Appericolo

Gioele Pasquini e Sara Montese

22 giugno 2022

Progetto di Programmazione Mobile

Anno accademico 2021/2022

Sotto la guida del

Prof. Emanuele Storti

1 Introduzione

Appericolo è un'app di geolocalizzazione, dedicata in particolare al sesso femminile, che mira a porre fine alla sensazione di insicurezza che troppo spesso si manifesta quando le donne sono da sole in svariate situazioni, dalla passeggiata con il cane, al jogging, al semplice tornare a casa a piedi la sera. Chiunque ha il diritto di muoversi in totale sicurezza e libertà, ovunque e in qualsiasi momento, senza dover provare paura o preoccupazione. Per questa ragione, ci siamo impegnati a creare un'applicazione che non solo cerchi di ridurre la sensazione di insicurezza, ma prevenga anche possibili minacce e sia di aiuto in caso di emergenza.
Nello specifico, Appericolo è di un'app di tracciamento che consente di monitorare il tragitto di un utente verso la destinazione scelta. Tale app è utile non solo in caso di pericolo effettivo, ma anche per tranquillizzare un parente, un amico o il proprio partner quando si viaggia o si gira per strada di notte.

2 Sviluppo Android

2.1 Analisi dei requisiti

2.1.1 Requisiti funzionali e non funzionali

In questa sezione, vengono illustrati i requisiti funzionali, ovvero l'insieme delle funzionalità che dovranno essere implementate nell'applicazione. Essi sono elencati qui di seguito:

- Registrazione di un nuovo account;
- Login e logout dall'applicazione;
- Possibilità di pianificare una chiamata finta ad una determinata ora; questa funzione può essere utilizzata dall'utente per sottrarsi a situazioni scomode;
- Tracciamento e condivisione della posizione: sfruttando il GPS e permettendo di geolocalizzare l'utilizzatore, l'applicazione consente di condividere la propria posizione con i contatti preimpostati;
- Gestione della lista dei contatti più stretti, a cui l'utente condivide la propria posizione;
- Gestione della lista delle destinazioni più frequenti dell'utente, in modo da facilitare la configurazione di un nuovo tracciamento della posizione.

I requisiti non funzionali, invece, definiscono l'insieme dei vincoli e delle regole imposte sull'app, sia di tipo realizzativo che tecnico. Questi vengono riportati qui di seguito:

- l'app dovrà avere un design moderno, user-friendly e dalle tonalità scure per non dare nell'occhio;
- l'app dovrà avere delle buone performance ed essere reattiva;
- l'app dovrà essere il più affidabile possibile, evitando crash improvvisi;
- l'app dovrà chiedere all'utente il consenso per accedere alla sua posizione e ai suoi contatti;
- i dati relativi ai contatti più stretti e alle destinazioni più frequenti vengono salvati su un apposito database locale, mentre i dati personali di ogni utente, verranno salvati su un database remoto su Firebase. In particolare, trattandosi di un'app di tracking della posizione, si rende necessario l'utilizzo di un database realtime.

2.1.2 Attori e Casi d'uso

Dopo aver analizzato i requisiti funzionali imposti all'applicazione, si identificano i potenziali attori. Un attore raffigura il ruolo che un utente esterno può ricoprire nel momento in cui si approccia all'applicativo. Distinguiamo due diverse tipologie di attore:

- colui che utilizza l'app per condividere la propria posizione con i contatti più stretti, nel tragitto da un punto ad un altro;
- colui che utilizza l'app per seguire in tempo reale il percorso tracciato da un altro utente.

Uno stesso utente può utilizzare l'app per entrambi i ruoli. Per poter usufruire delle funzionalità dell'app, entrambe le tipologie di attori devono possedere un account ed essere autenticati.
A partire dai requisiti funzionali, ogni attore può compiere determinate azioni chiamate "casi d'uso"; queste vengono riassunte nel diagramma illustrato di seguito (Figura 23).

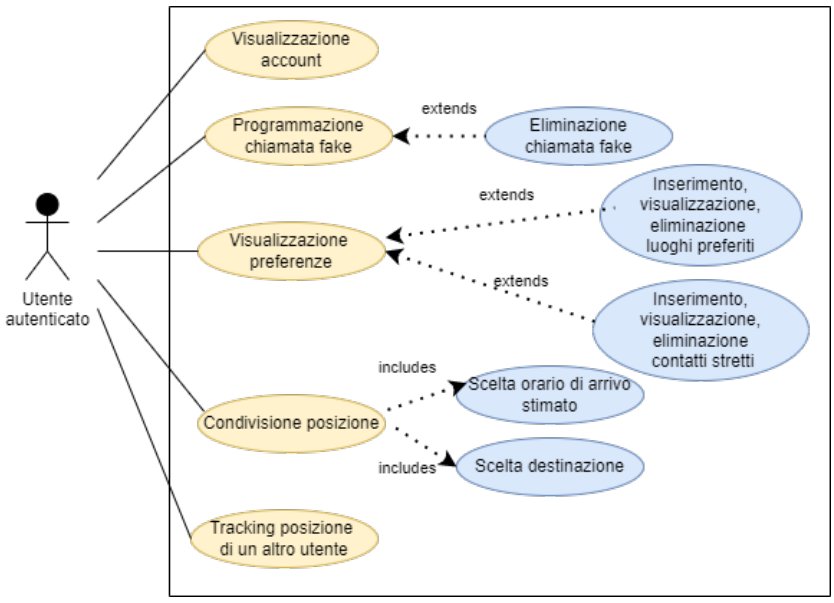

Figura 1: Diagramma dei casi d'uso relativi all'utente con account

2.2 Progettazione dell'applicazione

2.2.1 Architettura

La progettazione dell'app si è basata sull'architettura MVVM (Model View View-Model) proposta tra le best practice di Android. A partire da questa, l'app è stata strutturata su 4 livelli:

