicazioni stradali per poter tornare indietro verso una zona sicura. In caso non si riuscisse a dedurre cosa sta succedendo all'assistito l'applicazione avvisa tramite notifica il tutore.

L’applicazione considera e analizza anche la velocità dell’assistito, se si dovesse trovare in automobile la probabilità che viaggi al di fuori della zona sicura è molto alta, questo controllo riduce notevolmente i falsi positivi. Tutto questo viene fatto utilizzando e applicando la formula di Haversine discussa anche nei capitoli precedenti.

Infine, le sperimentazioni effettuate affermano l’efficienza dell’applicazione iWander, in particolare le funzionalità che offre [8].

A mio parere, l’applicazione è molto ben strutturata. L’unico difetto che ho notato fin da subito è la mancanza di un sistema per permettere ai tutori di poter controllare in qualsiasi momento la posizione dell’assistito. La gestione del wandering è complessa, ma poter osservare in qualsiasi momento l’assistito potrebbe essere una soluzione in più da considerare per una maggiore tutela. Non sono riuscito a testare l’app perché non è presente negli store online.

2.2.4 MyVigi

MyVigi, come riportato da B. Beauvais et al. [9] è un’applicazione il cui software si basa su un approccio partecipativo che coinvolge gli assistiti, i tutori e gli operatori sanitari. L’applicazione utilizza un accelerometro triassiale incorporato nello smartphone per rilevare le cadute. Grazie alla tecnologia GPS, l’applicazione è in grado di rilevare gli spostamenti utilizzando un sistema di posizionamento globale. In caso di pericolo, i tutori vengono automaticamente contattati tramite telefonate o SMS. Un sito web fornisce ai tutori anche una mappa dove è possibile visualizzare la posizione in tempo reale degli assistiti.

La Figura 9, mostra l’architettura dell’applicazione proposta. Per rilevare il vagabondaggio l’applicazione implementa un algoritmo che si basa sul tempo trascorso dall’assistito al di fuori delle sue zone sicure. Le zone sicure, la fascia oraria in cui queste aree sono visitabili e la durata del percorso da una zona all’altra vengono gestite dall’algoritmo.

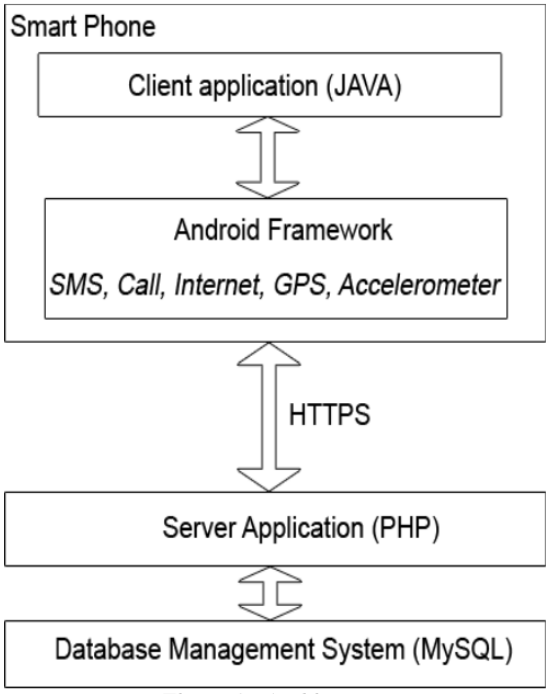


Figura 9 - Architettura proposta [9]

L'applicazione attiva un timer quando l'utente lascia una zona giorno e il vagabondaggio viene rilevato quando il timer supera la durata più lunga per andare in un'altra area. Come nella maggior parte delle applicazioni, le zone sicure, nell'app MyVigi sono delimitate da un cerchio, costituite da latitudine e longitudine. L'algoritmo dell'app utilizza l'API di Android per tracciare il dispositivo dell'assistito e capire se si trova in una di queste zone sicure.

L'applicazione proposta, offre il tracciamento della posizione del dispositivo dell'assistito anche in background, anche se questo tende a far consumare molto la batteria degli smartphone.

La Figura 10, mostra come funziona l'accuratezza per stabilire l'inclusione nella zona sicura. La X rappresenta la distanza tra la posizione dell'assistito e il centro della zona sicura/abitativa più vicina. La φ indica il raggio della zona sicura in cui si trova l'assistito. Il ME è il margine di errore dovuto al fatto che non sempre i dispostivi Android hanno un'accuratezza elevata nel tracciare la posizione. Sempre all'interno della Figura 10 vengono mostrate due formule: la prima, ovvero (1), indica che il tracciamento viene interrotto quando la precisione è sufficiente per stabilire che l'assistito si trova nella zona sicura mentre la seconda, ovvero (2), indica che l'assistito si trova fuori dalla zona sicura.

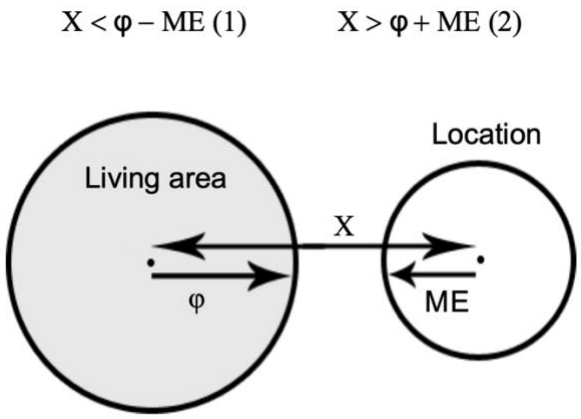


Figura 10 - Rilevamento proposto [9]

L'applicazione proposta manda un avviso automatico quando viene rilevata una situazione pericolosa. Anche l'assistito può chiedere aiuto premendo appositi pulsanti fisici nell'app. Quando l'assistito preme su un pulsante tattile di SOS, l'applicazione cerca di contattare i vari tutori che sono stati salvati precedentemente nel server dove risiedono tutti i dati dell'applicazione. I messaggi di SOS vengono inviati ogni 5 minuti e allo stesso tempo vengono effettuate chiamate ai tutori salvati nell'applicazione.

Infine, dalle sperimentazioni condotte è risultato che l'applicazione è facile da usare, conveniente ed eticamente accettabile per gli assistiti e i loro tutori [9].

Purtroppo non sono riuscito a testare l'applicazione perché non è presente negli store online.

2.3 Dispositivi anti-vagabondaggio

Dopo aver trattato, nei paragrafi precedenti, le applicazioni a supporto di chi ogni giorno affronta il problema del vagabondaggio, ora analizzo i dispositivi anti-vagabondaggio.

Si tratta di dispositivi molto importanti per tutelare al meglio l'assistito. Al giorno d'oggi, la tecnologia ci accompagna in quasi tutto l'arco della giornata, ma se la tecnologia cresce sempre di più in maniera esponenziale, al tempo stesso anche le conoscenze che servono per poterla usare crescono sempre di più. Questo fattore mette in luce alcune problematiche che non possono essere messe in secondo piano. Nei paragrafi precedenti abbiamo parlato di applicazioni molto utili,

innovative e soprattutto tecnologiche. Non tutte le persone, però, riescono utilizzare queste tecnologie. Un esempio, sono le persone anziane che potrebbero non riuscire ad utilizzare smartphone e quindi non usufruire delle applicazioni messe a disposizione per tutelare la loro sicurezza. Al tempo stesso, anche i bambini con disturbi dello spettro autistico potrebbero avere difficoltà nell'utilizzo di cellulari via via sempre più tecnologici.

Proprio per questo, in questo paragrafo voglio descrivere alcuni dispositivi anti-vagabondaggio presenti sul mercato.

2.3.1 AngelSense

AngelSense è un dispositivo di localizzazione anti-vagabondaggio. Questo dispositivo è destinato non solo ai bambini, ma anche ad adolescenti e anziani. Per quest'ultimi i dispositivi possono essere diversi ma le funzionalità offerte sono le stesse.

Infatti, vengono proposti due dispositivi sul sito ufficiale[5].

Un primo dispositivo, raffigurato in Figura 11, è principalmente destinato ai bambini. Si tratta di un piccolo dispositivo dalla grandezza di un biscotto (misura: 2.4 X 1.73 X 0.63) che può essere tascabile oppure indossato con un cinturino, che permette di alleviare lo stress e le ansia dei genitori.

Le funzionalità proposte[6] sono varie e all'avanguardia. Tra le funzionalità, c'è ovviamente la localizzazione GPS dell'assistito con l'utilizzo di mappe in tempo reale e sensori che permettono di avere una localizzazione precisa ed accurata. Ancora, vengono inviate notifiche quando c'è una situazione di pericolo per l'assistito o quando viene rilevato un pericolo di vagabondaggio. Una funzionalità molto importante è poter attivare il vivavoce sul dispositivo dell'assistito in modo tale da poter interloquire. Mette a disposizione gli strumenti di ricerca avanzati per garantire che l'assistito, in una situazione di pericolo, venga trovato rapidamente. Infatti, possono essere configurate più persone in modo tale da avere una rete di "soccorritori" vasta. Incluso, anche un sistema di rilevamento cadute. La Figura 12, invece, mostra un dispositivo AngelSense per adolescenti e persone anziane.

[5] https://www.angelsense.com/kids/
[6] https://www.angelsense.com/gps-tracker-lifesaving-features/

Figura 11 - Dispositivo AngelSense per bambini[5]

Figura 12 - Dispositivo AngelSense per adolescenti e anziani[7]

2.3.2 SmartWatch GPS di FindMyKids

FindMyKids, sul proprio sito ufficiale[8], propone uno strumento di tracciamento innovativo.

Simile a uno smartwatch, l'orologio GPS di FindMyKids implementa una fotocamera che permette ai tutori, genitori ecc. di poter videochiamare il proprio figlio in maniera rapida utilizzando l'applicazione e di poter scattare foto direttamente dall'orologio quando c'è una situazione di pericolo. A differenza dell'applicazione, lo smartwatch implementa un sistema di localizzazione più

[7] https://www.angelsense.com/gps-tracker-for-elderly/
[8] https://gps-watch.findmykids.org/row?utm_source=landing&utm_medium=seo&utm_campaign=it/

accurato e preciso grazie all'utilizzo di Apple Maps, Google Maps e Open Street Maps. Quindi, permette di osservare la cronologia delle posizioni in maniera precisa e dettagliata. Dispone, di un sistema per monitorare l'audio e quindi ascoltare cosa succede nell'ambiente circostante. Per il pulsante di SOS non è presente un tasto virtuale all'interno dell'orologio ma c'è un vero e proprio tasto fisico che l'assistito può premere e inviare subito una richiesta di aiuto in caso di pericolo o smarrimento. È resistente all'acqua e alla polvere, per permettere, in questo caso ai bambini, di potersi divertire nei momenti di svago senza creare danni all'orologio.

La Figura 13, mostra uno smartwatch GPS di FindMyKids.

Figura 13 - Smartwatch GPS di FindMyKids[9]

[9] https://gps-watch.findmykids.org/row?utm_source=landing&utm_medium=seo&utm_campaign=it/

Capitolo 3 – Progettazione

In questo capitolo, prima di spiegare come funziona la mia applicazione, voglio dare una panoramica sugli strumenti che mi hanno permesso di poterla realizzare. In particolare, il sistema operativo Android grazie al quale ho potuto creare, implementare, testare e quindi rendere reale la mia applicazione. Il framework Flutter con il linguaggio di programmazione Dart che hanno reso possibile sviluppare le mie idee grazie a un utilizzo intuitivo degli strumenti messi a disposizione. La piattaforma serverless Firebase che mi ha permesso di effettuare lo storage dei dati della mia applicazione. Successivamente spiego come avviene la localizzazione della posizione geografica tramite GPS. Dopo mostro il nome della mia applicazione e come interagiscono tra loro i vari moduli e componenti software.

3.1 Android

Android è un sistema operativo principalmente per dispositivi mobili, sviluppato da Google. Può essere eseguito su numerosi dispositivi come ad esempio smartphone, televisori, orologi, computer[10]. Utilizza un kernel Linux modificato. La piattaforma Android fu introdotta da Open Handset Alliance nel 2007. Ad oggi, la maggior parte delle applicazioni Android sono scritte in Java. Alla piattaforma Android viene affiancata la macchina virtuale Dalvik, infatti le classi Java vengono prima compilate in eseguibili Dalvik e poi eseguite sulla macchina virtuale Dalvik[11]. Android è un progetto quasi totalmente libero e soprattutto open source rilasciato con licenza Apache 2.0. Bisogna capire, però, cosa significa piattaforma aperta. Vuol dire che non è possibile contribuire mentre una versione è in fase di sviluppo, ma l'apertura inizia quando il suo codice sorgente viene rilasciato al pubblico dopo la sua finalizzazione, cioè chiunque può reperire il codice e modificarlo come crede[11]. Nel momento in cui si vuole creare un'applicazione bisogna richiedere l'Android SDK importante perché offre gli strumenti necessari e soprattutto le API. Il sistema operativo Android, è il più diffuso tra i dispositivi mobili in tutto il mondo. Oggi, l'ultima versione rilasciata è la 13, che prende il nome di Android Tiramisù, caratteristici sono i nomi delle varie versioni che prendono spunto da vari dolciumi. Android include un "application framework" che permette il riutilizzo dei componenti sviluppati[12]. Prevede il supporto per fotocamera, GPS, bussola e accelerometro che ho sfruttato anche nella mia applicazione.

[10] https://it.wikipedia.org/wiki/Android
[11] https://it.theastrologypage.com/android-platform
[12] http://www.diit.unict.it/users/gascia/COURSES/sist_emb_13_14/download/SE12_Android_01.pdf

3.1.1 Android SDK

Android SDK (Software Development Kit), comprende tutti gli strumenti necessari per lo sviluppo di applicazioni eseguite sul sistema operativo Android. L'Android SDK contiene librerie, API, un emulatore, un debugger, ecc[13].

Elemento fondamentale per lo sviluppo di applicazioni è l'emulatore offerto da Android SDK che permette di simulare un device mobile con schermo touch screen, audio, GPS, fotocamera e quasi tutte le funzionalità disponibili all'interno degli smartphone. L'emulatore è uno strumento molto importante per chi sviluppa applicazioni perché permette di fare test accurati della propria applicazione prima di estrapolare l'apk e installarla sullo smartphone vero e proprio. Meglio ancora, permette anche di installare i driver necessari sul proprio pc per utilizzare il proprio smartphone come una sorta di emulatore. In poche parole, collegando lo smartphone al pc si riesce a testare l'applicazione anche sul proprio cellulare senza dover usare l'emulatore dell'SDK.

3.2 Flutter

Flutter è un framework open-source creato da Google per la creazione di app native di alta qualità su iOS, Android, Linux, macOS e Windows[14]. L'architettura del framework Flutter è costituita da componenti principali, che sono: il linguaggio di programmazione Dart, la Flutter Engine, la Foundation Library e i Design-specific widgets[14].

La Flutter engine è scritta principalmente in C++ e fornisce la generazione delle immagini a partire dal codice a basso livello utilizzando la libreria grafica di Google. Avviene anche un collegamento tra la flutter engine e l'SDK della piattaforma Android oppure iOS in base alle esigenze.

La flutter engine essendo scritta in Dart permette di effettuare un "hot reload" dell'applicazione. Nel mentre scriviamo la nostra applicazione è possibile testare quest'ultima tramite un emulatore, nel caso in cui dovessimo cambiare qualche parte di codice, con l'"hot reload" la modifica viene aggiunta all'istante all'interno dell'applicazione senza dover effettuare un riavvio completo dell'applicazione.

La Foundation library, scritta sempre in Dart, offre classi e funzioni di base per la costruzione di applicazioni che utilizzano Flutter[14].

I Widget, sono fondamentali per la creazione dell'interfaccia utente delle applicazioni in Flutter. Un'applicazione Flutter può essere vista come una sorta di assemblaggio di vari widget, che ovviamente devono essere creati prima di poterli usare. Rappresentano animazioni, testi, grafici che fanno parte dell'interfaccia utente.

[13] https://www.geekandjob.com/wiki/android-sdk
[14] https://it.wikipedia.org/wiki/Flutter_(software)

Il Design-specific widget comprende due contenitori di widget diversi tra di loro perché si riferiscono a widget di specifici linguaggi di programmazione. Ad esempio, i widget in stile Material Design comprendono il design di Google mentre i widget di Cupertino il design di iOS.

3.2.1 Dart

Dart è un linguaggio di programmazione orientato agli oggetti, completamente open source e sviluppato da Google[15]. È stato pubblicato nel 2011, era nato per sviluppare web app, ma non è stato preso in considerazione più di tanto in quel periodo. Ha avuto una sua rivalutazione quando il framework Flutter l'ha adottato come linguaggio di programmazione. Proprio perché Flutter rappresenta una delle tecnologie per sviluppare applicazioni all'avanguardia, ha permesso a Dart di crescere notevolmente. Non a caso, oggi Dart è uno dei linguaggi di programmazione più amati dagli sviluppatori. Permette di essere compilato per il web, per tutti i sistemi operativi presenti oggi sul mercato e compilato per Android e iOS scrivendo codice che è nativo per entrambe le piattaforme. Offre tutti gli attributi necessari per essere un linguaggio moderno. Per sviluppare le proprie applicazioni in Dart è possibile utilizzare come ambiente di sviluppo Visual Studio Code che spiego nel paragrafo successivo.

3.2.2 Visual Studio Code

Visual Studio Code è un ambiente di sviluppo efficiente dedicato allo sviluppo di applicazioni e non solo. Ideato da Microsoft per Windows, Visual Studio Code include il supporto per il debugging, un canale dedicato al controllo di Git, IntelliSense ovvero il completamento automatico delle istruzioni di codice. Come abbiamo detto anche in precedenza, Visual Studio Code è un editor cross-platform compatibile con i sistemi operativi presenti sul mercato. Uno dei punti di forza di questo ambiente di sviluppo sono le estensioni, che si possono scaricare e installare da un apposito canale. All'interno di questo canale possono essere cercate estensioni tra le più importanti per gli sviluppatori. Non a caso, VSC può essere usato con vari linguaggi di programmazione tra cui il linguaggio C, C++, C#, HTML, PHP, Java e ovviamente il nostro Dart. La Figura 14 mostra una schermata di Visual Studio Code.

