



# EventBuddy

Documentazione Progetto di Dispositivi Mobili 2023/2024

Gruppo Lanos:  
Alessandrini Simone - 876163  
Baraldi Andrea - 839452  
Lesinigo Simone - 899540  
Penati Lucrezia - 886092

# Sommario

<b>Introduzione.....</b>	<b>3</b>
<b>Caratteristiche principali.....</b>	<b>3</b>
<b>Github.....</b>	<b>3</b>
<b>Architettura.....</b>	<b>4</b>
<b>Librerie.....</b>	<b>5</b>
<b>Activity.....</b>	<b>6</b>
WelcomeActivity.....	6
LoginActivity.....	7
RegistrationActivity.....	8
ForgottenPasswordActivity.....	9
AccountSettingsActivity.....	10
PreferencesSettingsActivity.....	11
CreateEventActivity.....	12
EventDetailActivity.....	13
<b>Permette all'utente di visualizzare i dettagli di un evento:.....</b>	<b>13</b>
<b>Fragment.....</b>	<b>14</b>
SettingsFragment.....	14
FriendsFragment.....	15
EventFragment.....	16
ActiveFragment.....	17
<b>UI Layer.....</b>	<b>18</b>
ViewModel.....	18
<b>Data layer.....</b>	<b>18</b>
Repository.....	18
Worker.....	18
Services.....	18
Data Sources.....	19
Dao.....	19
Models.....	19
Util.....	20
<b>Design.....</b>	<b>20</b>
<b>Sviluppi futuri.....</b>	<b>21</b>

# Introduzione

EventBuddy è un'applicazione Android dedicata alla gestione e all'organizzazione di eventi, progettata per semplificare l'esperienza dell'utente nella pianificazione di incontri con i propri amici. Mezz'ora prima dell'inizio dell'evento, l'app metterà a disposizione una mappa che in tempo reale visualizzerà la posizione di tutti gli utenti che partecipano all'evento.

## Caratteristiche principali

### 1. Creazione di eventi

- Creare facilmente eventi personalizzati, specificando nome, data, ora e luogo.
- Aggiungere dettagli come la descrizione, per rendere l'evento più completo possibile

### 2. Inviti agli amici

- Invitare i propri amici a partecipare agli eventi creati.
- Gli amici avranno la possibilità di decidere se partecipare o meno.

### 3. Tracciamento in tempo reale

- Attivare il tracker mezz'ora prima dell'ora di inizio dell'evento per visualizzare la posizione in tempo reale di tutti gli utenti sulla mappa.

### 4. Interfaccia

- Design pulito e semplice per garantire una navigazione agevole.
- Accesso rapido alle funzionalità principali per una gestione efficiente degli eventi.