1. *View*, contenente un'Activity per ciascun requisito, Fragment, Adapter e altri classi collegate alla fruizione dell'app;
2. *ViewModel*, ovvero lo strato destinato a fornire dati all'interfaccia utente e a ricevere da questa direttive sulle operazioni da attuare.
3. *Repository*, da cui il ViewModel attinge le informazioni per l'interfaccia utente, indipendentemente dal fatto che esse provengano da un database locale o remoto.
4. *Model*, ovvero il modo in cui si rappresentano i dati. Esso di basa sui dati contenuti su Firebase e sul database locale (Sezione 2.2.3).

Nel progetto il pattern è stato seguito dove necessario, per evitare di aumentare la complessità del codice.

2.2.2 Mockup

In questa sezione, si analizzano i mockup delle interfacce dell'app, ovvero la rappresentazione grafica stilizzata delle schermate che costituiscono l'applicazione da sviluppare.

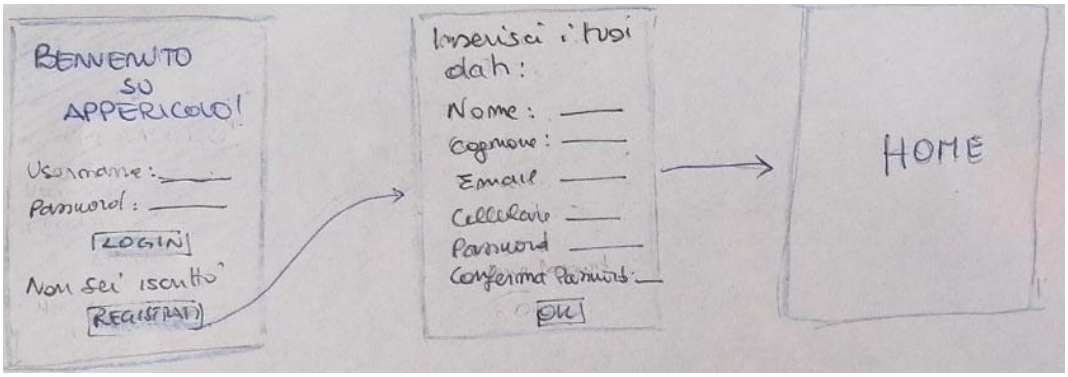

Figura 2: Registrazione

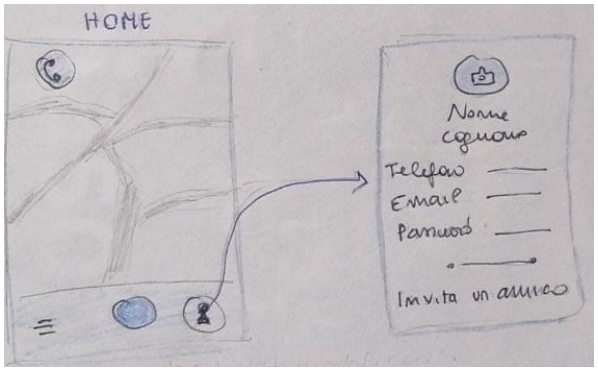

Figura 3: Registrazione

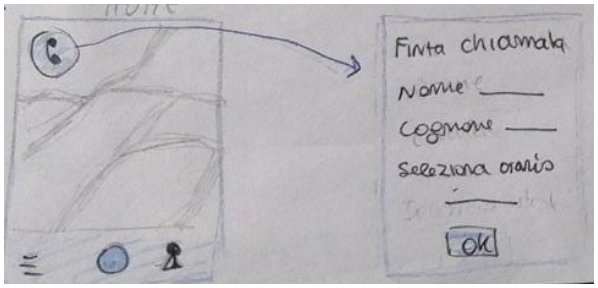

Figura 4: Definizione di una chiamata fake

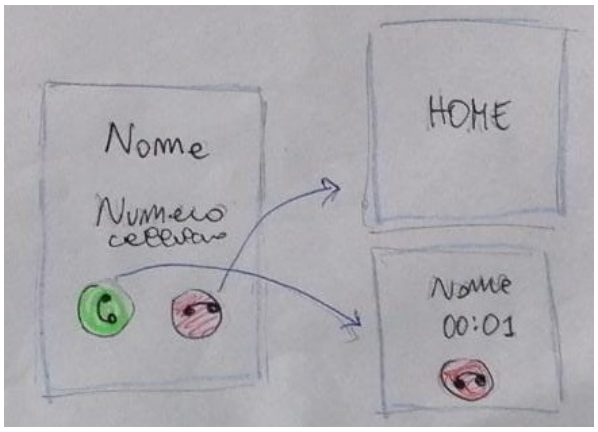

Figura 5: Arrivo di una chiamata fake

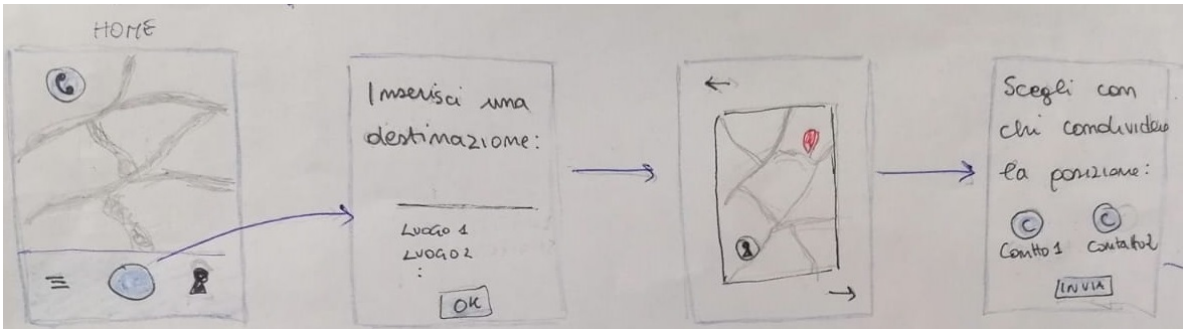

Figura 6: Scelta della destinazione e dei contatti fidati

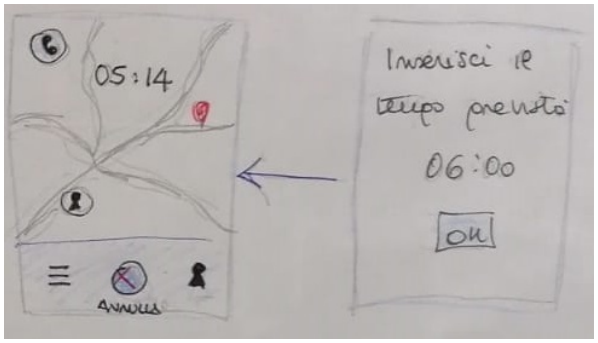

Figura 7: Scelta del tempo di arrivo stimato alla destinazione

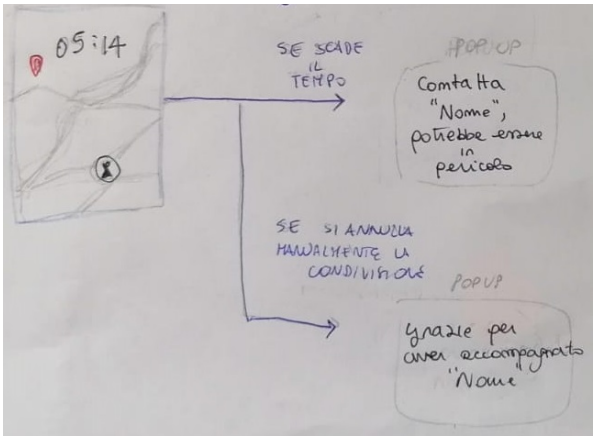

Figura 8: Tracciamento della posizione da parte dei contatti stretti

2.2.3 Gestione dei dati

Firebase
Per la gestione dei dati degli utenti si è deciso di utilizzare Firebase, un servizio di backend realizzato da Google, che offre servizi di database, di autenticazione, di hosting web, oltre a molte altre funzioni. Tale scelta è dovuta al fatto che Firebase offre alcune funzionalità che rientrano nei requisiti dell'applicazione, come la gestione dell'autenticazione, ma in particolare per la possibilità di utilizzare un database Realtime, utile per il tracking in tempo reale della posizione di un utente.