[15] https://www.html.it/pag/376858/introduzione-a-dart/

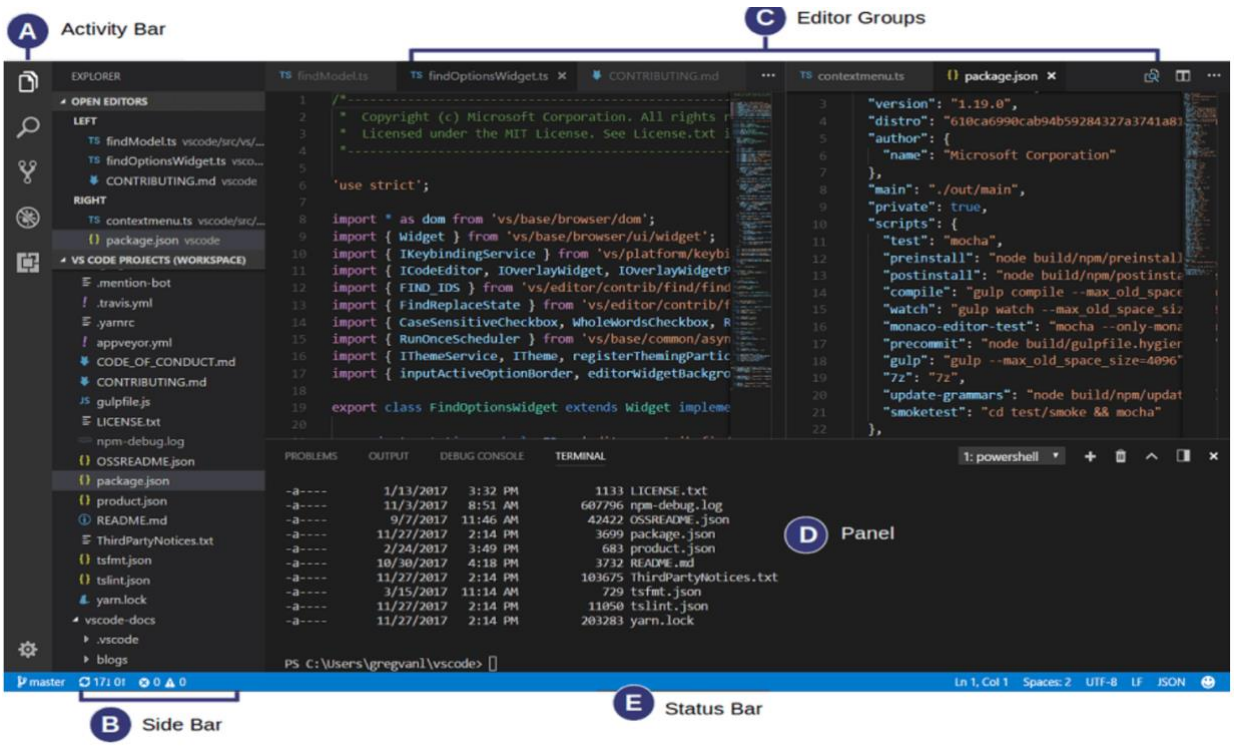


Figura 14 - L'ambiente di sviluppo, Visual Studio Code[16]

3.2.3 Android Manifest e permessi

Il file Android Manifest è un file XML obbligatorio in ogni progetto Android, che descrive le informazioni essenziali dell'applicazione e le sue componenti. Al giorno d'oggi, il tema della privacy è diventato tema centrale quando parliamo di software, applicazioni e tutti gli strumenti che hanno a che fare con la tecnologia odierna. Proprio perché parliamo di applicazioni, la privacy diventa tema centrale perché riguarda la sicurezza di chi utilizza le applicazioni. Android cerca quindi di gestire questo problema utilizzando l'Android Manifest. Questo file contiene le informazioni sul nome del pacchetto dell'applicazione, la versione dell'applicazione, le autorizzazioni richieste dall'applicazione, l'elenco delle attività e dei servizi. Il file viene usato da Android per avviare l'applicazione in questione e gestire il comportamento di quest'ultima. Paragonabile a una mappa dell'applicazione che Android utilizza per comprendere le funzionalità e le componenti.

All'interno dell'Android Manifest sono presenti i permessi o autorizzazioni, vere e proprie dichiarazioni che indicano alle applicazioni Android quali funzionalità dello smartphone, ad esempio, possono utilizzare. Ogni sviluppatore si deve affidare al file Android Manifest per definire i permessi dell'applicazione. Questi permessi servono poi all'applicazioni di accedere ad alcune funzionalità, come ad esempio la fotocamera, la posizione GPS, i contatti, la connessione Internet, i servizi in background, ecc. Esistono due categorie di permessi, i permessi normali e i permessi pericolosi. I primi, ovvero i permessi normali non rappresentano una minaccia per la privacy degli utenti e sono concessi automaticamente nel momento in cui viene installata l'applicazione. Al contrario, invece, i permessi pericolosi possono rappresentare una minaccia per la privacy degli utenti e richiedono un consenso esplicito agli utenti affinché possano essere utilizzati. Tutto questo

[16] https://code.visualstudio.com/docs/getstarted/userinterface

deve essere gestito nel migliore dei modi per evitare che alcune informazioni degli utenti vengano usate per scopi illeciti.

3.2.4 Il file pubspec.yaml

Il file *pubspec.yaml* è un file di configurazione che si trova nella radice di un progetto Flutter e definisce le dipendenze del progetto, le risorse come le immagini e le font, le configurazioni dell'applicazione come ad esempio il nome, la versione e la descrizione e altre informazioni progettuali. Questo file definisce un'informazione importante di Flutter come per esempio l'opzione *"uses-material-design"* che sta a indicare se l'applicazione deve usare o meno il Material Design per l'interfaccia grafica[17]. L'aspetto da tenere presente è che all'interno di questo file devono essere inserite e configurate le varie dipendenze che possono essere pacchetti di terze parti, librerie oppure risorse. Una volta inserite dovrà essere eseguito il comando "*pub get"* per scaricare e installare definitivamente le dipendenze nel progetto. Ancora, all'interno del file possono essere definite il nome e la versione dell'applicazione in modo tale da poter usati quando l'applicazione viene pubblicata nell'app store.

3.2.5 Stateful e Stateless Widget

Le fondamenta di un'applicazione Flutter sono gli Stateful e Stateless Widget. Come ho detto nei paragrafi precedenti, un'applicazione Flutter può essere vista come un insieme di widget che, messi in relazione tra loro, creano l'applicazione desiderata. Così come avviene quando si costruisce un palazzo, si parte dalle fondamenta e pian piano si costruisce l'applicazione dal basso verso l'alto. Ovviamente, c'è una grossa differenza tra i due, ovvero nell'abilità di ricaricare il widget durante il tempo di esecuzione.

Lo Stateless Widget (ovvero, widget senza stato) è un widget immutabile, il che significa che una volta creato, non può essere modificato. Il loro stato, durante il "runtime" o tempo di esecuzione, non cambia mai. Solitamente, vengono utilizzati per visualizzare tutte le informazioni di tipo statico, come il testo, le icone e le immagini. Quando viene creato uno Stateless Widget, allo stesso tempo deve essere definito un metodo build() che restituisce un nuovo widget ogni volta che Flutter richiede una ridisegnazione dell'interfaccia utente. Quando un'applicazione viene creata, lo sviluppatore deve concentrarsi anche sulla scelta degli elementi visivi da inserire nell'applicazione ovvero i widget. Gli stateless widget non cambiano il proprio comportamento neanche se dovessero esserci eventi o azioni scaturite dall'utente. Alcuni esempi tipici di Stateless Widget sono i Text, Row, Column e i Container.

Lo Stateful Widget (ovvero, widget con stato) ha lo scopo di tenere sotto controllo lo stato dei widget che gestisce. È composto da due classi: la classe principale del widget e una classe di stato

[17] https://cloudsurfers.it/index.php/sviluppare-applicazione-flutter-parte-1/#:~:text=Il%20file%20pubspec.,Design%20per%20l'interfaccia%20grafica.

('State') che si assicura di mantenere lo stato del widget. Ogni volta che avviene un cambiamento all'interno dello State, quest'ultimo deve forzare la ricreazione del widget che ovviamente sarà aggiornato e visibile all'utente. In particolare, quando avviene una modifica nello State o stato, viene chiamato un metodo setState(), il quale informerà dell'imminente ridisegnazione del widget per apportare le modifiche. Alcuni esempi tipici di Stateful Widget sono le Image, i Form e le CheckBox.

3.2.6 I Container

I Container sono widget che possono contenere altri widget. Questo significa che il Container può essere visto come una sorta di grande scatola dove al suo interno possono essere inseriti svariati oggetti, nel nostro caso più widget. Nel momento in cui vengono inseriti i widget all'interno del Container, questi vengono chiamati widget figli. La spiegazione del nome "widget figli" è molto intuitiva perché una volta inseriti nel Container, quest'ultimo prende il controllo del layout, dell'aspetto e della decorazione dei figli. Il Container offre allo sviluppatore una serie di proprietà che permettono di modificare l'aspetto dei widget figli, ma anche dell'aspetto dell'intera area creata dal Container stesso, a proprio piacimento. Facciamo un po' di esempi: c'è la proprietà *color* che specifica il colore di sfondo del Container, le proprietà *width* e *height* rispettivamente rappresentano la larghezza e l'altezza del widget, il *margin* sta a identificare la distanza tra i bordi dei vari widget figli presenti nel Container e il *padding* che invece permette di definire uno spazio però questa volta all'interno del o dei widget figli. Ovviamente, le proprietà non solo queste, ma ho voluto riassumere quelle più importanti e che personalmente ho usato di più nello sviluppo della mia applicazione.

3.2.7 Il widget Image e la cartella 'assets'

Il Widget Image, lo si capisce fin da subito dal nome, permette di visualizzare all'interno della nostra applicazione immagini o icone. Utilizzando il costruttore *"Image.asset"* è possibile caricare le immagini nell'applicazione, in particolare all'interno del costruttore deve essere inserito l'URL o il percorso dell'immagine. Di solito tutte le immagini vengono salvate all'interno di una cartella di progetto chiamata *"assets"*, per includere risorse statiche più precisamente. Quando viene utilizzata un'immagine all'interno dell'applicazione oltre ad inserire l'immagine nella cartella *assets* deve essere inserito anche nel file pubspec.yaml che contiene una sezione appunto chiamata "assets".

3.2.8 I Pulsanti in Flutter

I Pulsanti in Flutter, sono widget che nel momento in cui vengono premuti dall'utente eseguono un'azione. Il widget di pulsanti più utilizzato è sicuramente il RaisedButton widget che rappresenta a tutti gli effetti un pulsante sopra-elevato[18]. A sua volta, fanno parte dei RaisedButton widget i

[18] https://www.html.it/pag/382161/gestione-dei-pulsanti-con-i-raisedbutton-widget/

FlatButton che non definisce alcuna elevazione rispetto a tutto quello che c'è sotto e non crea un'ombra[18]. I FloatingActionButton che rappresenta un pulsante circolare con un icona[18]. I TextBotton che invece mostra un pulsante con un testo e nessuno sfondo. Anche i RaisedButton hanno delle proprietà importanti tra cui il *child* che permette di specificare il widget figlio del RaisedButton[18]. L' *onPressed* è un'altra proprietà fondamentale perché permette di definire la funzione di callback che si attiva quando l'utente preme sul pulsante.

3.2.9 Row e Column Widget e le ListView

Le Row e le Column sono widget che permettono di disporre altri widget orizzontalmente e verticalmente all'interno dell'applicazione. Per gli sviluppatori è importante poter disporre i vari widget in maniera ordinata all'interno dell'applicazione.

La Row permette di disporre i widget in una riga orizzontale che poi verranno allineati secondo le proprie esigenze.

La Column, a differenza della Row, permette di disporre i widget in una colonna verticale.

Entrambi hanno delle proprietà fondamentali, tra cui: la proprietà *children* che corrisponde a una vera e propria lista con tutti i widget, quest'ultimi vengono visualizzati in una colonna o riga in base al widget padre selezionato se Row o Column. Altra proprietà è la *crossAxisAlignment* dove i widget della Row o della Column vengono allineati al centro. Questa proprietà vale anche per le immagini. Il difetto principale delle Row e delle Column è che non permettono lo scroll orizzontale o verticale e se si dovessero visualizzare un numero elevato di elementi entra in gioco la ListView[19].

La ListView è un widget che permette di visualizzare una serie di elementi in modo scrollabile. Molto efficace come widget, permette di gestire numerose informazioni all'interno di un unico widget. Infatti, anche se l'elenco degli elementi dovesse essere lungo, grazie alla proprietà di scroll è possibile visualizzare il tutto.

3.2.10 Stack Widget

Lo Stack Widget entra in gioco quando si vuole creare un'interfaccia utente complessa. La parola stessa Stack fa pensare alla struttura dati della Pila, in effetti è proprio così. Infatti, lo stack widget può contenere molti widget figli che si trovano su livelli differenti, creando e componendo una vera e propria pila di elementi[20]. Permette di sovrapporre altri widget all'interno di una schermata, questi widget possono essere testi, immagini o pulsanti. Se lo Stack dovesse avere più widget figli, la collocazione di quest'ultimi avverrebbe secondo la struttura dati Pila, ovvero il primo widget figlio viene posizionato nella parte inferiore dello stack e tutti gli altri verranno sovrapposti. Quindi il

[19] https://www.html.it/pag/383628/row-e-column-widget-le-fondamenta-dei-multi-child-layout/
[20] https://www.html.it/pag/384770/stack-widget-per-la-sovrapposizione-di-widget/

secondo widget figlio verrà posizionato sopra il primo widget figlio e così via, l'ultimo widget figlio si troverà in cima allo Stack. Utilizzando la proprietà *positioned* si può specificare la posizione dei widget figli nello Stack. La funzionalità dello scroll non è ammessa all'interno di uno Stack Widget.

3.2.11 Navigare tra le schermate di un'app

Come ultimo paragrafo relativo a Flutter, ho deciso di spiegare in breve come navigare tra le schermate di un'app. È una funzionalità molto importante e usata che permette di navigare tra le schermate di un'app in maniera rapida e scambiando anche dati. Proprio per questo Flutter mette a disposizione tre componenti[21]: la Route, il Navigator, il MaterialPageRoute e il MaterialApp.

La Route sta a rappresentare una schermata dell'applicazione.

Il Navigator, e qui entrano in gioco di nuovo le strutture dati, è un widget che gestisce le Route mantenendole all'interno di uno stack FIFO. FIFO rappresenta la struttura dati della coda ovvero, l'elemento che viene inserito nella prima posizione è il primo a uscire dalla lista. Permette quindi all'utente di poter navigare avanti e indietro tra le route (schermate).

Il MaterialPageRoute, invece, permette all'utente di passare da una schermata a un'altra singola schermata dell'applicazione. Si tratta di una transizione singola.

Il MaterialApp rappresenta la radice di un'applicazione Flutter. Utilizzando questo widget si riesce a definire uno schema dell'intera applicazione. Infatti, può essere impostata una route iniziale e subito dopo vengono inizializzate una serie di *routes* ovvero di pagine che permettono poi allo sviluppatore di poter gestire al meglio la navigazione tra quest'ultime. Affinché si possa navigare tra le routes entra in gioco il Navigator. Il MaterialApp è importante se si vuole creare un'interfaccia utente ordinata, personalmente ho incluso questo widget, così come gli altri, all'interno della mia applicazione.

3.3 Geolocalizzazione in Flutter e GPS

Una delle funzionalità importanti all'interno di Flutter è la possibilità di poter lavorare con il plugin della Geolocalizzazione. Questo plugin sta diventando sempre più importante all'interno delle applicazioni perché permette di implementare una soluzione rapida ed efficace per individuare la posizione degli utenti che utilizzano l'applicazione. Flutter, offre questa possibilità attraverso diversi plugin[22]. I plugin più utilizzati sono il plugin *location* e il plugin *geolocator*. All'interno della mia applicazione ho deciso di utilizzare il plugin *geolocator* perché l'ho ritenuto più in linea con i gli obiettivi da me prefissati. Questo plugin, in particolare, l'ho utilizzato per raccogliere due importanti fattori: la posizione corrente dell'utente tramite le coordinate di latitudine e longitudine e

[21] https://www.html.it/pag/394799/flutter-navigazione-e-routing-le-basi/
[22] https://www.html.it/pag/397677/geolocalizzazione-flutter/

poi per recuperare l'indirizzo associato a quelle coordinate. Dovranno essere definiti dei permessi affinché si possa utilizzare questo plugin, dettagli che approfondisco nel Capitolo 4.

Qualsiasi dispositivo cellulare, al giorno d'oggi, contiene un sistema GPS. Aspetto fondamentale per poter creare applicazioni che utilizzando il sistema GPS per individuare le coordinate degli utenti. GPS è l'acronimo di Global Positioning System, è un sistema di navigazione satellitare di proprietà del governo degli Stati Uniti[23]. Il GPS è composto da una rete di satelliti in orbita intorno alla terra che trasmettono 24 ore su 24 segnali e dati ai ricevitori GPS sulla terra. Ciascun satellite invia un segnale che comprende parametri orbitali univoci che permettono ai dispostivi GPS di decodificarli[23]. Quindi, il ricevitore GPS utilizza i segnali provenienti dai satelliti per triangolare la propria posizione. Il ricevitore GPS misura la distanza da ciascun satellite in base al tempo necessario per ricevere un segnale trasmesso e, compiendo misurazioni multiple, determina la posizione dell'utente su uno schermo che può essere di un cellulare, di uno smartwatch, ecc. e permette, ad esempio, di misurare la strada per tornare a casa[23]. Nel mio caso l'utilizzo del sistema GPS viene utilizzato per geolocalizzare gli utenti che utilizzano l'applicazione.

3.3.1 Gestione delle mappe in Flutter

Dopo aver descritto i plugin necessari affinché si possa geolocalizzare la posizione degli utenti, nel mio caso utilizzando *geolocator,* in questo paragrafo descrivo cosa ho utilizzato per gestire la mappa nella mia applicazione. Tutte queste informazioni vengono approfondite nel Capitolo 4 in maniera dettagliata e soprattutto con esempi pratici che ho implementato nella mia applicazione. Il plugin che ho utilizzato per implementare la mappa è *google_maps_flutter*. Con questo plugin è possibile integrare le mappa di Google nella nostra applicazione, aggiungere vari marker e mostrare la posizione corrente dell'utente[24]. Inoltre, con questo plugin è possibile personalizzare le mappe come ad esempio lo stile, aggiungere i marker o marcatori, individuare la posizione degli utenti sulla mappa e tracciare percorsi da una posizione all'altra.