## Github

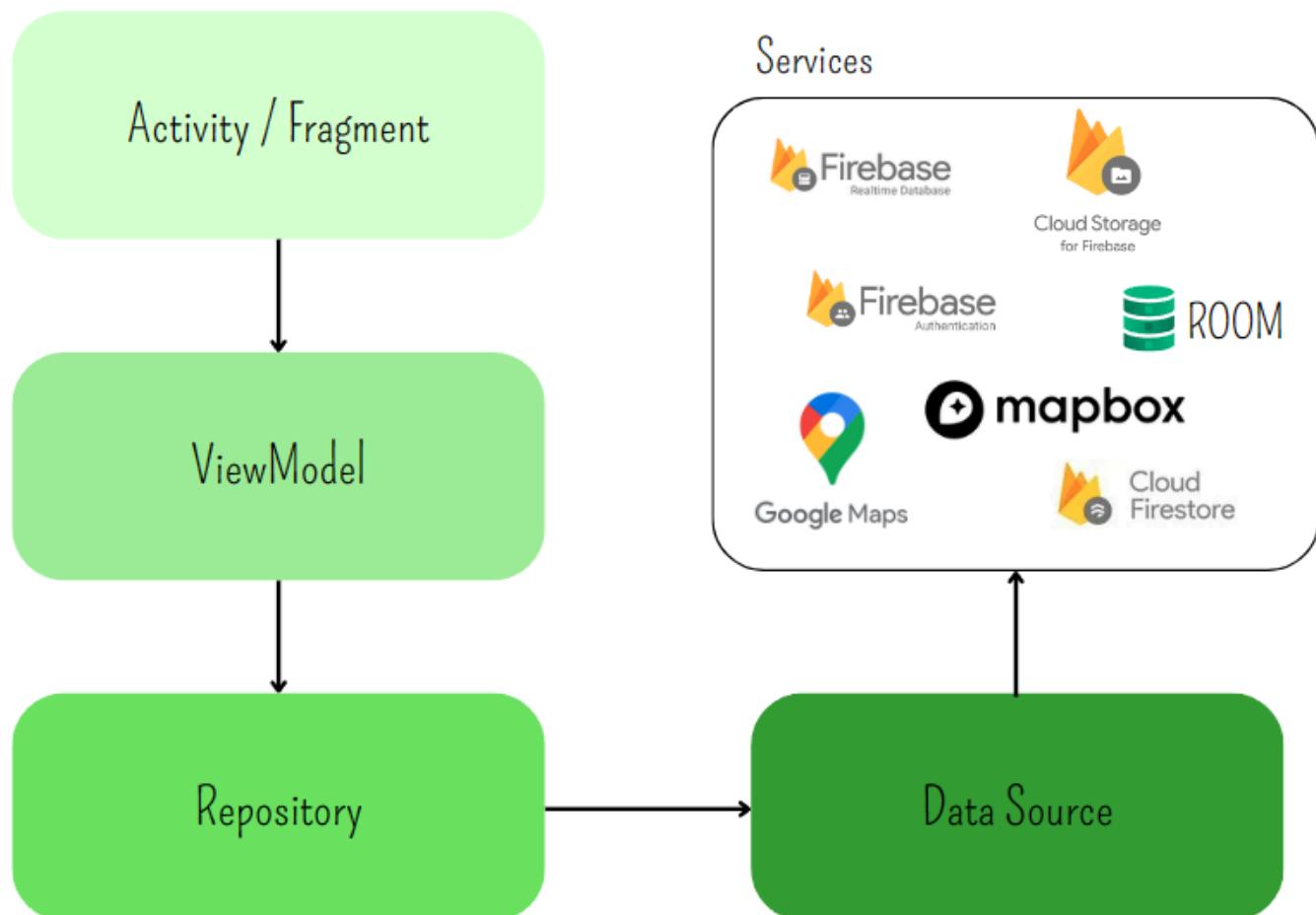
Per la gestione del progetto abbiamo deciso di dividere la repository nei seguenti branch:

- **Master**: questo branch contiene la release finale dell'applicazione, non è stato fatto nessuno sviluppo ma solamente il merge degli altri branch
- **Backend**: in questo branch è stato fatto lo sviluppo backend dell'applicazione
- **Frontend**: in questo branch è stato fatto lo sviluppo frontend dell'applicazione
- **Dev**: branch utilizzato per collegare il backend al frontend

# Architettura

L'architettura dell'applicazione si compone di due strati principali, strutturati in varie sezioni, come suggerito dal sito di android developers :

- **UI layer:** il layer UI si occupa di mostrare i dati a schermo ed è composto da due sezioni:
  - *UI elements*: sono l'insieme di activity e fragment che gestiscono la componente visiva dell'app, mostrando i dati e le informazioni sullo schermo. Gestiscono inoltre il passaggio di dati tra gli elementi di ui e le transizioni da una activity ad un'altra
  - *State holders*: sono l'insieme dei viewmodel che gestiscono la persistenza dello stato dei dati che vengono visualizzati dagli elementi della UI e si occupano di creare un collegamento tra frontend e backend.
- **Data layer:** il layer dei dati si occupa della logica di business, ovvero come i dati vengono creati, archiviati e modificati. Questo layer è composto da due sezioni:
  - *Repository*: sono le classi che espongono i dati al resto dell'applicazione, li elaborano e li gestiscono, centralizzando queste operazioni.
  - *Data sources*: sono le classi che collegano l'applicazione e i vari servizi che operano sui dati. Ogni data source ha il compito di lavorare con una singola fonte di dati e per questo può appartenere ad uno o più repository



Vengono inoltre utilizzati i seguenti servizi esterni:

- **l'API di MapBox**, per ottenere i suggerimenti automatici per completare la scrittura di un indirizzo;
- **Maps SDK for Android**: per la visualizzazione delle mappe
- **Firebase Authentication**: per l'autenticazione degli utenti, usando indirizzo email e password
- **Firebase Realtime Database**: per il salvataggio e la sincronizzazione delle location.
- **Firebase Cloud Firestore**: per il salvataggio delle informazioni degli utenti e degli eventi
- **Firebase Storage**: per il salvataggio delle immagini di profilo dell'utente

## Librerie

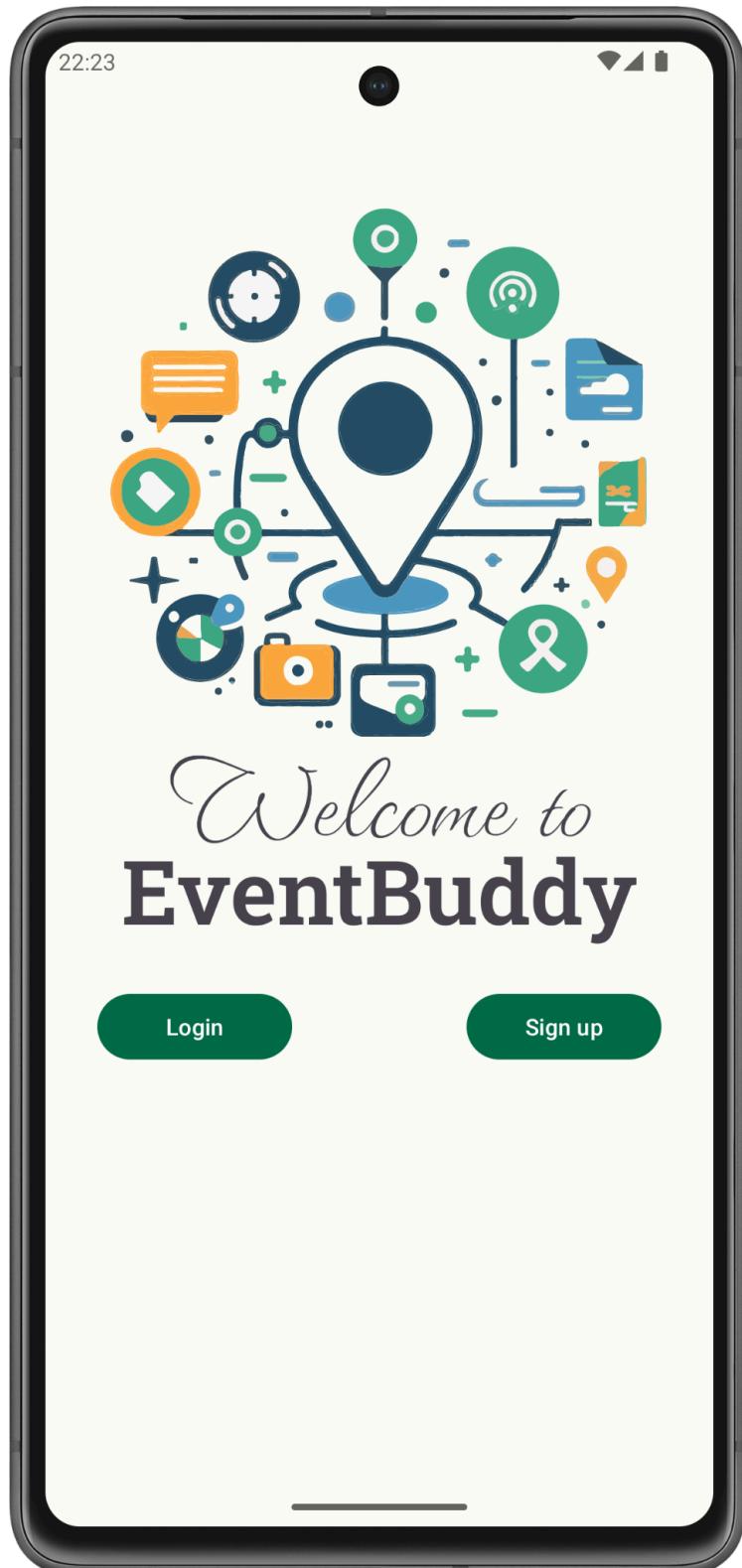
Sono state utilizzate le seguenti librerie, oltre a quelle standard per lo sviluppo:

1. **Glide** per la gestione delle immagini;
2. **Room** per il database offline;
3. **Retrofit** per l'autocompletamento degli indirizzi degli eventi.