In particolare, si è usufruito del servizio *Authentication* per l'autenticazione degli utenti nell'app, basata su email e password. Per ciascun utente registrato, viene creata un'instanza sul database Realtime di Firebase. Ciò è utile non solo per lo storage dei dati tipici dell'utente (nome, cognome, cellulare) e della lista dei suoi contatti stretti, ma soprattutto per tenere traccia in tempo reale dell'ultima posizione dell'utente e del suo registration token. Il token di registrazione è utile, nell'ambito del *FirebaseMessagingService*, per la ricezione di messaggi e notifiche sull'app; tale servizio viene sfruttato in particolare quando un utente vuole notificare ai contatti più stretti i propri spostamenti.
La scelta di utilizzare un Firebase Realtime Database ha comportato, tuttavia delle difficoltà: trattandosi di un database non relazionale, tale scelta ha aumentato la complessità di effettuare query sui dati.

Room
Per la memorizzazione delle preferenze dell'utente, si è fatto uso della libreria Room, che, fornendo un livello di astrazione su SQLite, consente di sfruttare tutta l'espressività del linguaggio di query. Tale libreria, in unione con ViewModel e LiveData, è stata utilizzata per:

- l'inserimento, l'eliminazione e la visualizzazione delle destinazioni più frequenti dell'utente, delle quali sono noti *indirizzo*, *latitudine* e *longitudine*;
- l'inserimento dalla rubrica, l'eliminazione e la visualizzazione dei contatti fidati dell'utente, dei quali sono noti *nome* e *cellulare*. La lista dei contatti stretti viene memorizzata anche sul database in remoto, in modo che l'utente eviti di definirli nuovamente ogni qual volta effettua il login.