3.4 Firebase

Firebase è una piattaforma serverless per lo sviluppo di applicazioni mobili e web[25]. È una piattaforma open source e viene supportata da Google, infatti Firebase utilizza alcune funzionalità di quest'ultima tra cui il cloud per fornire una serie di strumenti per scrivere, analizzare e mantenere applicazioni cross-platform ovvero con codice nativo per Android e iOS[25]. I vari servizi offerti includono l'autenticazione dell'utente, la gestione di due database, l'hosting, la messaggistica in tempo reale e l'analisi delle prestazioni di una qualsiasi applicazione. Mette a disposizione una serie di API e strumenti importanti per gli sviluppatori per permettere di creare applicazioni di alta

[23] https://www.garmin.com/it-IT/aboutgps/
[24] https://www.html.it/pag/397771/flutter-gestione-delle-mappe/
[25] https://www.geekandjob.com/wiki/firebase

qualità e sicure. Firebase è molto intuitivo da usare e facile, non a caso è diventata una delle piattaforme più usate per lo sviluppo di app.

All'interno della mia applicazione ho utilizzato alcuni strumenti messi a disposizione da Firebase tra cui: *Firebase Cloud Messagging (FCM), Firebase Auth, Real-Time Database e Cloud Firestore*.

All'interno dello studio condotto da C. Khawas et al. [10] viene spiegato cos'è Firebase, ma in particolare gli strumenti che mette a disposizione. Firebase utilizza JSON per memorizzare i dati ed è basato su NoSQL [10].

3.4.1 Firebase Cloud Messagging (FCM)

Come spiegato da C. Khawas et al. [10] , FCM è un servizio a pagamento che rappresenta una soluzione multipiattaforma per messaggi e notifiche Android, applicazioni Web e iOS [10]. In particolare, all'interno della mia applicazione ho utilizzato FCM per inviare notifiche ai dispositivi registrati tramite il token presente in ogni dispositivo cellullare.

3.4.2 Firebase Auth

Come spiegato nell'articolo di Khawas et al. [10], Firebase Auth supporta provider di login sociali come Facebook, Google, GitHub e Twitter. È un servizio in grado di autenticare gli utenti utilizzando solo codice lato client. Include anche un sistema di gestione degli utenti, infatti gli sviluppatori possono abilitare l'autenticazione degli utenti con l'email e la password di accesso memorizzate in Firebase[10]. All'interno della mia applicazione ho utilizzato questo strumento proprio per mettere agli utenti di autenticarsi nella mia app e salvare i dati su Firebase.

3.4.3 Real-time Database

Come spiegato nell'articolo di Khawas et al. [10], Firebase fornisce una serie di servizi come un database e un backend in tempo reale. Agli sviluppatori viene fornita un'API che consente di sincronizzare i dati delle varie applicazioni e di memorizzarli sul cloud di Firebase. Infatti, vengono fornite librerie client che consentono l'integrazione con applicazioni Android e iOS [10].
All'interno della mia applicazione ho utilizzato questo strumento per permettere la creazione, il salvataggio e il caricamento dei contatti di emergenza da parte degli utenti.

3.4.4 Cloud Firestore

Cloud Firestore[26] è un database NoSQL ospitato nel cloud in cui le app Android e iOS possono accedere direttamente tramite SDK nativi. Mantiene i dati sincronizzati tra le app client tramite listener in tempo reale e memorizza i dati in documenti che contengono campi associati a valori. A loro volta questi documenti sono salvati all'interno di raccolte ovvero dei contenitori per i documenti degli utenti[26]. È possibile all'interno del codice creare query da sottoporre a Firestore. Sono query che riescono a recuperare i dati a livello di documento senza dover recuperare l'intera raccolta[26]. All'interno della mia applicazione, ho utilizzato Cloud Firestore per salvare tutte le informazioni degli utenti quando si registrano all'applicazione. Infatti, vengono salvate all'interno di una raccolta *utenti,* e per ogni utente ci sono vari campi a cui verrà assegnato un valore.

3.5 Nome dell'applicazione

Il nome che ho scelto per la mia applicazione è **SafeKids**.

Ho scelto questo nome perché rappresenta appieno il mio lavoro di tesi, ma soprattutto il mio obiettivo, oltre quello di creare questa applicazione, anche di poter aiutare le persone in difficoltà in questo caso bambini, bambini con disturbi dello spettro autistico e anziani. Riuscire a dare un senso di sicurezza, da qui la parola "Safe", ai genitori, ai tutori e a chiunque quotidianamente affronta le problematiche trattate. Mentre la parola "Kids", ovvero "Ragazzi", per indicare come specificato poc'anzi a chi è rivolta questa applicazione.

3.6 Progettazione delle schermate e moduli software

La mia applicazione sostanzialmente è rivolta a bambini con e senza disturbi dello spettro autistico e anziani che soffrono di demenza. Non solo, è rivolta anche ai tutori, parenti e personale ospedaliero incaricato di controllare il proprio assistito.

Come prima schermata principale quando si apre l'applicazione ho pensato di inserire un messaggio di benvenuto, con la scelta tra due opzioni, da parte dell'utente, per registrarsi oppure accedere all'applicazione. La schermata della registrazione è intuitiva e semplice da compilare, con icone grandi e ben segnalate. Ovviamente che si tratti di un bambino o di una persona anziana è sempre meglio affidare questa fase ai genitori o ai tutori ecc. In poche parole, il modulo software sulla fase di autenticazione dovrebbe essere sempre seguito da un parente o da un tutore affinché la fase vada a buon termine. Allo stesso modo, anche il modulo software relativo all'associazione fra i due telefoni deve avvenire con l'aiuto di un genitore o tutore non solo per completare al meglio

[26] https://firebase.google.com/docs/firestore?authuser=0&hl=it

l’operazione ma anche perché l’operazione richiede che l’assistito e il suo tutore siano vicini per la scannerizzazione del QR Code. Il terzo modulo, invece, si divide in due parti. Una parte destinata ai genitori e quindi ai tutori dove è presente una schermata con la mappa in cui viene segnata la propria posizione, poi tre bottoni posti nella parte bassa, uno per accedere ai contatti di emergenza, uno per mostrare sulla mappa la posizione dell’assistito e l’ultimo per impostare il raggio della zona sicura che si andrà a creare in corrispondenza della propria posizione. Mentre nella parte alta della schermata sono presenti due icone, una per le Info dell’app e l’altra per accedere alle Impostazioni. La seconda parte, invece, destinata solo agli assistiti costituita da una schermata con la mappa che segnala la posizione degli assisiti e un pulsante per i contatti di emergenza posto in basso. Anche qui, nella parte alta ci sono le due icone: Info e Impostazioni. La schermata delle Impostazioni dove è possibile creare, salvare ed eliminare contatti di emergenza, è destinata sia ai tutori che agli assistiti.

Prima di spiegare come ho scelto di creare le varie schermate, voglio mostrare nella Figura 15 come interagiscono tra loro le varie schermate.

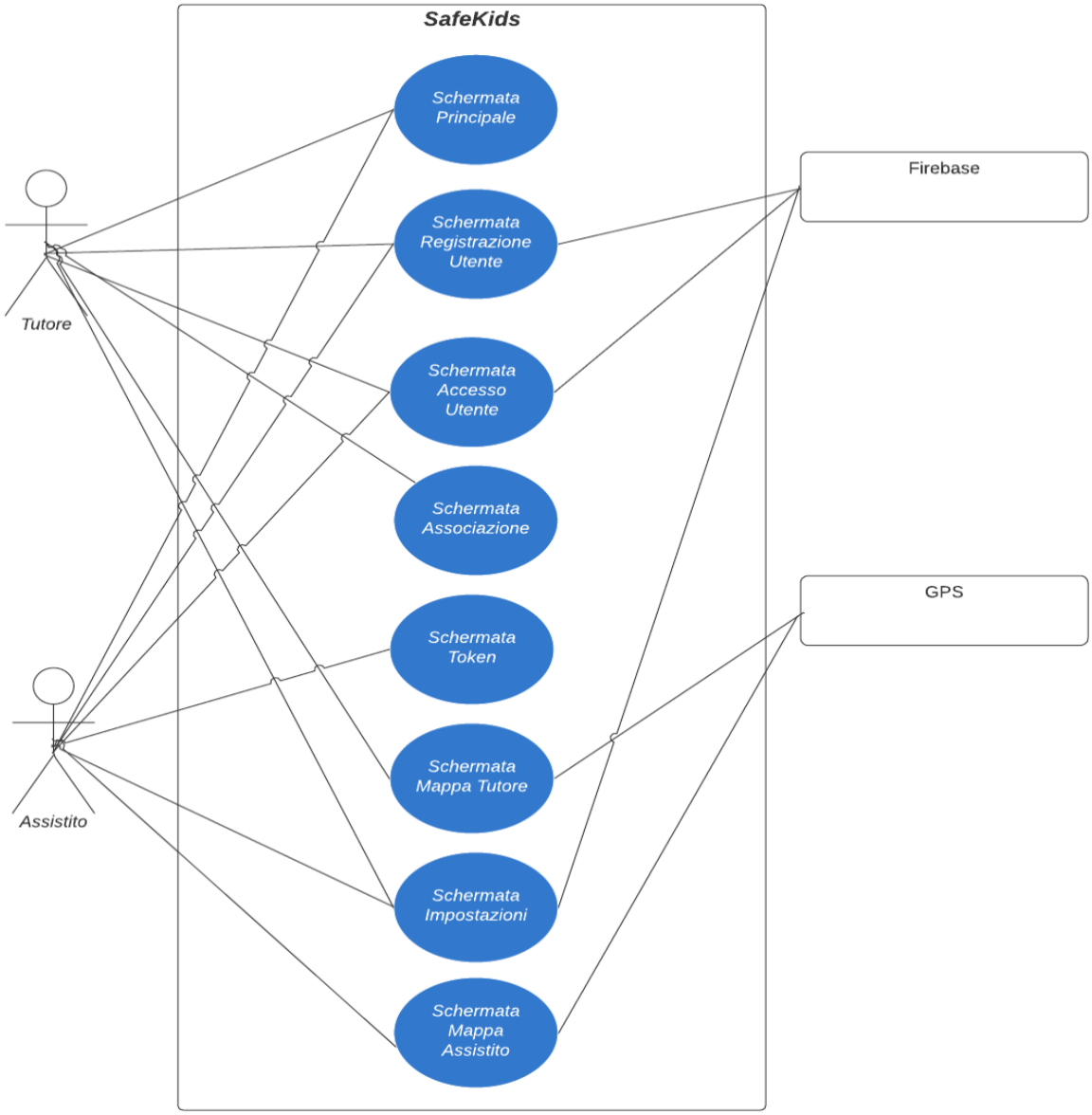


Figura 15 – Diagramma dei casi d'uso di SafeKids

3.6.1 Schermata principale

La schermata principale è la schermata che si apre nel momento in cui l'utente apre l'applicazione. All'utente viene mostrato un messaggio di "Benvenuto" e subito sotto due grandi bottoni: uno con la scritta "Accedi", l'altro con la scritta "Registrati". Sono molto visibili proprio perché l'applicazione è pensata per bambini e anziani e anche per i tutor.

Avere un'interfaccia utente semplice e intuitiva aiuta molto e soprattutto migliora l'usabilità dell'applicazione stessa. Ovviamente, premendo il bottone "Registrati" si apre la pagina per effettuare la registrazione dell'utente, mentre premendo il bottone "Accedi" si apre la pagina per permettere all'utente di accedere all'applicazione.

3.6.2 Schermata registrazione utente

Quando nella schermata principale viene premuto il bottone "Registrati" si apre la schermata registrazione utente, che permette all'utente di inserire vari campi per potersi registrare ed utilizzare l'applicazione. In particolare la schermata è costituita dal campo *"Nome", "Cognome", "Username", "Email", "Password", "Conferma Password" e "Ruolo:(genitore/figlio)"*. Viene controllato il campo *Password* per verificare l'esattezza della password e il campo *Ruolo* può avere solo due valori "genitore" o "figlio".

Se l'utente è un tutore dovrà inserire "genitore" nel campo Ruolo, mentre se l'utente è l'assistito dovrà inserire "figlio" nel campo Ruolo. In base a questa differenza l'applicazione, prende due strade separate.

Nella parte inferiore della schermata è presente un bottone con scritto "Registrati", che una volta premuto, avvia la registrazione dell'utente salvando tutti i campi all'interno di Firebase, in particolare nel Cloud Firestore, nella raccolta *utenti*.

Su Cloud Firestore, nel momento in cui l'utente si registra l'applicazione agisce in due modi diversi: se l'utente è il tutore e quindi seleziona nel campo Ruolo la stringa "genitore", al momento della registrazione su Firestore oltre a salvare i campi principali citati prima, vengono automaticamente aggiunti anche il campo "*token*" e il campo "*tokenFiglio*", quest'ultimo verrà poi avvalorato in fase di associazione con l'assistito. Invece, se l'utente è l'assistito e quindi seleziona nel campo Ruolo la stringa "*figlio*", al momento della registrazione su Firestore oltre a salvare i campi principali citati prima, vengono automaticamente aggiunti anche il campo "*token*" e i campi "*latitudine*" e "*longitudine*" che verranno poi avvalorati quando l'assistito aprirà la mappa.

I campi da compilare sono grandi e ben visibili cosi come il bottone per registrarsi, in modo tale da rendere la fase di registrazione semplice e intuitiva.

Dopo che l'utente si registra, viene aperta la schermata per poter Accedere all'applicazione.

3.6.3 Schermata accesso utente

Una volta premuto il bottone “Registrati” nella schermata per la registrazione dell’utente, si apre la pagina per poter accedere all’applicazione. In particolare, viene chiesto all’utente di inserire *Email* e *Password* create in fase di registrazione.

All’interno della schermata quindi ci sono due campi: un campo email e un campo password. Subito sotto un bottone con scritto “Accedi” che una volta premuto, l’applicazione recupera da Firebase l’email e la password dell’utente in questione. Ovviamente, se dovessero esserci problemi come ad esempio errori di battitura l’accesso all’applicazione non avviene.

L’applicazione oltre a recuperare l’email e la password su Firebase, recupera anche il Ruolo dell’utente che ha compiuto l’accesso, in modo tale da far prendere due strade differenti all’applicazione, una destinata al tutore e una all’assistito.

Se l’utente dovesse avere il campo Ruolo impostato a “genitore” e il campo “tokenFiglio” impostato a “nonAssociato” vuol dire che l’utente è il tutore dell’assistito ma non si è ancora associato e quindi si apre la schermata per scannerizzare il QR code presente sul telefono dell’assistito. Ancora, se l’utente dovesse avere il campo Ruolo impostato a “genitore” e il campo “tokenFiglio” impostato con un token, l’applicazione deduce che l’associazione tra tutore e assistito è avvenuta e quindi apre la schermata della mappa rivolta ai tutori.

Invece, se l’utente che compie l’accesso all’applicazione è l’assistito, quindi ha il campo Ruolo impostato a “figlio”, si apre la pagina per permettere di generare il QR code e anche in questa pagina ci sarà una verifica che spiego nella schermata token.

3.6.4 Schermata associazione

Questa schermata viene visualizzata dall’utente tutore non ancora associato, ovvero l’utente che ha come valore nel campo Ruolo la stringa “genitore” e come valore nel campo “tokenFiglio” la stringa “nonAssociato”.

La schermata si apre con un bottone posto al centro della schermata con scritto “Scannerizza QR code”, che una volta premuto dall’utente, permette di aprire la fotocamera del cellulare, chiedendo i permessi all’utente, per poter scannerizzare il QR code presente sul cellulare dell’assistito.

3.6.5 Schermata token

Questa schermata viene visualizzata dall'utente assistito, ovvero l'utente che ha il campo Ruolo impostato a "figlio". Quando si apre la schermata viene mostrato un messaggio stampato a schermo e subito sotto un bottone con scritto "Genera QR code".

Il messaggio di testo stampato a schermo informa l'utente assistito che se non è ancora associato ovvero, il token dell'assistito non è stato ancora associato a quello del tutore, cioè nel documento del tutore, non compare nel campo "tokenFiglio" premendo il bottone, compare il QR code, altrimenti in caso contrario il token dell'assistito viene trovato all'interno del campo "tokenFiglio" del tutore quindi l'associazione è stata compiuta e si apre la mappa dedicata all'assistito.

3.6.6 Schermata mappa tutore e avvio del rilevamento della posizione

Quando durante l'accesso l'utente è un tutore già associato con l'assistito, ovvero ha il campo Ruolo impostato a "genitore" e il campo "tokenFiglio" diverso da un valore nullo, si apre la schermata della mappa del tutore. Per prima cosa, viene chiesto all'utente il permesso di poter utilizzare la propria posizione. Questa schermata avrà funzionalità diverse dalla mappa dell'assistito. La mappa del tutore per prima cosa carica e fa comparire la mappa sullo schermo del dispositivo. La mappa è costituita dai classici pulsanti per lo zoom di Google e dall'icona di puntamento della propria posizione che una volta premuta geolocalizza la posizione esatta, sulla mappa, del tutore impostando un marker.

Nella parte alta, in particolare nella zona destra, della schermata c'è l'icona delle Info in cui vengono spiegate all'utente tutore alcune caratteristiche dell'applicazione che sono importanti da sapere per poter utilizzare al meglio l'applicazione. Affianco alle Info c'è l'icona delle Impostazioni per poter accedere, appunto, alla schermata delle Impostazioni.

Nella parte bassa della schermata sono presenti tre bottoni, il primo colorato in rosso con scritto "*SOS*" che permette di visualizzare i contatti di emergenza salvati dalle Impostazioni.

Il secondo bottone "*Imposta raggio zona sicura*" permette, una volta premuto, di impostare il raggio in metri della zona sicura. Dopo aver impostato il raggio, comparirà un cerchio intorno alla posizione del tutore, ovvero la zona sicura. Sostanzialmente il tutore decide l'area della zona sicura.

Infine, il terzo bottone "*Dov'è mio figlio?*", che una volta premuto avvia il monitoraggio della posizione. Dopo aver premuto questo bottone sulla mappa comparirà un nuovo marker che sta ad indicare la posizione del proprio assistito. Le coordinate dell'assistito vengono prelevate da Firebase quando si preme il bottone in questione, e questa operazione viene compiuta ogni minuto per permettere un tracciamento in tempo reale dell'assistito. Nel momento in cui vengono aggiornate le coordinate dell'assistito sulla mappa e quest'ultimo dovesse trovarsi al di fuori della zona sicura viene inviata al tutore una notifica sul dispositivo che lo avverte del pericolo a causa della fuoriuscita dell'assistito dalla zona sicura. In questo caso, potranno essere attuate le varie azioni per gestire la situazione di pericolo, come ad esempio premere il bottone di "SOS" e chiamare il proprio assistito o direttamente mandare aiuto verso la sua posizione.

3.6.7 Schermata impostazioni e contatti di emergenza

La schermata delle Impostazioni viene aperta nel momento in cui viene premuta l'icona degli ingranaggi affianco all'icona delle Info. Questa schermata si presenta con la scritta "Impostazioni di emergenza" e subito sotto un pulsante con scritto "Inserisci numero di telefono in caso di wandering" che una volta premuto apre la schermata per poter aggiungere un contatto di emergenza. La schermata fa apparire due caselle dove è possibile inserire il nome del contatto di emergenza e subito sotto inserire il numero di cellulare del contatto. Ancora sotto, c'è la possibilità di premere varie opzioni per identificare il contatto che si va a creare, ad esempio si può premere l'opzione per identificare il contatto come collega di Lavoro, oppure appartenente alla propria Famiglia, oppure appartenente alla compagnia di Amici oppure infine inserire anche l'opzione Altro se non si vuole identificare il contatto di emergenza. È possibile anche non selezionare nessuna possibilità. Alla fine, compare il bottone "Salva Contatto" che una volta premuto salva il contatto di emergenza all'interno del Real-time Database su Firebase. Ogni contatto ha una propria sezione sul database con tutte le caratteristiche inserite durante la fase di salvataggio. Tutti i contatti di emergenza sono visibili all'interno del bottone di "SOS" con tutte le informazioni salvate su Firebase, inoltre in questa sezione di SOS è possibile chiamare i contatti di emergenza, chiedendo all'utente sempre i permessi per effettuare le chiamate dal cellullare, oppure semplicemente eliminare il contatto di emergenza. Questa cancellazione viene fatta anche all'interno del Real-time database su Firebase.

3.6.8 Schermata mappa assistito

In questa schermata compare la mappa dedicata all'assistito. È una schermata differente da quella del tutore, la mappa viene caricata in egual modo cosi come i pulsanti per lo zoom e l'icona per puntare la posizione dell'assistito sulla mappa tramite un marker.

Anche nella parte alta compaiono le due Icone delle Info e delle Impostazioni che hanno esattamente le stesse funzioni presenti nella mappa del tutore, ovviamente le informazioni presenti nell'icona Info sono diverse per la mappa dell'assistito rispetto a quelle presenti nella mappa del tutore.

Infatti, questa schermata non ha i tre bottoni presenti nella mappa del tutore ma ha solo il bottone rosso dell'SOS che può essere premuto dall'assistito in caso di pericolo per poter chiamare i propri cari, i propri tutori ecc. Una volta geolocalizzata la posizione dell'assistito sulla mappa, ovviamente dopo aver chiesto i permessi all'utente per rilevare la posizione, le coordinate dell'assistito sulla mappa espresse in latitudine e longitudine vengono caricate nei campi "*latitudine*" e "*longitudine*" presenti nel documento dell'assistito sul Cloud Firestore.

Quando l'assistito si muove sulla mappa può controllare in tempo reale la propria posizione e in particolare ogni trenta secondi l'applicazione carica, sempre nei campi latitudine e longitudine sul

Firestore, le coordinate dell'assistito in modo che quest'ultime possono essere prelevate poi dal tutore. Questa funzionalità permette di avere una localizzazione quasi in tempo reale dell'assistito. Ovviamente, l'assistito non vede sulla propria mappa la zona sicura impostata dal tutore.

Capitolo 4 – Implementazione

All'interno di questo capitolo vado a spiegare come ho implementato l'intera applicazione, in particolare partendo dall'acquisizione GPS per poter rilevare la posizione degli utenti, l'implementazione della mappa e ovviamente le varie schermate spiegate nel capitolo 3.

Spiego come avviene il monitoraggio della posizione e la creazione della zona sicura, come ho implementato il modulo per l'associazione tra tutore e assistito, il modulo per l'autenticazione dell'utente e il modulo per creare, salvare ed eliminare i contatti di emergenza.

Alla fine di questo capitolo, spiego quali permessi devono essere attivati affinché l'applicazione funzioni al meglio.

4.1 GPS e Mappe

4.1.1 Acquisizione GPS

Un aspetto fondamentale della mia applicazione è proprio il meccanismo di acquisizione del GPS per poter avviare poi tutto il servizio di wandering detection.

All'interno della mia applicazione per poter attivare il servizio di geolocalizzazione ho deciso di utilizzare il plugin *geolocator* come mostrato in Figura 16:

Figura 16 - Il plugin geolocator con rispettiva versione

Il plugin *geolocator*[27] è un plugin di geolocalizzazione di Flutter che fornisce un facile accesso ai servizi di localizzazione in base alla piattaforma che si sta utilizzando. Utilizzando una piattaforma Android. viene utilizzata la classe LocationManager che a sua volta viene utilizzata da *geolocator*. LocationManager[28] è una classe che fornisce l'accesso dei servizi di localizzazione del sistema.

Grazie a questo plugin sono riuscito all'interno del codice e dell'applicazione ad ottenere la posizione corrente del dispositivo, una ricezione continua di aggiornamenti sulla posizione e soprattutto controllare se l'utente abbia attivato i servizi di localizzazione.

[27] https://pub.dev/packages/geolocator
[28] https://developer.android.com/reference/android/location/LocationManager

Quindi, dopo aver inserito nel file pubspec.yaml la dipendenza di *geolocator*, come mostrato in Figura 16, ho dovuto aggiungere alcuni permessi importanti affinché potessi realizzare la funzionalità della geolocalizzazione. In particolare, per poter attivare e inserire questi permessi bisogna andare nel file di Android chiamato *AndroidManifest.xml*. La Figura 17 mostra i permessi che ho inserito nel codice.

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

Figura 17 - Autorizzazioni per servizi di localizzazione

Ho deciso di utilizzare queste due autorizzazioni, perché l'ACCESS_FINE_LOCATION richiede di accedere alle informazioni di localizzazione in maniera precisa, da qui la parola FINE, utilizzando il sistema GPS. A questo, ho aggiunto anche l'autorizzazione ACCESS_COARSE_LOCATION che permette di utilizzare i servizi di rete come ad esempio il Wi-Fi per calcolare la posizione, meno precisa rispetto al FINE, del dispositivo.

A partire da Android 10[27] è necessario aggiungere l'autorizzazione ACCESS_BACKGROUND_LOCATION se si vuole continuare a ricevere aggiornamenti quando l'applicazione è in esecuzione in background. Nella mia applicazione ho deciso di inserirla e la Figura 18 mostra proprio le tre autorizzazioni importanti da inserire per usufruire al meglio del plugin *geolocator*.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
```

Figura 18 - Autorizzazioni complete

Dopo aver completato queste importanti configurazioni, all'interno del codice sono riuscito ad usufruire del plugin *geolocator*. Nel codice per poter implementare tutto questo, all'interno del file Dart dove ho implementato l'acquisizione della posizione e la mappa (che spiego nel paragrafo successivo), ho creato una funzione asincrona chiamata _determinePosition() (che viene attivata nel momento in cui viene premuta l'icona di puntamento della posizione) al cui interno ho definito dapprima i vari permessi che vengono chiesti all'utente per poter utilizzare la posizione GPS del dispositivo, utilizzando la classe Geolocator[29] che mette a disposizione la funzione isLocationServiceEnabled() per verificare se c'è l'autorizzazione nel dispositivo a rilevare la propria posizione. Ho definito quindi un oggetto *permission*, di tipo LocationPermission, che mi permette di effettuare alcuni controlli, offerto sempre dalla classe Geolocator. Però, prima di verificare questo, la classe Geolocator mette a disposizione le funzioni checkPermission e requestPermission, rispettivamente il primo serve per controllare se l'utente ha autorizzato l'applicazione ad accedere alla posizione e il secondo indica se l'utente ha concesso l'autorizzazione ad accedere alla posizione del dispositivo. Fatto questo, dopo che l'utente ha

[29] https://pub.dev/documentation/geolocator/latest/geolocator/Geolocator-class.html

accettato i servizi di localizzazione, all'oggetto *permission* viene assegnata la funzione getCurrentPosition() che fornisce la posizione dell'utente.

4.1.2 Google Maps Flutter

Oltre all'acquisizione della posizione tramite GPS, ho implementato le mappe per vedere all'interno di quest'ultime la propria posizione. Per prima cosa, ho utilizzato il plugin *google_maps_flutter* di Flutter per implementare le mappe all'interno del mio codice e dell'applicazione. Ho inserito questo plugin come dipendenza all'interno del mio file pubspec.yaml come mostrato in Figura 19.