2.3 Implementazione dell'applicazione

All'avvio dell'applicazione (Figura 9), l'utente si trova di fronte una schermata in cui può inserire le credenziali per effettuare il login. Nel caso in cui l'utente si approcci per la prima volta all'app, in fondo alla schermata è presente un link che permette la registrazione di un nuovo account. Cliccando sul link, viene avviata la schermata per la registrazione (Figura 10) che consente l'inserimento dei dati necessari.

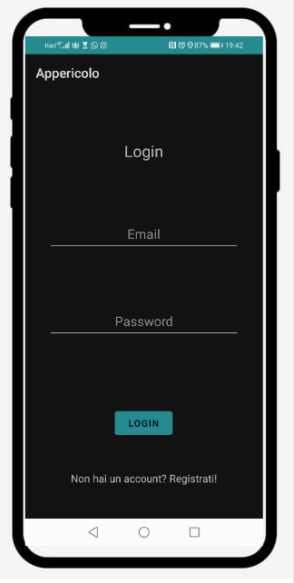

Figura 9: Login

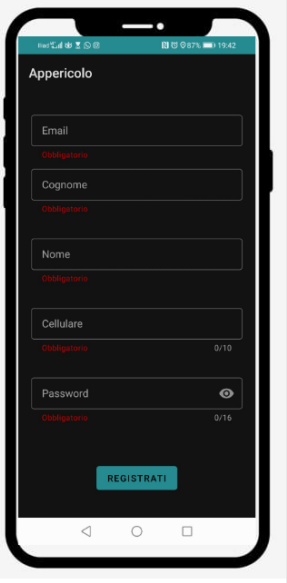

Figura 10: Registrazione

Dopo aver effettuato il login con successo, l'utente visualizza l'Homepage dell'app (Figura 11). In questa prima fase, all'utente si richiedono i permessi necessari per consentire all'app di accedere alla sua posizione. Una volta concessi i permessi, l'utente può visualizzare la propria posizione attuale.

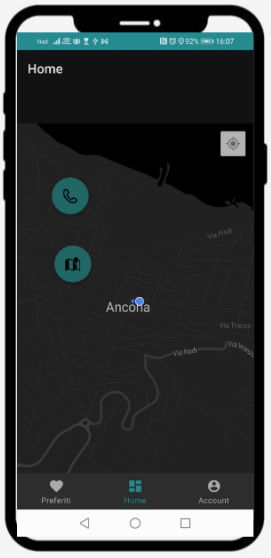

Figura 11: Home

Cliccando sulla relativa icona presente nella Home, è possibile programmare una chiamata finta ad un orario ben preciso. Si viene direzionati in una nuova activity che consente l'inserimento dei dati fittizi della chiamata, quali il mittente, il numero di cellulare e l'orario a cui si intende riceverla (Figura

12). Inoltre, da questa schermata è possibile anche eliminare la chiamata programmata, attraverso l'apposita icona del calendario sulla Action Bar.

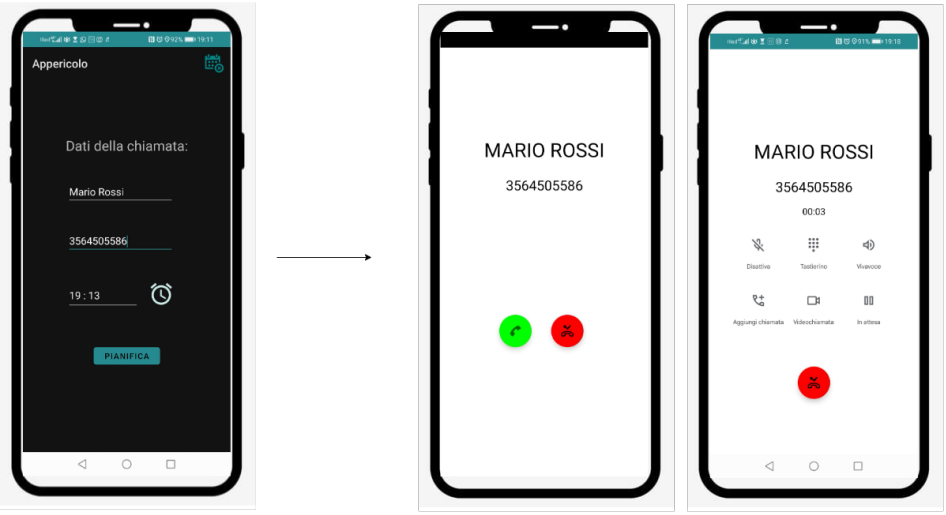

Figura 12: Creazione e ricezione della chiamata fake

Quando giunge l'ora specificata, possono verificarsi due casi:

- il device è sbloccato: esso squilla e si riceve una notifica per la chiamata in entrata; cliccando sulla notifica, è possibile accettare o rifiutare;
- il device è bloccato: esso squilla e lo schermo si illumina, mostrando la chiamata in entrata e la relativa notifica. Anche in questo caso è possibile accettare o rifiutare.

Nel testing dell'app, si è notato che la chiamata finta giunge al device con circa 2 minuti di ritardo rispetto all'orario stabilito. Cliccando sull'icona della mappa presente nella Home, si avvia la funzionalità centrale dell'app: la condivisione in tempo reale della posizione dell'utente con i contatti fidati (Figura 13).