```
google_maps_flutter: ^2.2.3
```

Figura 19 - Il plugin google_maps_flutter con rispettiva versione

Il plugin *google_maps_flutter*[30] fornisce un widget di Google Maps per un'applicazione Android. Affinché si possa utilizzare questo plugin bisogna definire dei permessi importanti. Per prima cosa ho dovuto creare un'API Key per utilizzare le mappe su Android. In particolare, ho effettuato la registrazione all'interno della pagina Google Developers Console, dove ho inserito il mio progetto dell'applicazione in modo tale da poter attivare su quest'ultimo l'API Key di Google Maps. Ho attivato quindi il Google Maps SDK[31] per la mia applicazione. Dopo aver preso la mia API Key, ho definito i permessi nel codice per poter usufruire delle mappe. All'interno dell'*AndroidManifest.xml* ho aggiunto la mia API Key precedentemente creata. Dopo aver fatto queste operazioni, ho implementato la mappa all'interno del codice che viene rappresentata da un oggetto di tipo GoogleMap.

4.2 Implementazione schermata principale

Questa è la schermata che l'utente, sia tutore che assistito, visualizza quando l'applicazione viene aperta. Ho deciso di implementare un widget Scaffold con sfondo bianco. All'interno dello Scaffold ho poi implementato un widget di tipo Text con la scritta "Benvenuto!" e subito sotto due bottoni di tipo RoundedBotton, uno con la scritta "Accedi" che una volta premuto, tramite un Navigator.pushNamed apre la pagina per effettuare il login/accesso all'applicazione. Il secondo, invece, con la scritta "Registrati" e utilizzando sempre il Navigator.pushNamed apre la pagina per effettuare la registrazione.

La scelta dei colori è ricaduta su tipi di colore accesi e ben visibile proprio perché l'applicazione viene usata da bambini e anche da persone anziane, per rendere l'interfaccia più usabile anche dal loro punto di vista.

[30] https://pub.dev/packages/google_maps_flutter
[31] https://developers.google.com/maps/documentation/android-sdk/overview

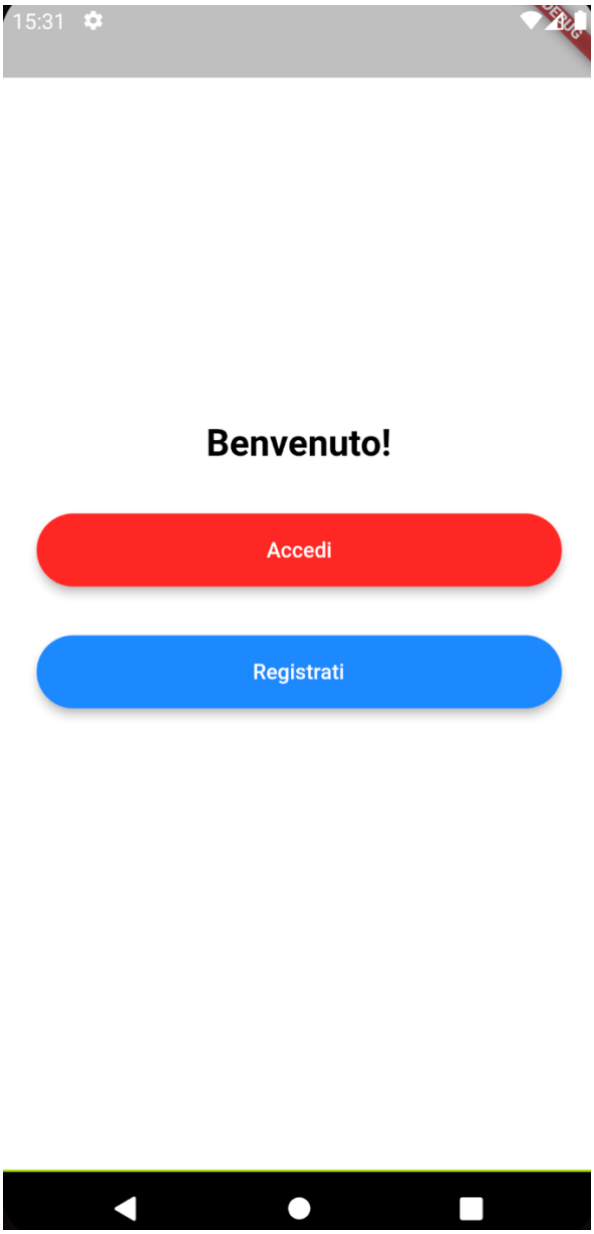


Figura 20 - La schermata principale

4.3 Implementazione schermata registrazione utente

Questa schermata permette all'utente di registrarsi all'applicazione. Per prima cosa ho implementato i vari TextEditingController dei vari campi per la registrazione, che permettono di creare un campo di testo modificabile se inseriti all'interno di un TextField.

Tutti questi campi vengono inseriti all'interno di un widget Scaffold, con in cima una scritta che informa l'utente di inserire i dati all'interno dei campi per potersi registrare. Ovviamente i campi devono essere tutti compilati.

Subito dopo ho creato una funzione asincrona getToken() che grazie alla classe FirebaseMessagging e alla funzione getToken() permette di prelevare il token del dispositivo dell'utente perché deve essere poi salvato in Firestore.

Ho implementato, poi, una funzione asincrona signUp(), che grazie all'utilizzo della classe FirebaseAuth crea un istanza della funzione createUserWithEmailAndPassword() per salvare l'email e la password degli utenti, che si registrano, all'interno della schermata Autentication di Firebase. Sempre all'interno di signUp() viene richiamata la funzione per prelevare il token ovvero getToken() e subito dopo la funzione addUserDetails che permette di aggiungere tutte le informazioni, che l'utente inserisce nei campi di registrazione, all'interno del Cloud Firestore ovvero nella raccolta *utenti* discussa nel Capitolo 3.

Nella funzione addUserDetails ho inserito due specifici controlli per verificare se nel campo Ruolo è stata inserita la stringa "genitore" o la stringa "figlio" per capire se chi si sta registrando è il tutore o l'assistito. Nel caso del tutore e quindi del "genitore" l'applicazione, oltre a salvare le informazioni nei campi che vengono visualizzati dall'utente, salva di default un campo "token" dove viene inserito il token del dispositivo prelevato dalla funzione getToken() e un campo "tokenFiglio" a cui viene assegnata la stringa "nonAssociato". Mentre, nel caso dell'assistito, che ha il campo Ruolo impostato a "figlio", oltre a salvare le informazioni nei campi che vengono visualizzati dall'utente, salva di default un campo "token" dove viene inserito il token del dispositivo prelevato dalla funzione getToken() e due campi "latitudine" e "longitudine" che successivamente verranno riempiti.

Alla fine ho implementato un bottone di tipo RoundedButton con la scritta "Registrati" che una volta premuto attiva la funzione signUp() e permette la registrazione dell'utente.

Premendo "Registrati" oltre ad attivare la funzione signUp(), viene aperta tramite un Navigator.pushNamed la pagina per effettuare l'accesso all'applicazione.

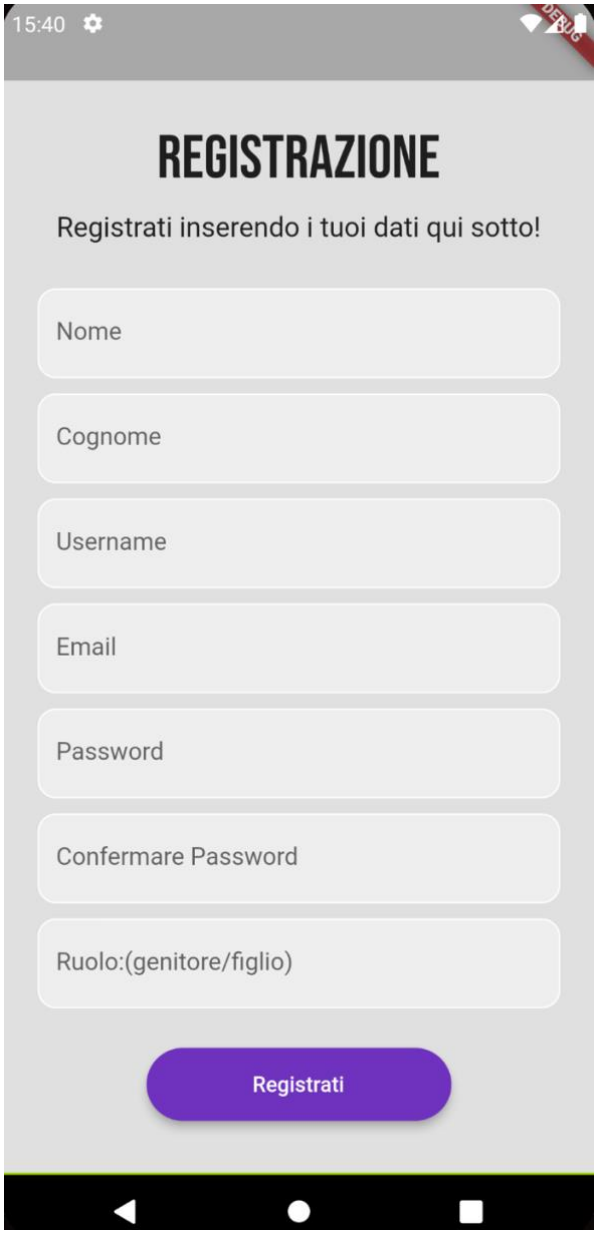


Figura 21 - Schermata Registrazione

4.4 Implementazione schermata accesso utente

Questa schermata permette all'utente di poter accedere all'applicazione ovviamente dopo essersi registrato.

Per prima cosa ho creato due TextEditingController per l'email e la password che l'utente deve inserire per poter accedere all'applicazione. Il TextEditingController l'ho associato poi a un TextField per creare un vero e proprio campo di testo modificabile.

Ho creato poi una funzione asincrona signIn(), che prende l'email e la password inserite dall'utente e crea un oggetto userCredential di tipo UserCredential, che è una classe contenente l'autenticazione di un utente identificato da email e password, a cui ho assegnato l'istanza della

classe FirebaseAuth, chiamata signInWithEmailAndPassword() che permette all'utente di autenticarsi prendendo l'email e la password inseriti e controllando quest'ultime nella sezione Autentication di Firebase.

All'interno del codice ho creato funzioni asincrone come getRuoloFromDatabase(), getTokenFromDatabase(), getTokenFiglioFromDatabase() che implementano tutte delle vere e proprie query ovvero delle interrogazioni a Firebase.

In particolare getRuoloFromDatabase() prende in input l'email dell'utente che accede e effettua una query a Firestore per ricercare all'interno della collezione *utenti*, l'utente che ha la stessa email che è stata passata in input alla funzione e infine restituisce il campo "ruolo" di quel determinato utente.

In particolare getTokenFromDatabase() prende in input l'email dell'utente che accede e effettua una query a Firestore per ricercare all'interno della collezione *utenti*, l'utente che ha la stessa email che è stata passata in input alla funzione e infine restituisce il campo "token" di quel determinato utente.

In particolare getTokenFiglioFromDatabase() prende in input l'email dell'utente che accede e effettua una query a Firestore per ricercare all'interno della collezione *utenti*, l'utente che ha la stessa email che è stata passata in input alla funzione e infine restituisce il campo "tokenFiglio" di quel determinato utente.

Grazie all'implementazione di queste funzioni, si aprono due strade una dedicata al tutore e l'altra dedicata all'assistito che vengono controllate sempre all'interno della funzione signIn(). La prima strada, ovvero quella dedicata al tutore, si apre nel momento in cui l'utente che effettua l'accesso ha il campo "ruolo" impostato a "genitore", quindi è un tutore e per verificare questo ho utilizzato la funzione getRuoloFromDatabase(). Se il tutore in questione, che ha effettuato l'accesso, ha il campo "tokenFiglio", recuperato grazie alla funzione getTokenFiglioFromDatabase(), diverso dalla stringa "nonAssociato" vuol dire che ha già effettuato l'associazione e quindi tramite un Navigator.pushNamed si aprirà la mappa relativa al tutore e viene passata l'email del tutore come argomento. Invece, se il campo "tokenFiglio" del tutore dovesse avere come valore la stringa "nonAssociato" tramite un Navigator.pushNamed si aprirà la schermata associazione e viene passato l'oggetto userCredential come argomento.

La seconda strada, invece, quella dedicata all'assistito si apre nel momento in cui l'utente che effettua l'accesso ha il campo "ruolo" impostato a "figlio", quindi è l'assistito. Viene recuperato il token dell'assistito grazie alla funzione getTokenFromDatabase() e un Navigator.pushNamed aprirà la schermata token passando il token dell'assistito come argomento.

Dopo aver fatto tutto questo e aver inserito i campi dell'email e della password, l'utente deve premere il bottone, di tipo RoundedBotton, con scritto "Accedi". Una volta premuto, viene attivata la funzione signIn().

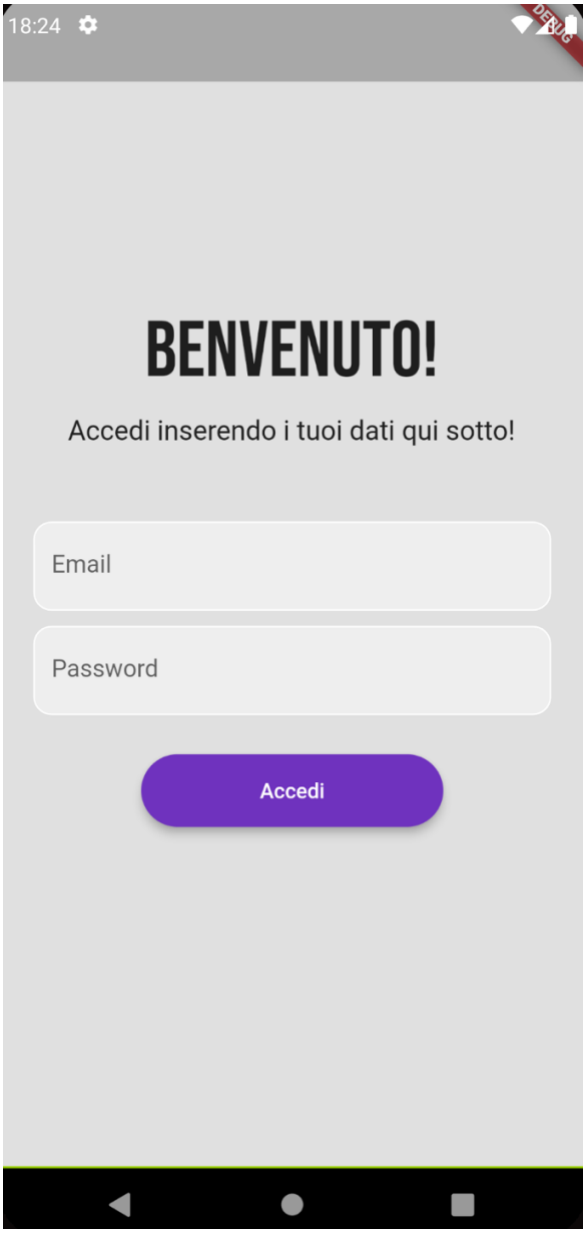


Figura 22 - Schermata di accesso

4.5 Implementazione schermata associazione

Questa schermata, come detto in precedenza, si apre se l'utente che compie l'accesso è un tutore che non si è ancora associato con l'assistito. La schermata dell'associazione è importante perché permette di associare il tutore e l'assistito tramite i token del dispositivo.
A questa schermata ho passato come costruttore l'oggetto userCredential di tipo UserCredential, che si porta dietro l'email e la password dell'utente che ha effettuato l'accesso.

Ho implementato poi una funzione getTokenFromDatabase() che prende in input l'email dell'utente che accede (utilizzando l'oggetto userCredential) e effettua una query a Firestore per ricercare all'interno della collezione *utenti*, l'utente che ha la stessa email che è stata passata in input alla funzione e infine restituisce il campo "token" di quel determinato utente.

Per prima cosa ho implementato la funzione per scannerizzare il QR code presente sul telefono dell'assistito. Per fare questo ho creato una funzione asincrona _showQRScanner(). All'interno di questa funzione a una variabile stringa di nome barcodeScanRes ho assegnato la funzione scanBarcode() della classe FlutterBarcodeScanner, che è una classe che permette di implementare le funzionalità per scannerizzare QR Code ma anche Bar Code. Ovviamente, ho settato solo l'opzione per scannerizzare i QR Code utilizzando ScanMode.QR. Infatti, il QR Code sul telefono dell'assistito conterrà il suo token che viene poi assegnato alla variabile barcodeScanRes.

Sempre all'interno della funzione _showQRScanner(), se la scannerizzazione del QR Code va a buon fine viene effettuata una query a Firestore. Questa query interroga Firestore per cercare all'interno della collezione *utenti* l'utente che ha la stessa email che si trova nell'oggetto userCredential. Fatto questo, se il documento dell'utente prelevato non è vuoto, viene effettuata una nuova query a Firestore, in particolare viene chiesto, sempre a Firestore, di cambiare tramite un "update" il campo "tokenFiglio" dell'utente prelevato poc'anzi, che è il tutore, con la variabile barcodeScanRes che contiene la stringa del token dell'assistito. Fatto questo, tramite una chiamata HTTP POST e grazie a Firebase Messagging vengono inviate due notifiche sui dispositivi del tutore e dell'assistito che gli avvertono dell'associazione avvenuta con successo e che quindi sono associati. L'invio delle notifiche viene fatto grazie all'utilizzo dei token dei due dispositivi che, ovviamente, vengono prima prelevati tramite la funzione getTokenFromDatabase().

La schermata dell'associazione si apre con un AppBar con la scritta "Associazione" in riferimento alla schermata. Al centro ho inserito un bottone di tipo RoundedBotton con scritto "Scannerizza QR Code" che una volta premuto attiva la funzione _showQRScanner().

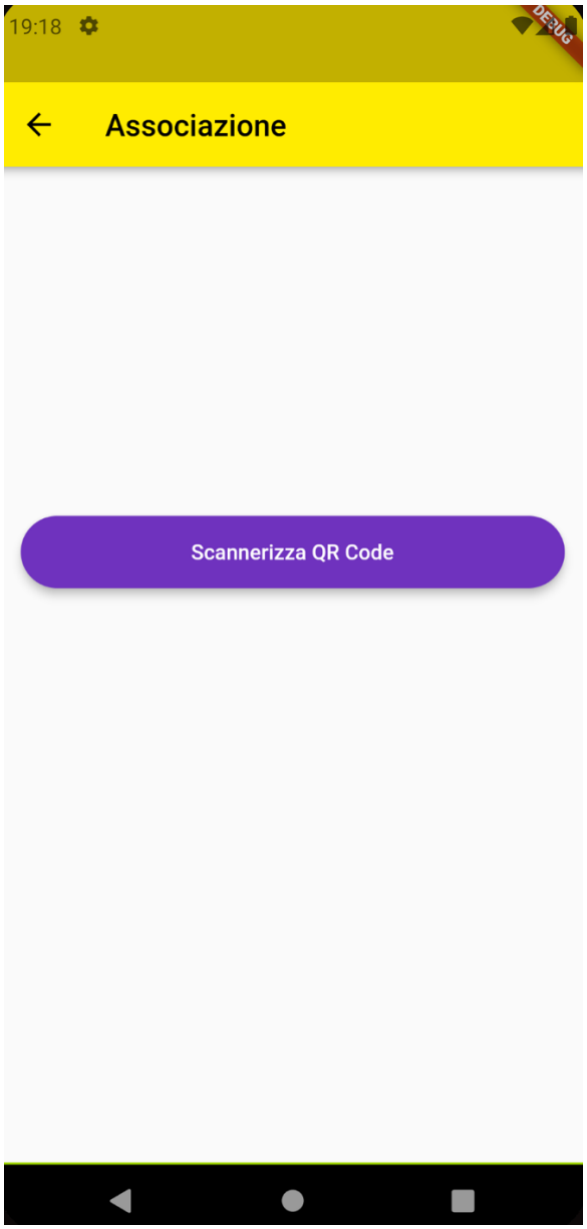


Figura 23 - Schermata associazione

4.6 Implementazione schermata token

Questa schermata si apre quando l'utente che effettua l'accesso all'applicazione è un assistito, ovvero ha il campo "ruolo" settato a "figlio". La schermata è fondamentale perché permette di generare il QR Code per permettere l'associazione tra tutore e assistito.

Alla schermata ho passato come costruttore la stringa "token", dalla schermata dell'accesso, che contiene il token del dispositivo dell'assistito.

All'interno della schermata, ho implementato fin da subito una funzione asincrona checkToken(). Questa funzione per prima cosa effettua una query a Firestore. Questa query deve trovare all'interno della collezione *utenti* il documento dell'utente che ha il campo "ruolo" settato a "genitore" e il

campo “tokenFiglio” uguale al token passato come costruttore alla pagina che sarebbe il token dell’assistito. Se questa query va a buon fine, tramite un Navigator.pushNamed si apre la mappa relativa all’assistito e viene passato il token dell’assistito come argomento. Questa funzione checkToken() ha come obiettivo di effettuare un controllo.

Infatti, sempre all’interno della schermata, ho implementato un’AppBar con scritto “Token” da qui il nome della schermata. Subito sotto, ho impostato un messaggio che da all’assistito un’informazione ovvero, premendo il bottone posto subito sotto, sé l’assistito è gia associato a un tutore verrà indirizzato alla sua mappa altrimenti comparirà il QR Code per l’associazione.

Il bottone che si trova sotto, è un bottone di tipo RoundedButton con scritto “Genera QR Code” che una volta premuto attiva la funzione per il controllo checkToken(). Se la funzione checkToken() dovesse dare esito negativo compare un QR Code che ha come parametro “data” il token dell’assistito.

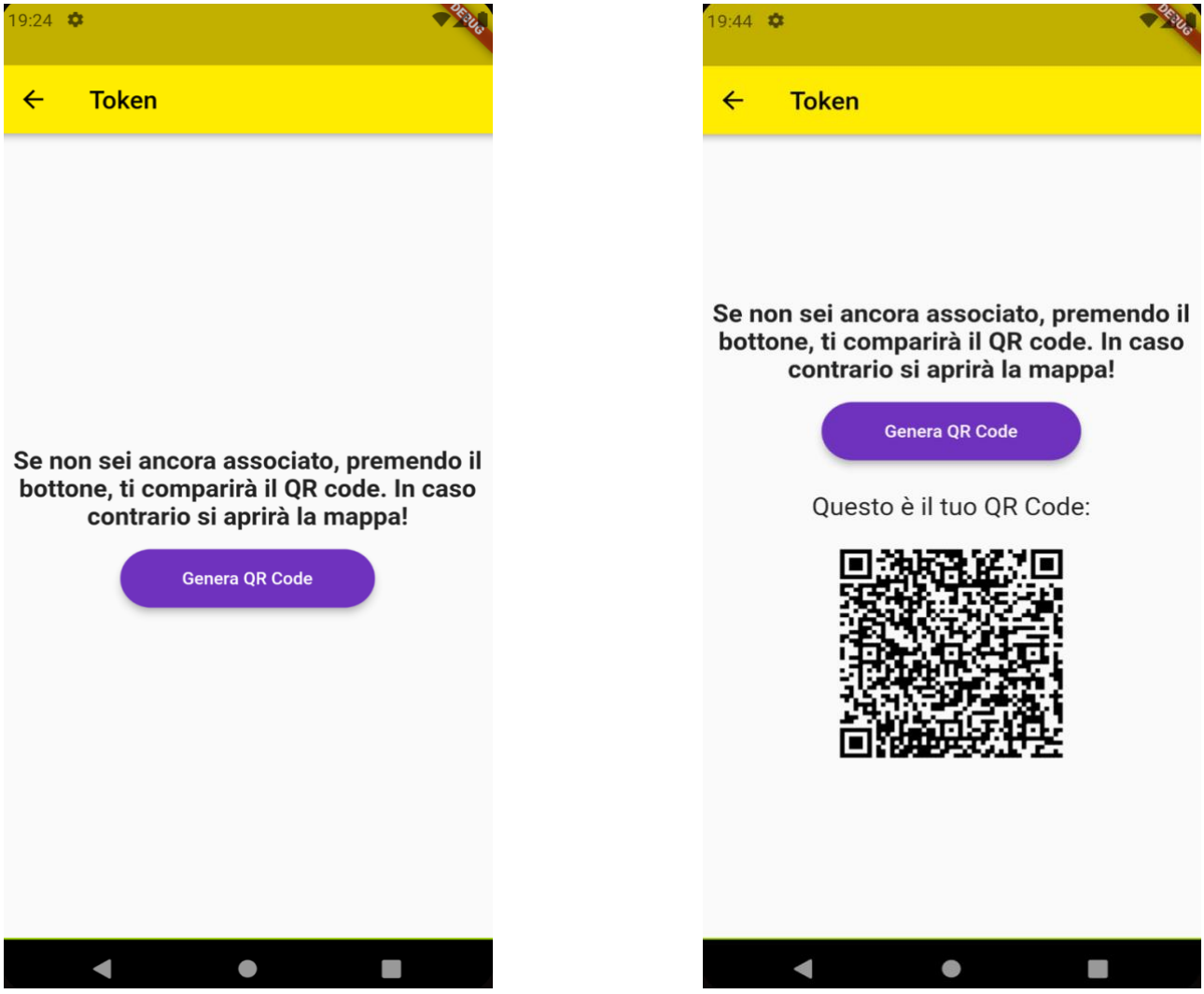


Figure 24 - Schermata token e schermata token con generazione del QR Code

4.7 Implementazione schermata mappa tutore e avvio del rilevamento della posizione

Questa schermata si apre quando l'utente che accede all'applicazione è un tutore associato, ovvero un utente che ha il campo "ruolo" impostato a "genitore" e il campo "tokenFiglio" impostato con il token dell'assistito.

Nel mio codice ho deciso di creare un file Dart chiamato "my_page" al cui interno ho inserito tutte le caratteristiche che si vedono all'interno della schermata, come la mappa, le due icone Info e Impostazioni e i tre bottoni in basso. Come mostrato in Figura 26.

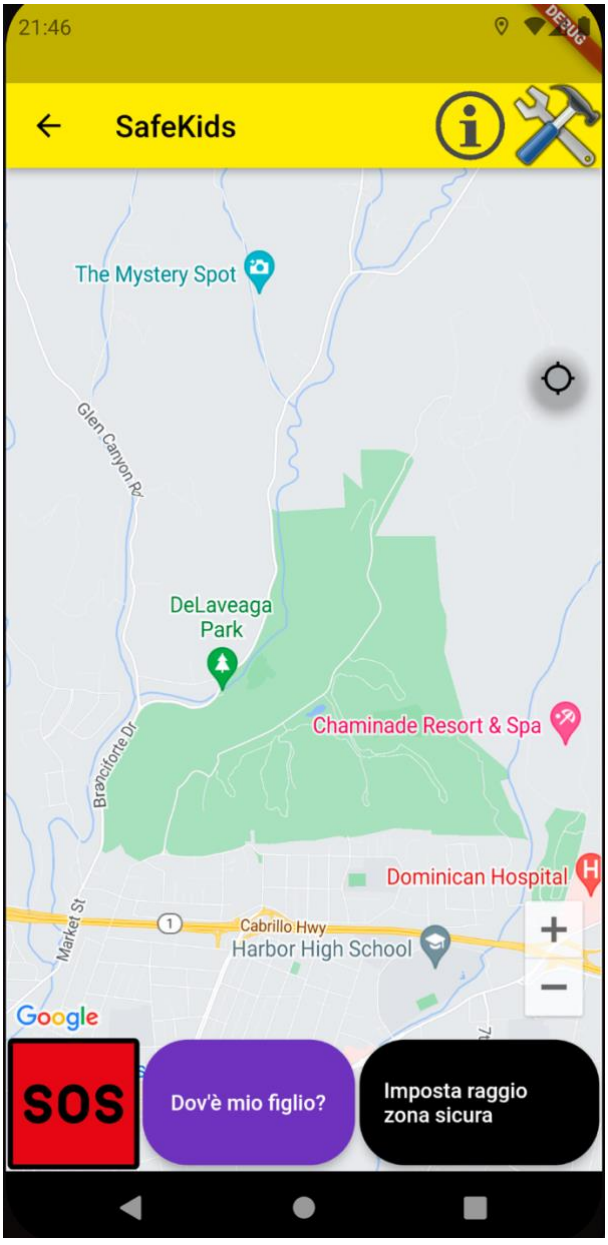


Figura 25 - Schermata relativa al tutore

Nella pagina "my_page" passo come costruttore l'email del tutore derivante dal processo di autenticazione all'applicazione. Subito dopo, ho implementato la funzione asincrona

getTokenSonFromDatabase() che prende in input l'email del tutore e effettua una query a Firestore. Questa query consiste nel trovare all'interno di Firestore nella collezione *utenti,* l'utente con la stessa email passata in input alla funzione, quindi recuperare il documento del tutore e in particolare restituire il campo "tokenFiglio" del tutore. Questo valore del campo "tokenFiglio" recuperato dalla funzione viene salvato all'interno di una variabile che ho chiamato "tokenSon" per comodità.

Per la schermata della mappa del tutore ho deciso di utilizzare uno Scaffold al cui interno ho inserito un'AppBar con il nome della mia applicazione ovvero "**SafeKids**". All'interno della stessa AppBar, nella parte a destra, tramite un widget Row ho inserito un'icona delle Info. Quest'ultima è un bottone, principalmente un InkWell che ha la particolarità di colorarsi in base alla pressione del dito. All'icona delle Info ho inserito nell'imagePath una classica immagine di una "i" cerchiata che sta a rappresentare le informazioni dell'applicazione, infatti una volta premuta l'icona si apre una showDialog in cui ho inserito tutte le informazioni per poter usare al meglio l'applicazione. Affianco all'icona delle Info, ho inserito l'icona delle Impostazioni che descrivo più avanti.

All'interno dello Scaffold della schermata, ho inserito un body con un widget Stack che a sua volta contiene un Container al cui interno richiamo la classe "MyMap" in cui ho implementato la mappa del tutore, passando la variabile tokenSon come argomento. Subito sotto ho inserito un altro Container al cui interno ho inserito il bottone di "SOS" che descrivo più avanti.

Nella classe "MyMap" del file Dart "MyMap.dart" ho implementato la mappa relativa al tutore. Per prima cosa la classe ha come costruttore la variabile stringa "tokenSon" passata come argomento dalla classe "my_page". Per prima cosa ho dichiarato un oggetto googleMapController di tipo GoogleMapController[32] che permette di definire un controller, in questo caso googleMapController, per ogni istanza di GoogleMap in esecuzione sul dispositivo. Poi ho definito, usando la classe CameraPosition, la posizione iniziale che il tutore vede nel momento in cui si apre la mappa specificando un punto sulla mappa a caso, definito da LatLng. Subito dopo ho definito una collezione di oggetti di tipo Set<Marker> chiamata "markers", che sta a rappresentare un insieme di punti unici sulla mappa, infatti il marker rappresenta un punto unico e specifico sulla mappa. Ho definito anche un insieme di cerchi di tipo Set<Circle> chiamata "circles", che rappresenta una serie di cerchi che possono essere inseriti sulla mappa. All'inizio ho definito il raggio del cerchio uguale a 1 metro per comodità in modo tale da non farlo vedere all'utente.

Dopo aver fatto questo, ho definito una funzione asincrona getTokenPadreFromDatabase() dove viene creata una variabile "token" a cui viene assegnato il valore del costruttore "tokenSon" contenente il token dell'assistito e viene effettuata una query a Firestore per ricercare all'interno della collezione *utenti*, l'utente che ha la stesso valore del campo "tokenFiglio" uguale alla variabile "token" e infine restituisce il campo "token" di quel determinato utente che sarebbe il tutore, in poche parole restituisce il token del tutore.

Per implementare la mappa ho creato un widget Stack al cui interno ho inserito il widget GoogleMap che implementa la mappa di GoogleMaps. La mappa è costituita dai pulsanti per effettuare lo zoom e ho implementato l'icona per puntare la posizione, che una volta premuta attiva la funzione _determinePosition() per centrare la posizione dell'utente. A questo viene segnato anche un marker per indicare la posizione dell'utente e viene disegnato un cerchio sulla posizione

[32] https://www.html.it/pag/397771/flutter-gestione-delle-mappe/

dell'utente che non sarà visibile perché avente raggio pari a 1 metro. Il cerchio è indicato da un oggetto "circle" di tipo Circle che permette non solo di creare cerchi sulla mappa ma anche di modificarli a proprio piacimento. Per modificare il raggio del cerchio ho implementato un bottone di tipo RoundedButton all'interno di un Container, con scritto "Imposta raggio zona sicura" che una volta premuto fa comparire una showDialog con un campo di testo in cui il tutore inserisce il raggio della zona sicura in metri. Come mostrato qui sotto:

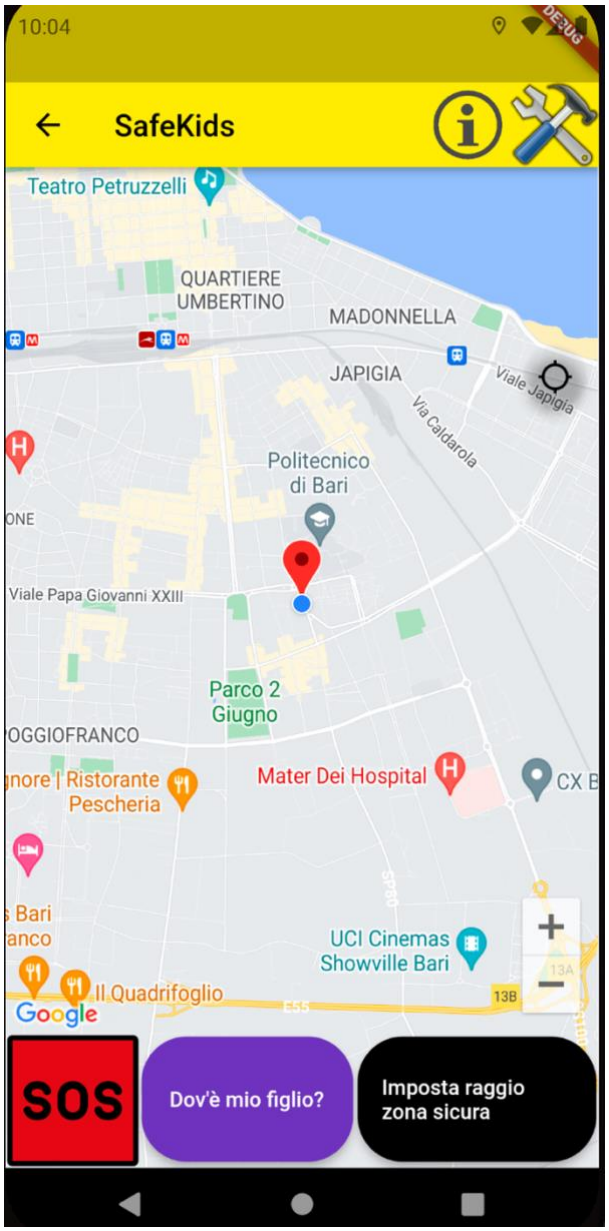


Figura 27 - Marker sulla posizione del tutore

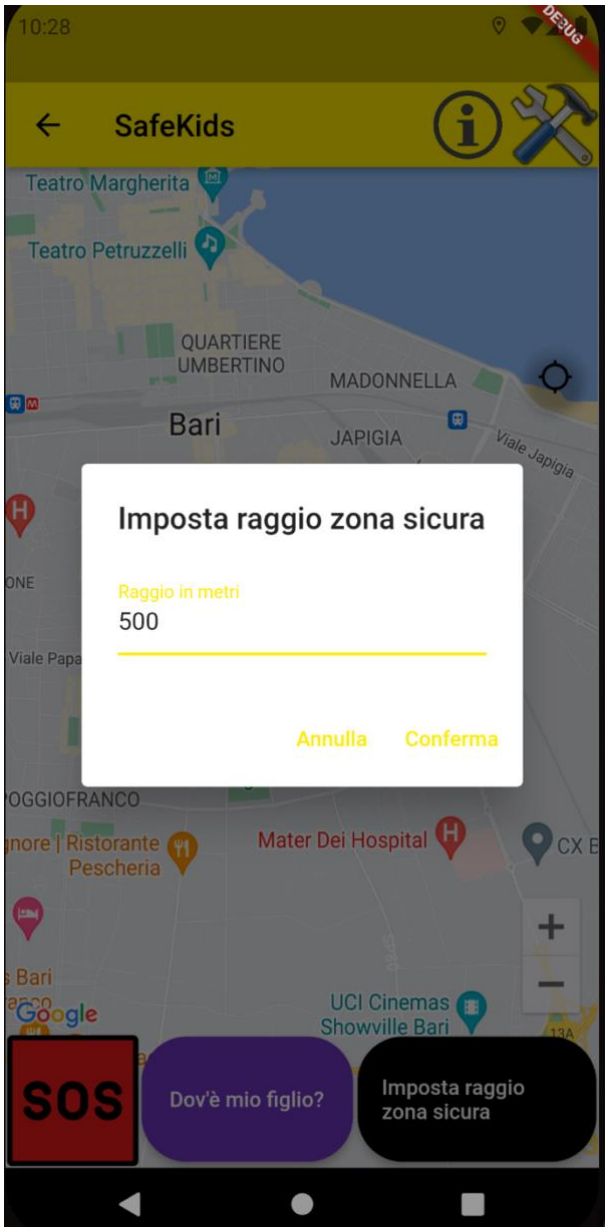


Figura 28 – Bottone per impostare la zona sicura, ad esempio di 500 metri

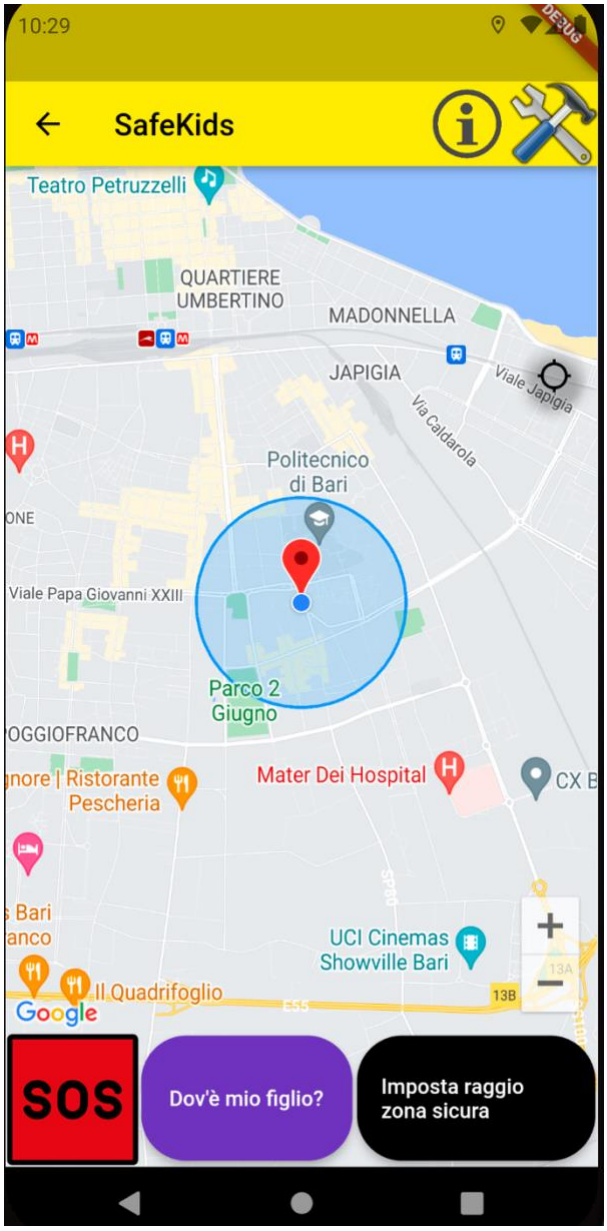


Figura 29 – Creazione della zona sicura con un raggio di 500 metri

Una volta individuata la posizione del tutore sulla mappa tramite la funzione _determinePosition(), dopo aver segnato con un marker e con un cerchio la posizione del tutore sulla mappa, per avviare il wandering detection ho creato una funzione asincrona findMySon() che per prima cosa effettua una query a Firestore per ricercare all'interno della collezione *utenti* l'utente che abbia il valore del campo "token" uguale al valore della variabile tokenSon, quindi in poche parole cercare il proprio assistito, infatti viene recuperato il documento dell'assistito che si è associato con il tutore. Fatto questo, vengono prelevate dal documento del proprio assistito le coordinate geografiche, sempre dell'assistito, specificate nei campi "latitudine" e "longitudine" e vengono segnate sulla mappa con un marker. Poi ho creato una variabile di tipo numerica chiamata "distance" a cui ho assegnato la funzione computeDistanceBetween, della classe SphericalUtil della libreria "map_tool", che permette di calcolare la distanza tra punti geografici, nel mio caso per calcolare la distanza tra la posizione dell'assistito e il centro del cerchio impostato sulla posizione del tutore. Calcolata questa

distanza, se quest'ultima è maggiore del raggio del cerchio allora vuol dire che l'assistito si trova al di fuori della zona sicura e quindi viene inviata una notifica di allerta sul dispositivo del tutore. Altrimenti se è minore, vuol dire che l'assistito si trova nella zona sicura.

Per implementare il bottone "Dov'è mio figlio?" di tipo RoundedButton ho utilizzato un Container. Nel momento in cui viene premuto il bottone viene attivata la funzione findMySon() si avvia il wandering detection.

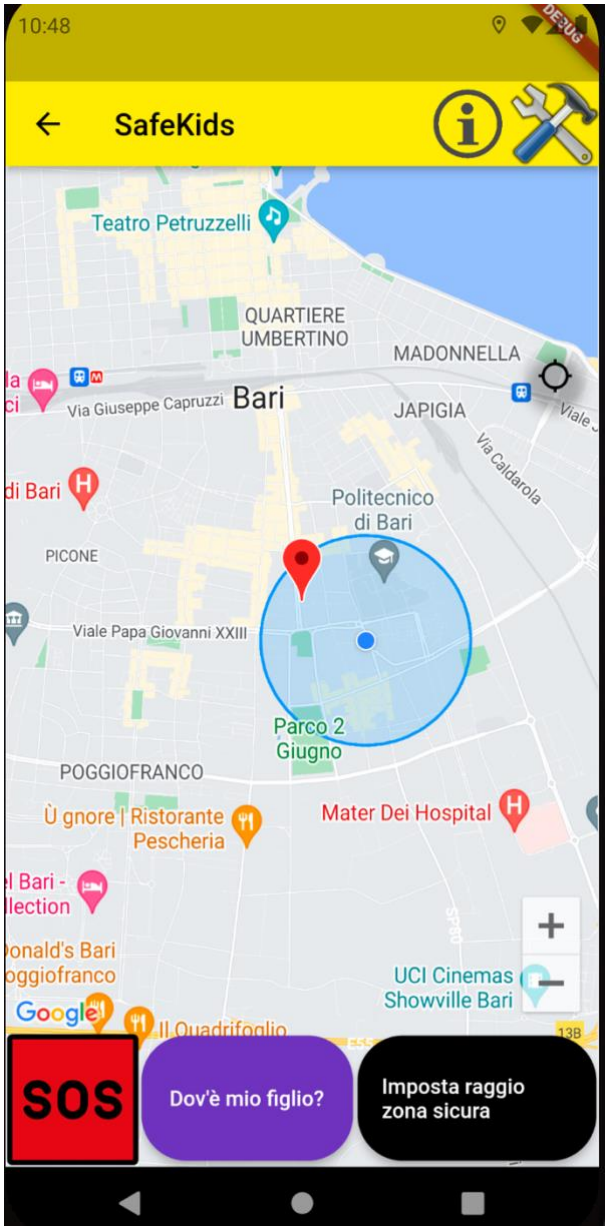


Figura 30 – Marker sulla mappa che indica la posizione dell'assistito, dopo aver premuto "Dov'è mio figlio?"

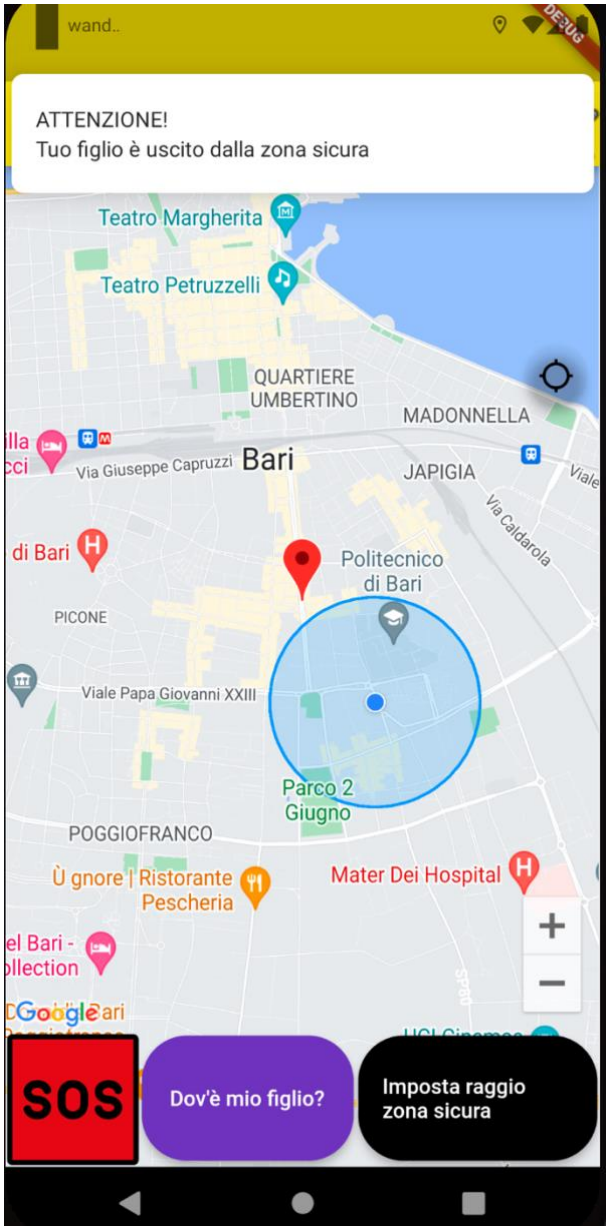


Figura 31 – L'assistito esce dalla zona sicura e viene inviata una notifica di allerta sul dispositivo del tutore

4.8 Implementazione schermata impostazioni e contatti di emergenza

In questa schermata ho implemento le impostazioni dell'applicazione. In particolare, l'icona delle Impostazioni è un widget di tipo InkWell che ha come imagePath l'immagine di un cacciavite e un martello per indicare appunto le impostazioni dell'applicazione. Ho inserito poi questo bottone all'interno di una classe chiamata "SettingsPage". Una volta premuto il bottone si apre la schermata delle Impostazioni. Questa schermata è costituita da un widget Scaffold al cui interno c'è un AppBar con scritto "Impostazioni App" e subito sotto ho inserito un widget di tipo ListView con un Container con scritto "Impostazioni di emergenza" e subito sotto un altro Container con all'interno un widget/pulsante di tipo InkWell con scritto "Inserisci numero di telefono in caso di wandering".

Figura 32 – Schermata Impostazioni

La schermata Impostazioni è uguale sia per assistito che tutore.

Una volta premuto il pulsante “Inserisci numero di telefono in caso di wandering” attraverso un Navigator.push si apre la schermata per creare e salvare i contatti di emergenza.

Per la schermata per la creazione e il salvataggio dei contatti di emergenza, ho implementato un’AppBar con la scritta “Salva Contatti”, subito sotto tramite una Column ho inserito per primo il campo di testo per inserire il nome del contatto affiancato da un’icona con un omino e subito sotto un altro campo di testo per inserire il numero di telefono del contatto. Ancora sotto, ho implementato attraverso dei widget di tipo _buildContactType delle scelte che l’utente può inserire per rappresentare il contatto che sta creando. Ad esempio, può identificare il contatto come persona legata al lavoro, oppure un parente, un amico e infine può scegliere “altro” per non specificare nulla. Per ultimo, un bottone di tipo ElevatedButton con scritto “Salva Contatto”.