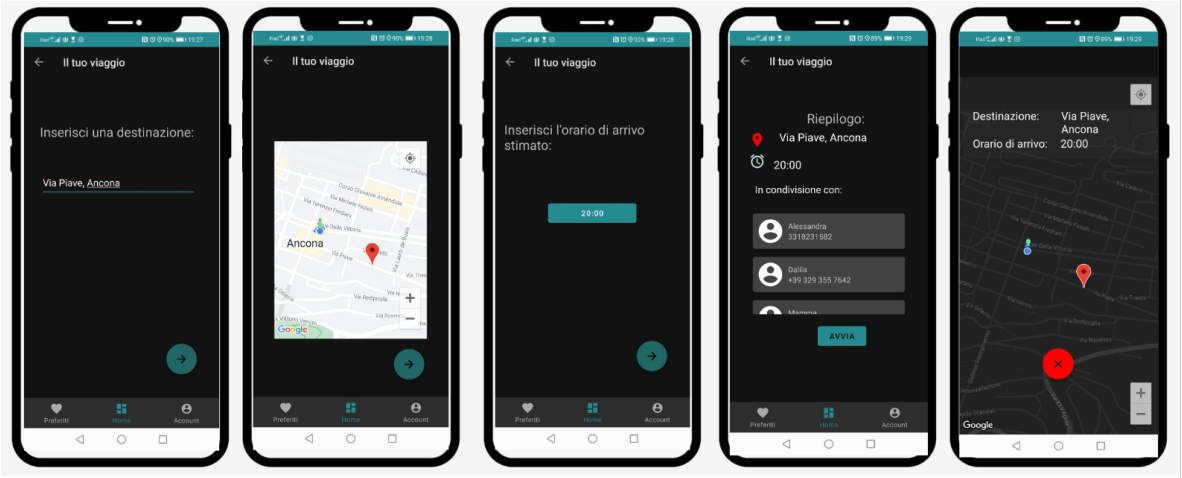

Figura 13: Condivisione della posizione in tempo reale (Client)

Dopo il click, grazie ad un *navigation graph*, l'utente naviga attraverso una serie di fragment per definire le specifiche della condivisione. In particolare, l'utente deve scegliere e confermare la destinazione da raggiungere; nell'inserimento dell'indirizzo è possibile sfruttare l'autocompletamento della TextView in base ai luoghi definiti nelle preferenze. Successivamente, l'utente seleziona l'orario di arrivo stimato e, solo dopo aver confermato i dati inseriti, può avviare la condivisione.

Dopo aver premuto sul tasto *Avvia*, i contatti fidati dell'utente ricevono una notifica di inizio condivisione; cliccando sulla notifica essi potranno monitorare gli spostamenti dell'utente in tempo reale e visualizzare l'orario di arrivo stimato (Figura 14).
Inizialmente, tale procedimento prevedeva di navigare una serie di Activity attraverso degli intent. Tuttavia, tale approccio ha creato problemi nel momento in cui l'utente tentava di tornare indietro nelle Activity precedenti per modificare i dati specificati. Tale problematica è stata risolta sostituendo le Activity con dei Fragment e facendo uso, come anticipato, di un *navigation graph*.

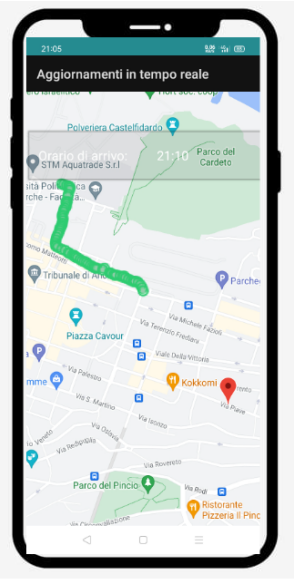

Figura 14: Tracciamento in tempo reale della posizione (Receiver)

I contatti stretti dell'utente riceveranno una ulteriore notifica se:

a) l'utente interrompe la condivisione, premendo l'apposito pulsante rosso;

b) l'orario di arrivo stimato è passato e la posizione dell'utente non si trova nel raggio di 150 metri dalla destinazione.

Durante il testing, si è notato che l'app presenta comportamenti anomali nel caso in cui un contatto fidato lasci accumulare le notifiche di condivisione senza aprirle. Il passaggio da una schermata all'altra avviene attraverso una bottom navigation bar. La sezione a destra della home è dedicata alle informazioni dell'account personale dell'utente (Figura 15). Qui vengono mostrati i dati inseriti al momento della registrazione e si ha la possibilità di uscire dall'applicazione.
La sezione a sinistra della Home è dedicata alle preferenze dell'utente (Figura 16). Si tratta di un *tabbed layout* con:

- una sezione per l'inserimento, la visualizzazione e l'eliminazione dei contatti fidati dell'utente, provenienti direttamente dalla rubrica (dopo aver concesso i relativi permessi). Nell'implementazione si è supposto che i contatti scelti dall'utente possiedano un account di Appericolo (in caso contrario, essi non vengono notificati al momento della condivisione);

- una sezione per l'inserimento, la visualizzazione e l'eliminazione delle destinazioni tipiche dell'utente, in modo da facilitare successivamente la scelta dell'indirizzo da raggiungere. Tale lista viene svuotata ogni volta che l'utente effettua il logout dall'applicazione.