Per salvare i contatti di emergenza, ho utilizzato la classe DatabaseReference che mi ha permesso di creare, all’interno del Real-time Database di Firebase, un database chiamato “Contacts”. Questo database Contacts accetta coppie di valori costituite da chiave e valore associato. Infatti, quando viene premuto il bottone “Salva Contatto” viene attivata la funzione saveContact() che prende i valori inseriti nei due campi di testo “nome” e “numero” e li salva all’interno del database “Contacts” di Firebase. Viene creato un oggetto chiamato “contact” di tipo Map<String,String> che rappresenta una struttura dati come le HashMap che permettono di associare a una chiave un valore, nel mio caso i campi nome, numero e tipo (famiglia, lavoro, amici o altro).

Figura 33 – Schermata creazione e salvataggio contatti

Nella schermata della mappa, è presente il bottone di “SOS” che una volta premuto permette di far comparire la schermata con i contatti di emergenza salvati in precedenza dalle Impostazioni. Per implementare questa schermata ho utilizzato sempre la classe DatabaseReference che mi ha permesso di recuperare il database “Contacts” creato in precedenza. Quindi vengono visualizzati tutti i contatti di emergenza creati. Oltre a mostrare il nome e il numero del contatto di emergenza, ho inserito un piccolo pulsante con l’icona di un cellulare che una volta premuto fa partire la chiamata. Ovviamente, vengono chiesti all’utente i permessi per poter effettuare chiamate. Affianco al pulsante del cellulare per effettuare chiamate, ho inserito un altro pulsante con la scritta “Elimina” che permette di eliminare definitivamente il contatto dall’applicazione ma anche dal database “Contacts”.

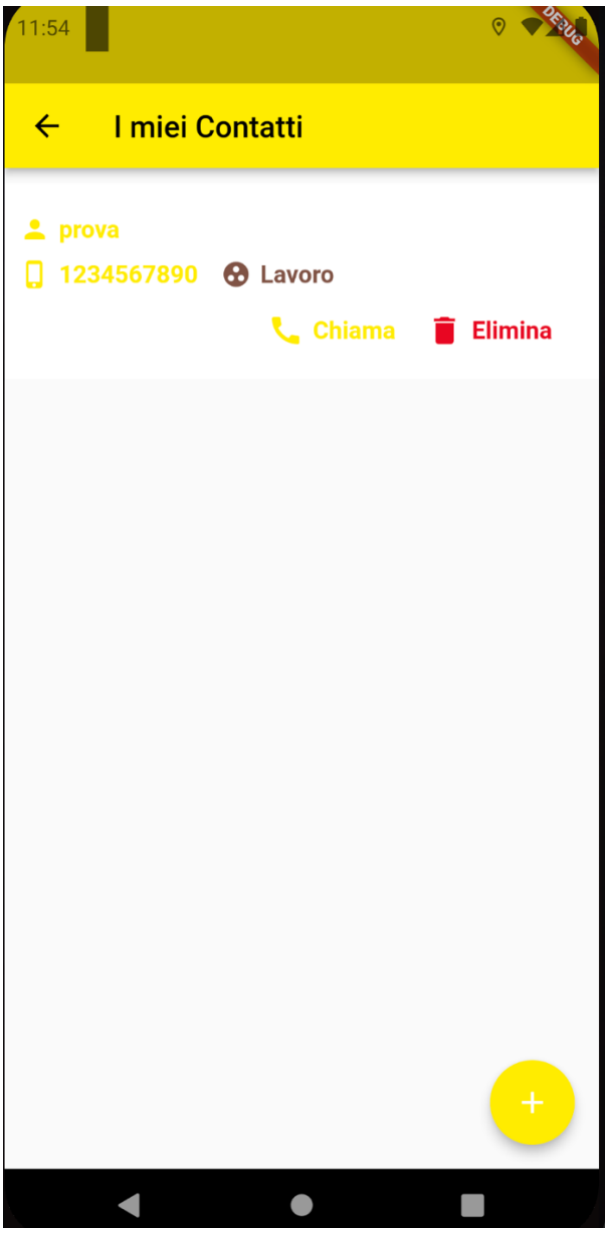


Figura 34 – Schermata per visualizzare, chiamare ed eliminare i contatti di emergenza

4.9 Implementazione schermata mappa assistito

In questa schermata ho implementato la mappa visualizzata solo dall'assistito. Nel momento del login quando l'utente assistito accede all'applicazione, viene passato come argomento il token dell'assistito. Infatti, la schermata della mappa dell'assistito ha come costruttore la stringa token corrispondente al token del dispositivo dell'assistito. Qui, la maggior parte delle caratteristiche è uguale alla schermata mappa del tutore. L'AppBar è uguale, il widget di GoogleMap lo stesso, cambiano solo i bottoni presenti nella parte bassa. Infatti, proprio perché si tratta dell'assistito ho deciso di lasciare solo il bottone di "SOS" per visualizzare i contatti di emergenza.

La particolarità di questa schermata è l'implementazione di una funzione asincrona _updateLocation() che una volta centrata la posizione dell'assistito attraverso un marker sulla mappa, tramite l'icona della geolocalizzazione, si prende queste coordinate del tipo LatLng (latitudine e longitudine) ed effettua una query a Firestore. Questa query preleva all'interno della collezione *utenti,* l'utente che ha il campo "token" uguale al valore del token passato come costruttore alla schermata. Cioè, si va a trovare il documento dell'utente assistito su Firebase. Fatto questo, sempre all'interno della funzione, viene fatta una seconda query di "update" in cui vengono aggiornati i valori dei campi "latitudine" e "longitudine" nel documento dell'assistito, con le coordinate effettive di quest'ultimo. Questa funzione viene ripetuta ogni 30 secondi per avere un tracciamento continuo della posizione dell'assistito. Queste coordinate, come già spiegato, vengono poi prelevate dal tutore nella schermata della mappa relativa al tutore.

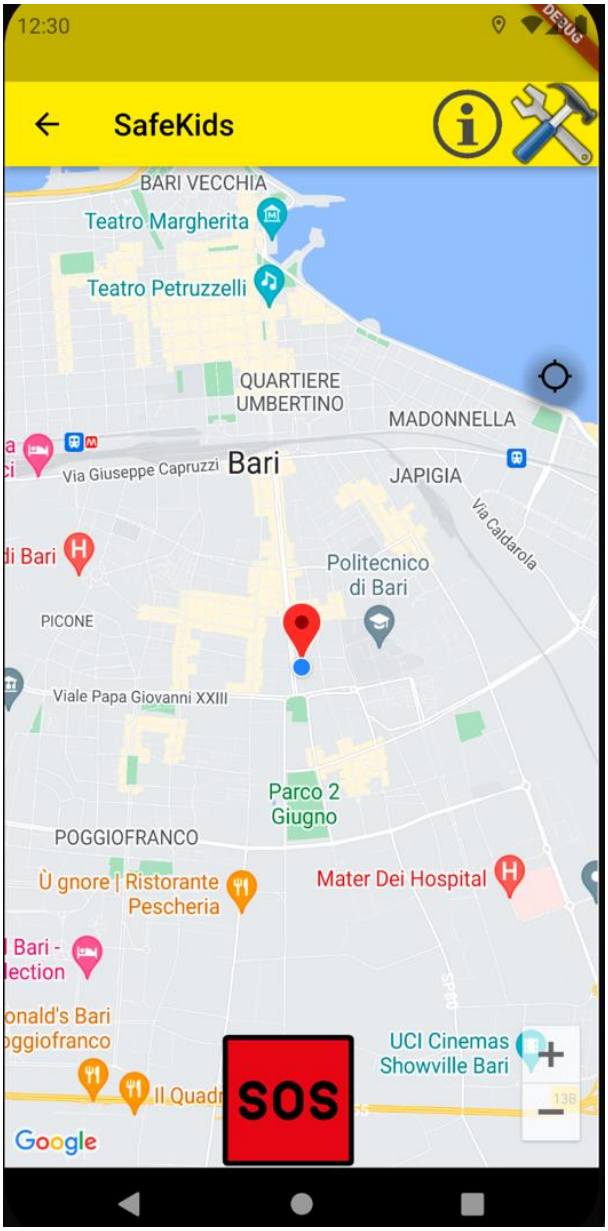


Figura 35 – Schermata mappa assistito

Capitolo 5 – Sperimentazione

Dopo aver spiegato le varie schermate della mia applicazione e come sono state implementato, ho effettuato una sperimentazione personale dell'applicazione. In particolare ho condotto esperimenti individuali sui singoli moduli con casi limite. Come fase inziale, ho testato l'applicazione attraverso l'emulatore android messo a disposizione da Android Studio, che permette di simulare qualsiasi tipo di applicazione Android in maniera eccellente. La sperimentazione, dapprima, è avvenuta con un emulatore e con un dispositivo fisico perché il secondo modulo, per l'associazione, prevede l'utilizzo di due dispositivi per poter scannerizzare il QR Code. In secondo luogo, ho effettuato la sperimentazione utilizzando due dispositivi fisici. Le sperimentazioni sono avvenute in ambiente reale e sono stati riscontrati risultati positivi su ogni modulo. Posso dire che l'obiettivo proposto dalla mia applicazione è stato raggiunto a pieno.

Nella seconda parte, ho deciso di svolgere la sperimentazione con utenti reali con lo scopo di valutare l'usabilità della mia applicazione e per valutare i vari moduli presenti.

5.1 Test con utenti reali

Il test è stato fatto con 10 utenti: 5 utenti sono di sesso femminile e gli altri 5 di sesso maschile, tra i 20 e i 60 anni di età. La maggior parte degli utenti possiede uno smartphone e sa come muoversi all'interno di esso. Ovviamente non tutti gli utenti sono esperti nell'utilizzo degli smartphone, ovvero non hanno tutti la stessa dimestichezza. Ho deciso di avere questa varietà di utenti per poter testare al meglio la mia applicazione e i moduli che la compongono.

Ogni utente ha lavorato in coppia perché l'applicazione richiede due tipologie di utenti (Tutore e Assistito). Sono state create 5 coppie di utenti. Ho chiesto agli utenti di effettuare quattro task che rappresentano le funzionalità più importanti dell'applicazione. Gli utenti hanno svolto autonomamente i vari task, sono stati liberi di commentare ad alta voce gli step adottati e tutte le cose che osservavano durante l'esecuzione. Dopo ogni task ho sottoposto loro tre domande:

1) "L'impressione generale sull'esperienza di completamento del task è positiva?";
2) "Ti è stato facile capire come completare il task?";
3) "Il tempo richiesto per completare il task è ragionevole?";

A ciascun utente, per poter rispondere a queste domande, è stata assegnata una scala Likert con 5 punti, dove 1 = "Per niente d'accordo", 2 = "Non d'accordo", 3 = "Neutrale", 4 = "D'accordo", 5 = "Pienamente d'accordo".

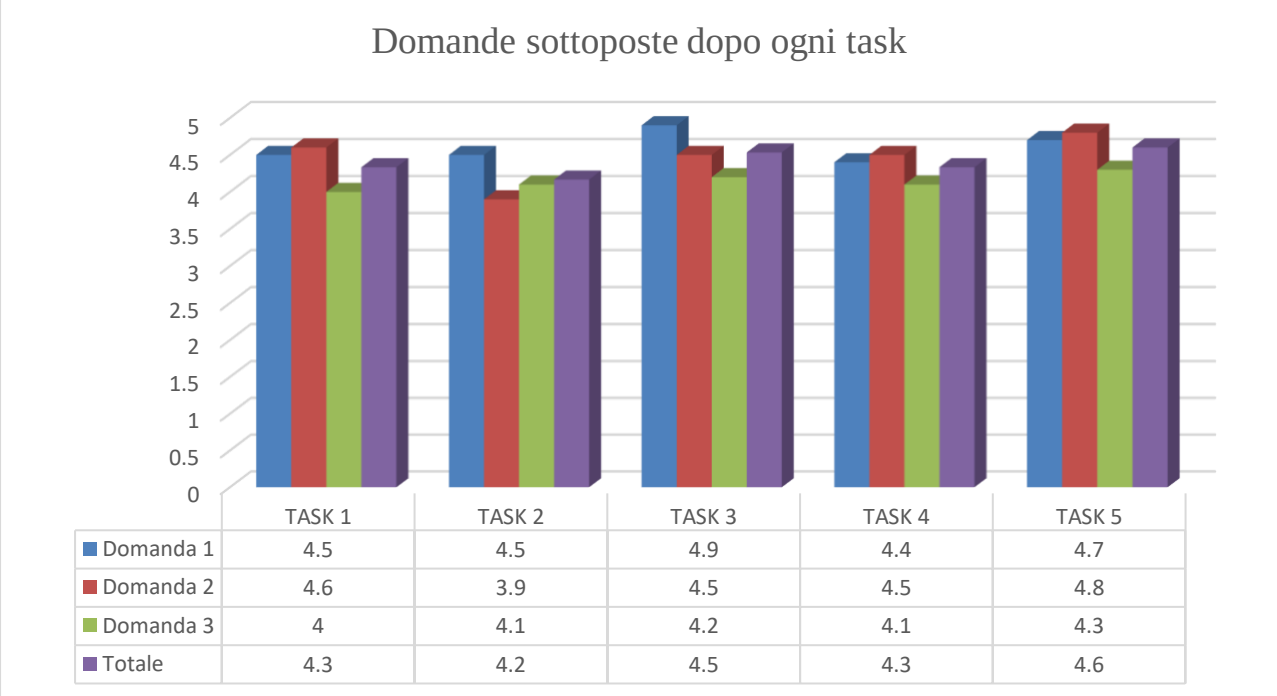
Alla fine di tutto il test, agli utenti è stato somministrato il questionario SUS e NPS per valutare il mio prodotto.

La scelta dei quattro task, da far svolgere agli utenti, è ricaduta sulle operazioni fondamentali che è possibile svolgere all'interno dell'applicazione. In particolare, i task sono:

1) "Creazione di un profilo utente (che può essere Tutore o Assistito)";
2) "Associazione fra due profili utente (quindi fra Tutore e Assistito)";
3) "Monitoraggio della posizione (il Tutore imposta una zona sicura e controlla la posizione dell'Assistito)";
4) "Creazione e salvataggio dei contatti di emergenza (accessibili dalle Impostazioni dell'app);
5) "Effettuare chiamate di emergenza e cancellazione dei contatti di emergenza (accessibili dal bottone di SOS);

L'applicazione richiede due tipologie di utenti (Tutore e Assistito), proprio per questo i task sono stati svolti in coppie, con la possibilità alla fine di ogni task di scambiarsi i ruoli affinché tutti gli utenti potessero testare le parti corrispondenti alle due tipologie di utenti.

Gli utenti sono stati liberi di effettuare i vari task autonomamente e soprattutto è stata data la possibilità di potersi esprimere ad alta voce su qualsiasi cosa presente all'interno delle varie schermate, magari soffermandosi sul design e su piccoli particolari presenti nell'applicazione. Il pensare ad alta voce, mi ha permesso anche di capire cosa manca all'interno della mia applicazione e soprattutto capire cosa bisogna implementare di nuovo.



	TASK 1	TASK 2	TASK 3	TASK 4	TASK 5
Domanda 1	4.5	4.5	4.9	4.4	4.7
Domanda 2	4.6	3.9	4.5	4.5	4.8
Domanda 3	4	4.1	4.2	4.1	4.3
Totale	4.3	4.2	4.5	4.3	4.6

Figura 36 - Punteggi delle domande sottoposte dopo ogni task

Ci sono diverse osservazioni da fare sui risultati ottenuti. Per prima cosa, posso dire che i vari task hanno avuto successo, ma alcuni hanno avuto più o meno successo rispetto agli altri. Partendo dai task più apprezzati, i due che hanno avuto maggior successo sono il task numero 3 ("Monitoraggio della posizione (il Tutore imposta una zona sicura e controlla la posizione dell'Assistito)" e il task numero 5 ("Effettuare chiamate di emergenza e cancellazione dei contatti di emergenza (accessibili dal bottone di SOS)). In particolare, il task numero 5 è il task più apprezzato rispetto a tutti gli altri. I task che hanno avuto un punteggio uguale sono il task numero 1 ("Creazione di un profilo utente (che può essere Tutore o Assistito) e il numero 4 ("Creazione e salvataggio dei contatti di emergenza (accessibili dalle Impostazioni dell'app)). Mentre il task numero 2 ("Associazione fra due profili utente (quindi fra Tutore e Assistito)) ha avuto il punteggio più basso rispetto agli altri. Tutti i risultati ottenuti rispecchiano a pieno i commenti e le osservazioni poste dagli utenti mentre svolgevano i task. Partendo dal task numero 2, il più basso, i commenti e le osservazioni che venivano poste dagli utenti riguardavano principalmente la schermata dedicata all'utente Assistito e in particolare al messaggio di testo che compare sopra il bottone "Genera QR Code" (ovvero, "Se non sei ancora associato, premendo il bottone, ti comparirà il QR Code. In caso contrario si aprirà la mappa!") che in alcuni utenti ha creato confusione, in genere negli utenti più anziani. Infatti, grazie a questo sono riuscito a capire cosa dovrebbe essere cambiato, bisognerebbe creare due schermate separate. Ovvero se l'utente si deve ancora associare bisogna far aprire la schermata per generare il QR Code, altrimenti se già associato far aprire direttamente la mappa. Mentre, per quanto riguarda il task numero 1 le impressioni da parte di tutti gli utenti sono state positive, è stata apprezzata l'interfaccia molto intuitiva e semplice e soprattutto la facilità nel completare il task. Ho notato che tutti gli utenti hanno espresso commenti e osservazioni simili nella schermata della Registrazione, in particolare sul campo Ruolo, dove si può inserire solo la stringa "genitore" oppure la stringa "figlio". I dubbi, simili in tutti gli utenti, sono nati dal fatto che se non viene inserita bene la stringa "genitore" o "figlio" l'applicazione non procede alla registrazione dell'utente proprio perché le stringhe devono essere inserite correttamente. Gli utenti, però hanno risolto subito l'inconveniente e sono andati avanti. L'altro task, che ha avuto il punteggio uguale al task numero 1, è il numero 4. Anche in questo task i commenti tutto sommato sono stati positivi, gli utenti hanno apprezzato molto l'icona delle Impostazioni molto intuitiva. All'interno si accede alla schermata per creare e salvare i contatti, oltre ai commenti positivi sull'interfaccia moderna, gli utenti hanno notato che alcune parti dell'interfaccia si leggono poco a causa del colore giallo che rende alcuni tratti poco leggibili. Il task numero 3, il secondo per punteggio più alto, è stato apprezzato molto. Gli utenti hanno posto osservazioni su come viene gestita la mappa e tutte le altre componenti nella schermata in maniera positiva. Gli utenti hanno usato con facilità la mappa, hanno capito fin da subito in modo del tutto autonomo che bisogna impostare una zona sicura prima di continuare con il rilevamento della posizione. Infatti, la creazione della zona sicura è stata valutata egregiamente per la facilità e per la modernità dell'operazione. Gli utenti non si aspettavano che sulla mappa comparisse effettivamente una zona sicura visibile di colore blu chiaro e nel momento in cui rilevavano la posizione dell'assistito, sono rimasti colpiti su come è avvenuta tutto il processo di geolocalizzazione. Alcuni utenti sono rimasti colpiti dal fatto che una funzionalità del genere è di grande aiuto per i genitori e per la sicurezza dei loro figli e dei loro assistiti. Infine, il task con il punteggio più alto, ovvero il numero 5 è stato apprezzato molto. Gli utenti sono rimasti colpiti, anche in questo caso, dall'intuitività dell'operazione perché sull'interfaccia è visibile un bottone di medie dimensione con scritto "SOS" di colore rosso che ha fatto pensare fin da subito che servisse per effettuare chiamate di emergenza e quindi chiedere aiuto.