Cliccando sul singolo elemento della RecyclerView, si ha la possibilità di eliminare il dato dalla lista.

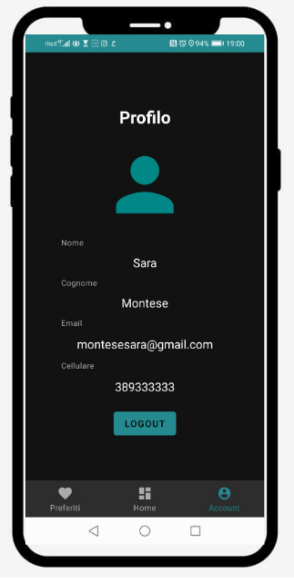

Figura 15: Account

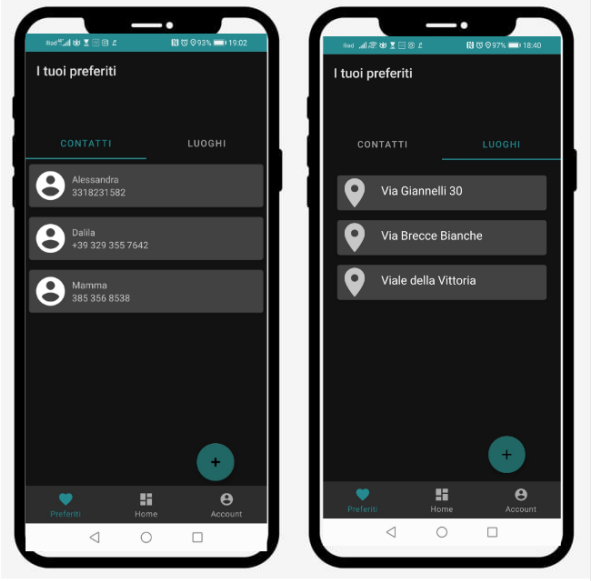

Figura 16: Contatti stretti e destinazioni frequenti

Di seguito si mostra una porzione del Firebase Realtime Database:

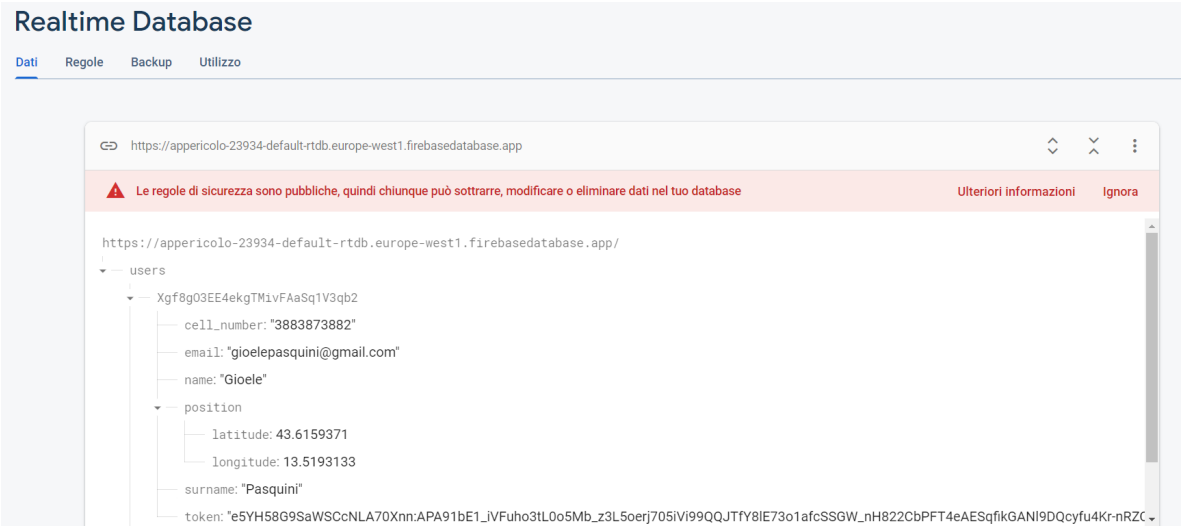

Figura 17: Porzione del Firebase Realtime Database

Per far funzionare le mappe presenti nell'applicazione, è necessario definire nel file *local.properties* la seguente variabile: MAPS_API_KEY=AIzaSyCorfjUSP9DKFfy6z9TpFW0mHMx2kCdK5g

2.3.1 Testing

Nello sviluppo dell'applicativo sono stati implementati diverse tipologie di test: unit test e Functional UI test. Per i local unit tests è stato utilizzato il framework di testing JUnit. In questo gruppo rientrano le classi di test per le funzioni utilizzate nella condivisione della posizione e le funzioni del dao dei contatti. Esse sono:

- La funzione di verifica se due posizioni sulla mappa sono vicine (ovvero incluse in un raggio di 150 metri) o lontane;
- La funzione di verifica se l'orario selezionato non è precedente all'orario attuale;
- La funzione del dao dell'entità Contatto utilizzata per inserire nuove istanze nel database Room.

La seconda tipologia di test è utile per simulare l'interazione dell'utente con l'applicativo e verificare che il comportamento dell'app sia quello previsto. Il framework di testing utilizzato è Espresso. I test eseguiti riguardano;

- La navigazione dal login alla main activity in caso di credenziali corrette;
- La segnalazione di errori quando si tenta di registrare un utente con dati incompleti

3 Sviluppo Flutter

3.1 Analisi dei requisiti

3.1.1 Requisiti funzionali e non funzionali

Nello sviluppo in Flutter le funzionalità dell'app sono state ridotte. Esse sono elencate qui di seguito:

- Registrazione di un nuovo account;
- Login e logout dall'applicazione;
- Visualizzazione della posizione attuale;
- Inserimento, visualizzazione e eliminazione dei contatti stretti a partire dalla rubrica.

3.1.2 Attori e Casi d'uso

Le limitate funzionalità dell'app consentono di identificare un'unico attore, che utilizza l'app per visualizzare la propria posizione e per selezionare i contatti fidati (Figura 18). Per poter usufruire delle funzionalità dell'app, l'utente deve essere autenticato.

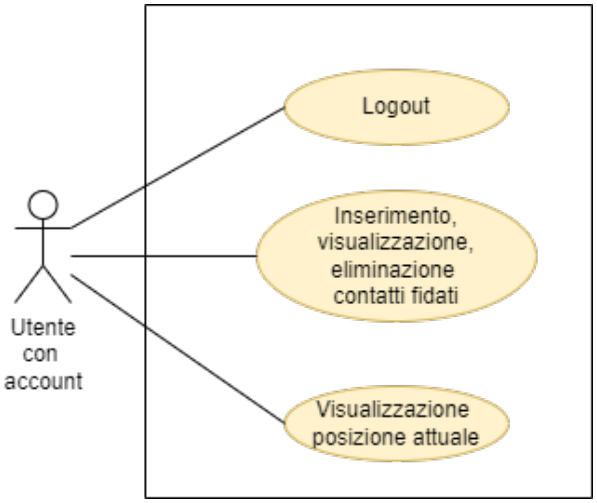

Figura 18: Diagramma dei casi d'uso relativi all'utente con account

3.2 Progettazione dell'applicazione

3.2.1 Architettura

Per lo sviluppo dell'applicazione in Flutter non è stato utilizzato nessun pattern particolare. Tutti i file ".dart" sono contenuti all'interno della cartella lib/screens, ad eccezione del punto di accesso dell'applicazione, il file "main.dart", contenuto nella cartella lib.

3.2.2 Gestione dei dati

Firebase
Come per la versione Kotlin dell'applicazione, si è usufruito del servizio Authentication per l'autenticazione (basata su email e password) degli utenti nell'app.
Per quanto riguarda la gestione dei dati, la versione Flutter di Appericolo fa uso di un database Firestore per il salvataggio dei contatti stretti selezionati dall'utente.
Una volta effettuato il logout, il database viene svuotato.

3.3 Implementazione dell'applicazione

All'avvio dell'applicazione (Figura 19), l'utente si trova di fronte una schermata in cui può inserire le credenziali per effettuare il login. Nel caso in cui l'utente si approcci per la prima volta all'app, in fondo alla schermata è presente un link che permette la registrazione di un nuovo account. Cliccando sul link, viene avviata la schermata per la registrazione che consente l'inserimento dei dati necessari. Dopo aver effettuato il login con successo, l'utente visualizza l'Homepage dell'app (Figura 20). In questa prima fase, all'utente si richiedono i permessi necessari per consentire all'app di accedere alla sua posizione. Una volta concessi i permessi, l'utente può visualizzare la propria posizione attuale. Dalla schermata Home, cliccando sulla relativa icona, l'utente può effettuare il logout o gestire la lista dei contatti fidati (Figura 21).

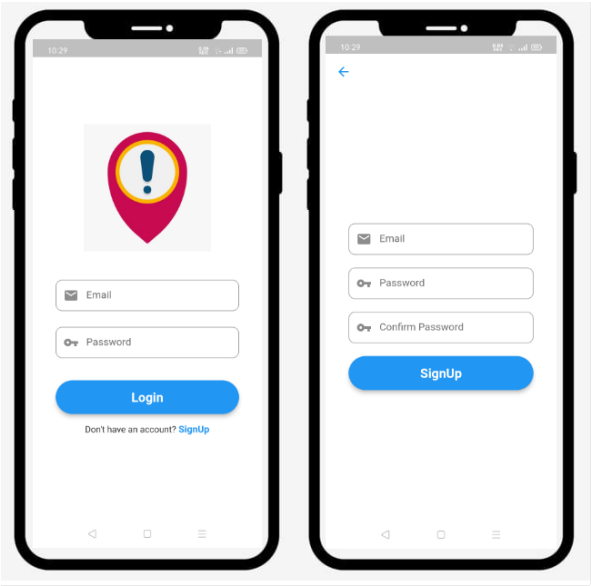

Figura 19: Login e Registrazione

Figura 20: Home

Nella schermata di visualizzazione dei contatti selezioni, cliccando sul singolo elemento è possibile eliminare il contatto dalla lista (Figura 22).