Dopo aver terminato i vari task, agli utenti è stato somministrato il questionario SUS e NPS per valutare il mio prodotto.

5.2 Questionario SUS

Agli utenti, dopo aver terminato i vari task, è stato sottoposto il questionario SUS. Quest'ultimo è il più utilizzato nel campo dell'usabilità. È molto usato per valutare l'usabilità di siti web e applicazioni come nel mio caso. È costituito da 10 affermazioni con scala Likert a 5 punti e sono disposte in modo ordinato e specifico e comprendono la facilità di apprendimento del sistema, l'efficienza, la capacità di memorizzare e di recuperare le informazioni e cosi via. Le affermazioni, all'interno del questionario, che sono indicate da un numero dispari (ovvero 1,3,5,7,9) sono formulate in modo positivo, le affermazioni che sono indicate, invece, da un numero pari (ovvero 2,4,6,8,10) sono formulate in modo negativo come se rappresentassero l'esatto opposto. Il motivo di questo è la compilazione veritiera del questionario, l'utente dovrà quindi rispondere con attenzione e leggere bene le affermazioni affinché possa rispondere a tutte le domande in maniera corretta e veritiera.

Per calcolare i punteggi delle risposte, si usano 3 criteri importanti:

1) Per gli item dispari, dopo aver sommato le risposte bisogna sottrarre 5;
2) Per gli item pari, dopo aver sommato le risposte bisogna sottrarle a 25;
3) Si sommano i risultati dei punti 1 e 2 e lo si moltiplica per 2,5 ottenendo un risultato/punteggio espresso in centesimi.

Un qualsiasi sito o applicazione ha uno score di usabilità buono se il risultato finale è maggiore di 68. Il risultato può variare da 0 a 100. La Figura 37 mostra i risultati ottenuti.

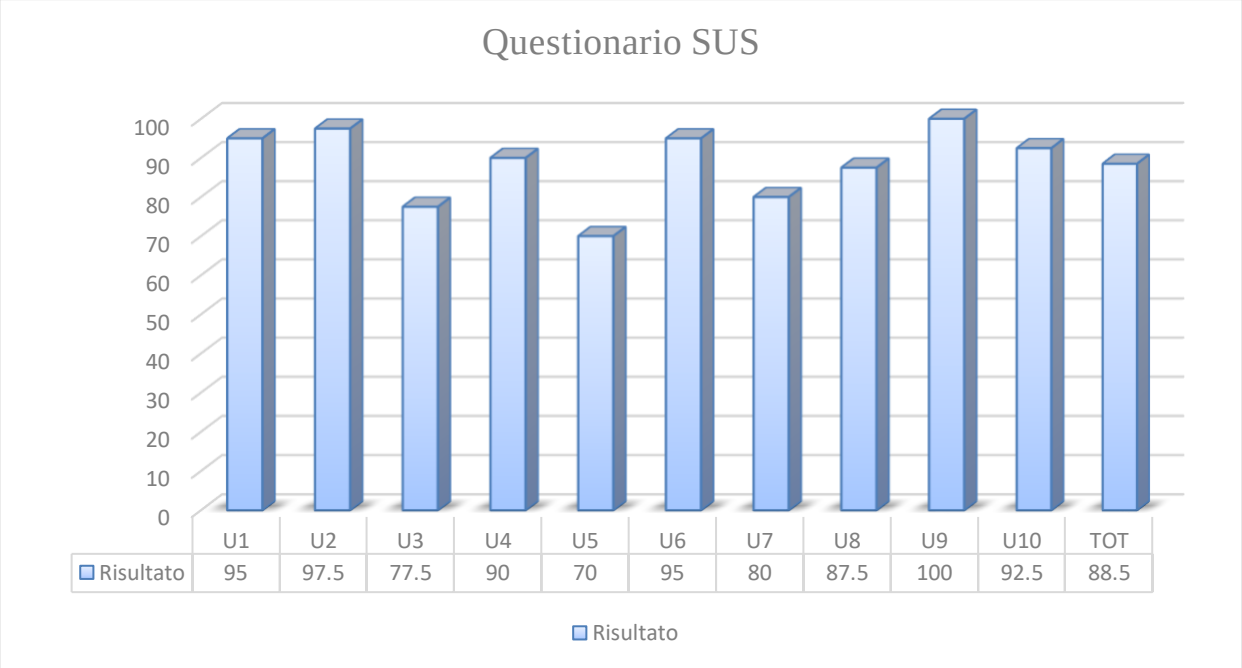


Figura 37 - Questionario SUS

Come si può vedere dal grafico, complessivamente la mia applicazione è stata apprezzata molto dagli utenti che hanno svolto i vari task. Dai risultati si deduce che ciascun modulo di SafeKids offre funzionalità che vengono apprezzate e soprattutto offre funzionalità che possono servire a famiglie, tutori ecc. Solo con un utente si raggiunge il valore 100, mentre tutti gli altri oscillano fra 80 e 97. Come si può notare solo un utente ha generato un risultato pari a 70 che è il più basso del questionario. Il questionario SUS raggiunge una media totale dei punteggi pari a **88,5**; significa che la mia applicazione, dal punto di vista dell'usabilità, è a un buon se non ottimo livello. I punteggi più bassi, ovvero quelli al di sotto di 80, probabilmente sono dovuti al fatto che alcuni utenti hanno poca dimestichezza con l'uso degli smartphone e delle varie applicazioni.

5.3 Questionario NPS

Il questionario NPS è un parametro basato su una singola domanda: "***In una scala da 0 a 10, quanto probabilmente raccomanderesti SafeKids ad un amico o collega?***". I partecipanti rispondono in base a una scala da 0 a 10, dove 0 significa "molto improbabile" e 10 significa "molto probabile". In base al punteggio, i partecipanti vengono suddivisi in tre categorie:

- Promotori – Punteggio da 9 a 10;
- Passivi – Punteggio da 7 a 8;
- Detrattori – Punteggio da 0 a 6;

Il valore dell'NPS viene calcolato facendo la differenza fra il numero di promotori e il numero di detrattori, il risultato diviso il numero di utenti. Il risultato complessivo moltiplicato per 100. L'NPS è un modo efficace per valutare la soddisfazione degli utenti che devono valutare un prodotto, in questo caso la mia applicazione. La Figura 38 mostra i risultati ottenuti.

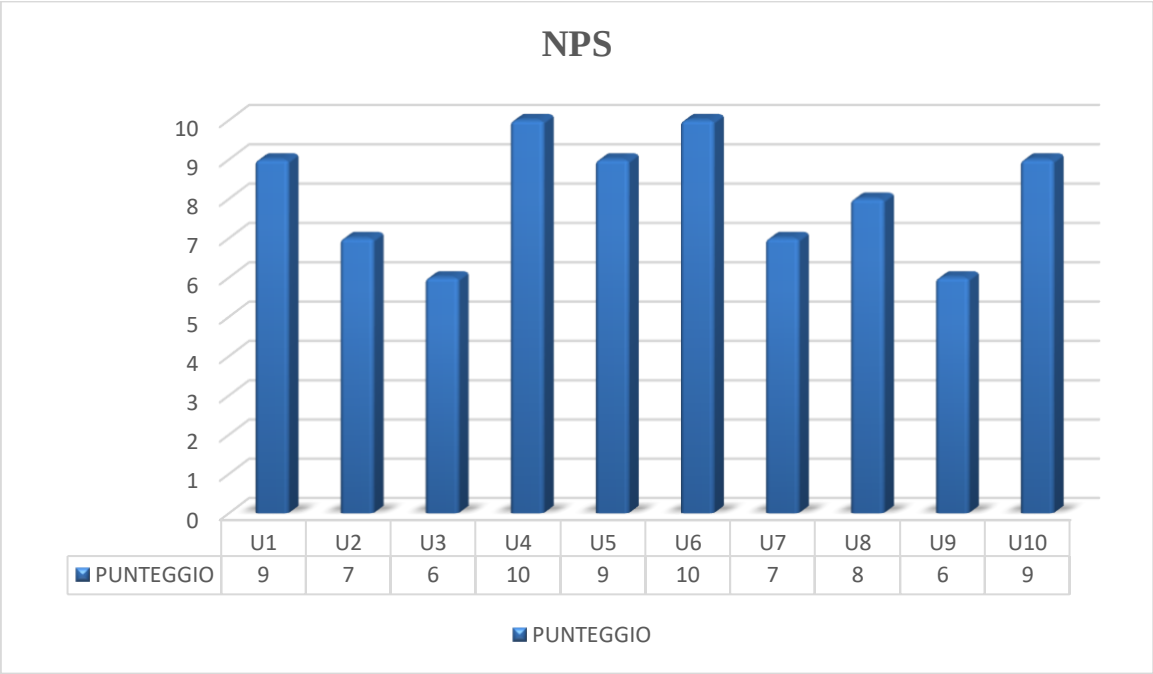


Figura 38 – Risultati NPS

Il punteggio dell'NPS è il seguente, mostrato in Figura 39.

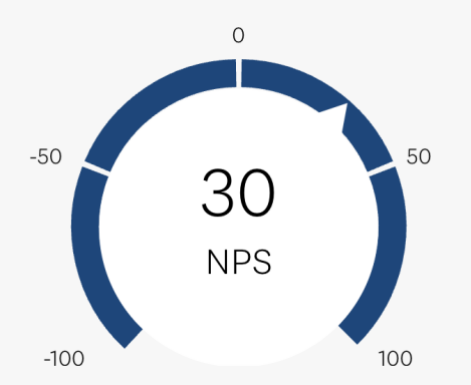


Figura 39 - Punteggio NPS

Il questionario NPS ci mostra che sono presenti 5 promotori, 3 passivi e 2 detrattori. Ci fa capire che l'applicazione è nella media. La maggioranza degli utenti è promotrice.

5.4 Risultati finali

Per concludere, i due questionario hanno dimostrato che nel complesso la mia applicazione, SafeKids, è un prodotto valido, efficiente e apprezzato.

Si tratta di un'applicazione nella media, ci sono alcuni aspetti che vanno migliorati. Infatti, effettuare test con utenti reali mi ha dato la possibilità di capire cosa modificare per rendere il tutto più efficiente. Sono riuscito a capire quali sono i problemi principali della mia applicazione che riguardano l'interfaccia utente nel quarto modulo e una funzionalità implementata nel secondo modulo.

Gli utenti sono rimasti colpiti dalla struttura dell'applicazione, per la sua modernità ma soprattutto per quello che offre. Sono stati molti i commenti per quanto riguarda l'utilità della mia applicazione. È un'applicazione rivolta ad aiutare il prossimo.

I vari utenti hanno elogiato anche l'originalità della mia applicazione e l'innovazione.

Capitolo 6 – Conclusioni

Con tutto il lavoro svolto sono riuscito a sviluppare un'applicazione Android basata sulla tecnologia Flutter con Dart per il Parental Control. Il mio obiettivo è stato fin da subito fornire un supporto tecnologico ed efficace per garantire la sicurezza delle persone affette da disturbi cognitivi e dei loro tutori, per aiutare i genitori e alleviare le loro paure nei confronti dei loro bambini.

Attraverso l'analisi di diverse soluzioni presenti in letteratura, ho individuato le principali caratteristiche e funzionalità che un'applicazione per la gestione e il monitoraggio della posizione dovrebbe avere. Sulla base di queste informazioni, ho sviluppato un'applicazione che consente di monitorare la posizione degli assistiti e di inviare notifiche di allarme in caso di vagabondaggio o di emergenza.

Durante lo sviluppo della mia applicazione, ho utilizzato la tecnologia di Flutter con Dart per garantire un'interfaccia utente intuitiva e reattiva, e ho integrato diverse funzionalità avanzate, come la geolocalizzazione e il rilevamento del wandering.

Effettuando i test con i vari utenti è emerso che la mia applicazione ha alcuni errori che devono essere risolti, ma allo stesso tempo è stata evidenziata la modernità e l'efficienza della mia applicazione e di questo ne sono fiero.

Grazie ai questionari SUS e NPS, somministrati agli utenti, ho capito che la mia applicazione è molto intuitiva e facile da usare, i commenti degli utenti durante l'utilizzo sono stati positivi sia per l'interfaccia utente che per le funzionalità. Ovviamente ci sono stati anche commenti riguardanti alcuni problemi emersi a livello di interfaccia e di funzionalità, che mi hanno aiutato a capire quali soluzioni adottare.

Grazie a questo lavoro, mi sono immedesimato molto nelle situazioni che vivono quotidianamente assistiti e tutori.

La voglia di poter gestire e monitorare la posizione è nata anche da un'esperienza che ho vissuto personalmente con il mio caro nonno affetto da Alzheimer.

Capitolo 7 – Sviluppi Futuri

La mia applicazione non è completa a livello di tutte le possibili funzionalità che possono essere inserite.

In particolare analizzando i vari articoli scientifici del Capitolo 2, sono riuscito a capire quali altre funzionalità o sistemi inserire e utilizzare per rendere ancora più efficiente la mia applicazione.

Riguardo alla mia applicazione per il Parental Control, possono essere aggiunti i seguenti moduli:

- Partendo dalla creazione della zona sicura, si potrebbe aggiungere una funzionalità che permette al tutore non solo di aggiungere una zona sicura sulla propria posizione, ma di disegnare altre zone sicure in altre zone della mappa;

- Una funzionalità che permette al tutore, premendo un apposito bottone, di ascoltare cosa succede nell'ambiente circostante dell'assistito;

- Una funzionalità che permette all'assistito di inviare messaggi di SOS sul telefono del tutore;

- Una rete bayesiana o una rete neurale addestrata a conoscere le tracce degli spostamenti dell'assistito e rilevare se si è effettivamente perso oppure no.

Bibliografia

[1] D. L. Algase, D. H. Moore, C. Vandeweerd, and D. J. Gavin-Dreschnack, “Mapping the maze of terms and definitions in dementia-related wandering,” *Aging Ment Health*, vol. 11, no. 6, pp. 686–698, 2007, doi: 10.1080/13607860701366434.

[2] L. D. Wiggins *et al.*, “Wandering Among Preschool Children with and Without Autism Spectrum Disorder,” *J Dev Behav Pediatr*, vol. 41, no. 4, pp. 251–257, 2020, doi: 10.1097/DBP.0000000000000780.

[3] W. Sansrimahachai, “Stream-based wandering monitoring system for elderly people with dementia,” in *2015 15th International Symposium on Communications and Information Technologies, ISCIT 2015*, Institute of Electrical and Electronics Engineers Inc., Apr. 2016, pp. 1–4. doi: 10.1109/ISCIT.2015.7458292.

[4] D. Martino-Saltzman, B. B. Blasch, R. D. Morris, and L. W. McNeal, “Travel behavior of nursing home residents perceived as wanderers and nonwanderers,” *Gerontologist*, vol. 31, no. 5, 1991, doi: 10.1093/geront/31.5.666.

[5] E. R. Pratama, F. Renaldi, F. R. Umbara, and E. C. Djamal, “Geofencing Technology in Monitoring of Geriatric Patients Suffering from Dementia and Alzheimer,” in *2020 3rd International Conference on Computer and Informatics Engineering, IC2IE 2020*, Institute of Electrical and Electronics Engineers Inc., Sep. 2020, pp. 106–111. doi: 10.1109/IC2IE50715.2020.9274637.

[6] M. Cs. Arfiani, Ika , S.T., “Application of the Haversine Formula Method in Geographical Information Systems for Land Area Measurement,” *Jurnal Sistem dan Teknologi Informasi (JUSTIN)*, vol. 10, no. 2, pp. 1262–1270, 2016.

[7] applicazioni.it, “Find My Kids, il tracker GPS per famiglie,” May 24, 2019. https://www.applicazioni.it/find-my-kids-il-tracker-gps-per-famiglie/#Find_My_Kids_cos8217e_e_come_funziona

[8] F. Sposaro, J. Danielson, and G. Tyson, “IWander: An Android application for dementia patients,” in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC’10*, 2010, pp. 3875–3878. doi: 10.1109/IEMBS.2010.5627669.

[9] B. S. Beauvais, V. Rialle, and J. Sablier, “MyVigi: An Android application to detect fall and wandering,” in *UBICOMM 2012 - 6th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2012, pp. 156–160.

[10] C. Khawas and P. Shah, “Application of Firebase in Android App Development-A Study,” *Int J Comput Appl*, vol. 179, no. 46, pp. 49–53, Jun. 2018, doi: 10.5120/ijca2018917200.