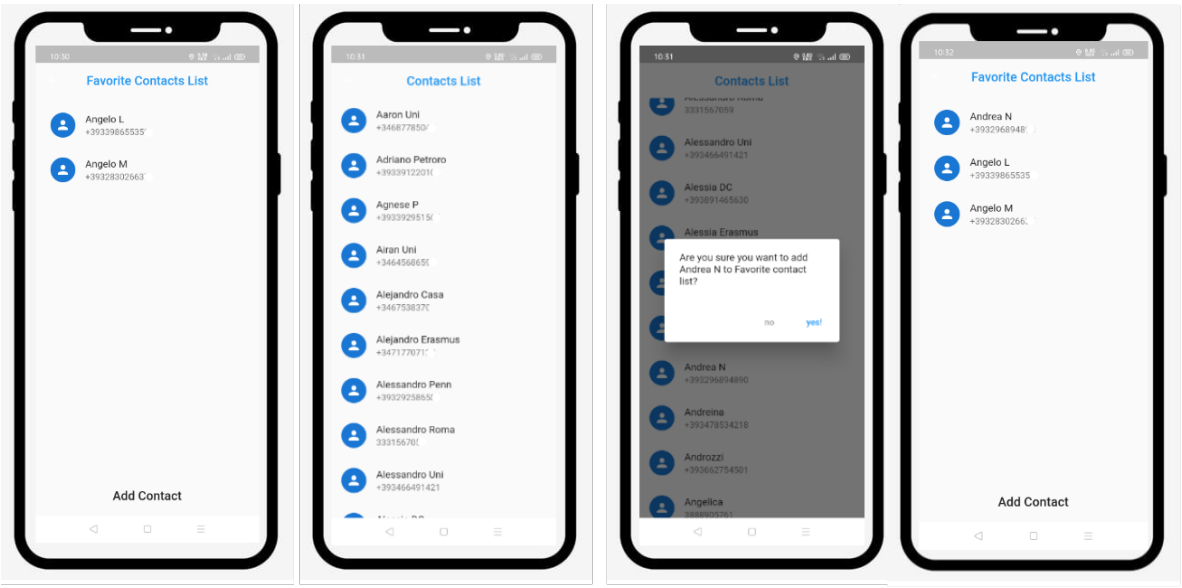

Figura 21: Aggiunta di un contatto stretto

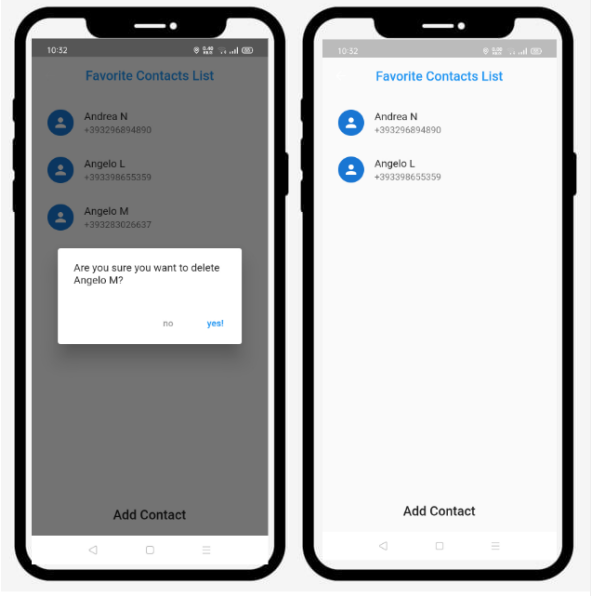

Figura 22: Eliminazione di un contatto stretto

Di seguito si mostra una porzione del Firebase Firestore Database:

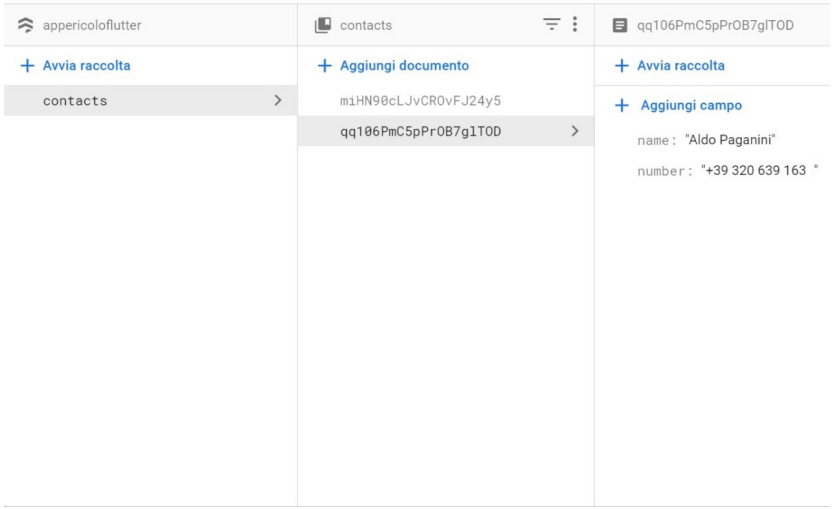

Figura 23: Porzione del Firebase Firestore Database

4 Conclusioni

Appericolo non è solo un'app, ma è un alleato tecnologico nella lotta alla sensazione di insicurezza su strada.
Si tratta di un progetto che ci ha permesso di applicare in maniera pratica concetti e metodologie studiate in altri corsi come Basi di Dati e Programmazione ad Oggetti.
Per la prima volta nella nostra carriera universitaria, sentiamo di aver fatto qualcosa di socialmente utile, di aver creato da zero un'applicazione che, anche con un solo utilizzo reale, ripagherà tutti i nostri sforzi.
Ringraziamo il professore Storti per aver accettato la nostra idea e per averci dato fiducia.