# Activity

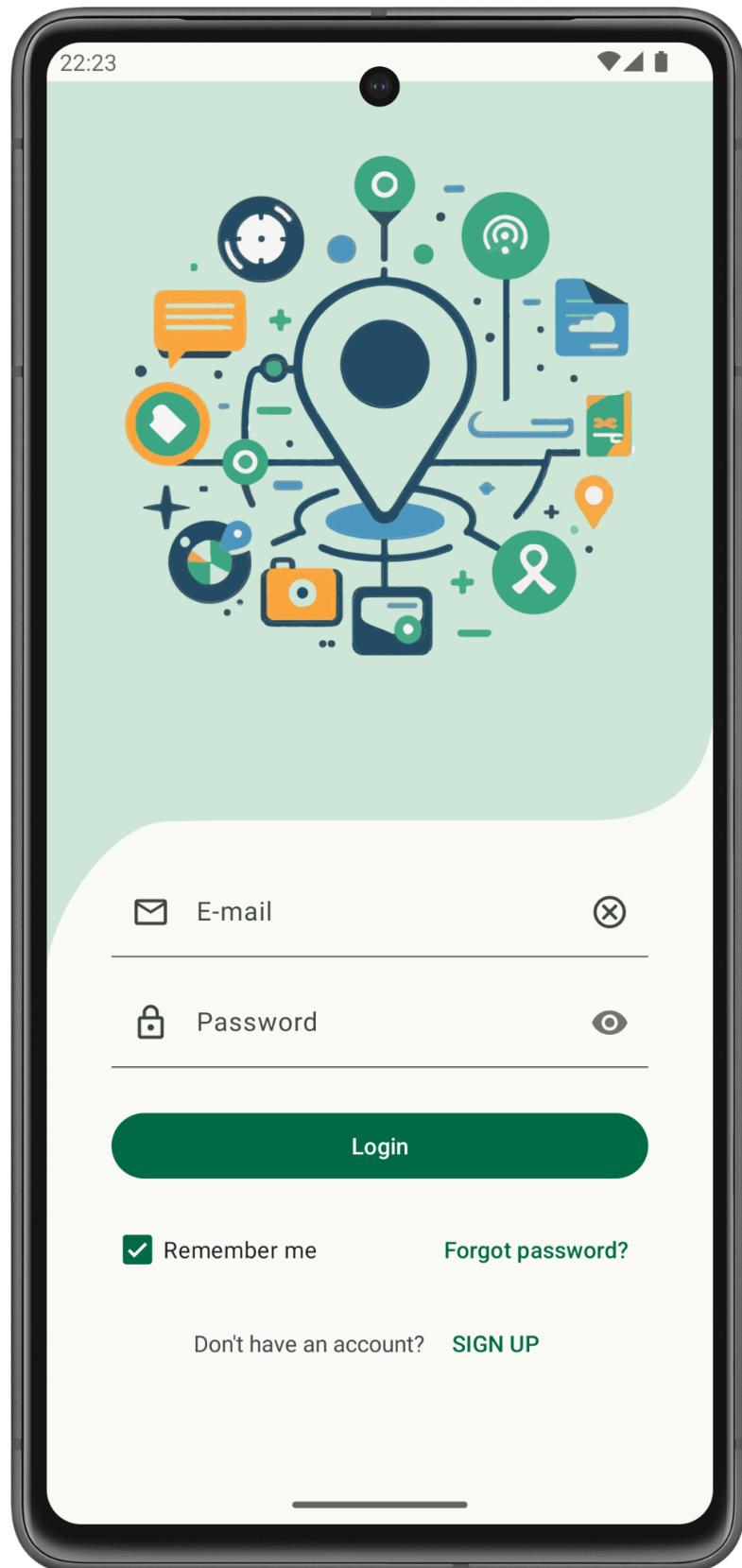
## WelcomeActivity

È la prima schermata che un utente visualizza quando non ha ancora effettuato l'accesso all'applicazione. Consente di scegliere se registrarsi creando un account o se effettuare l'accesso.



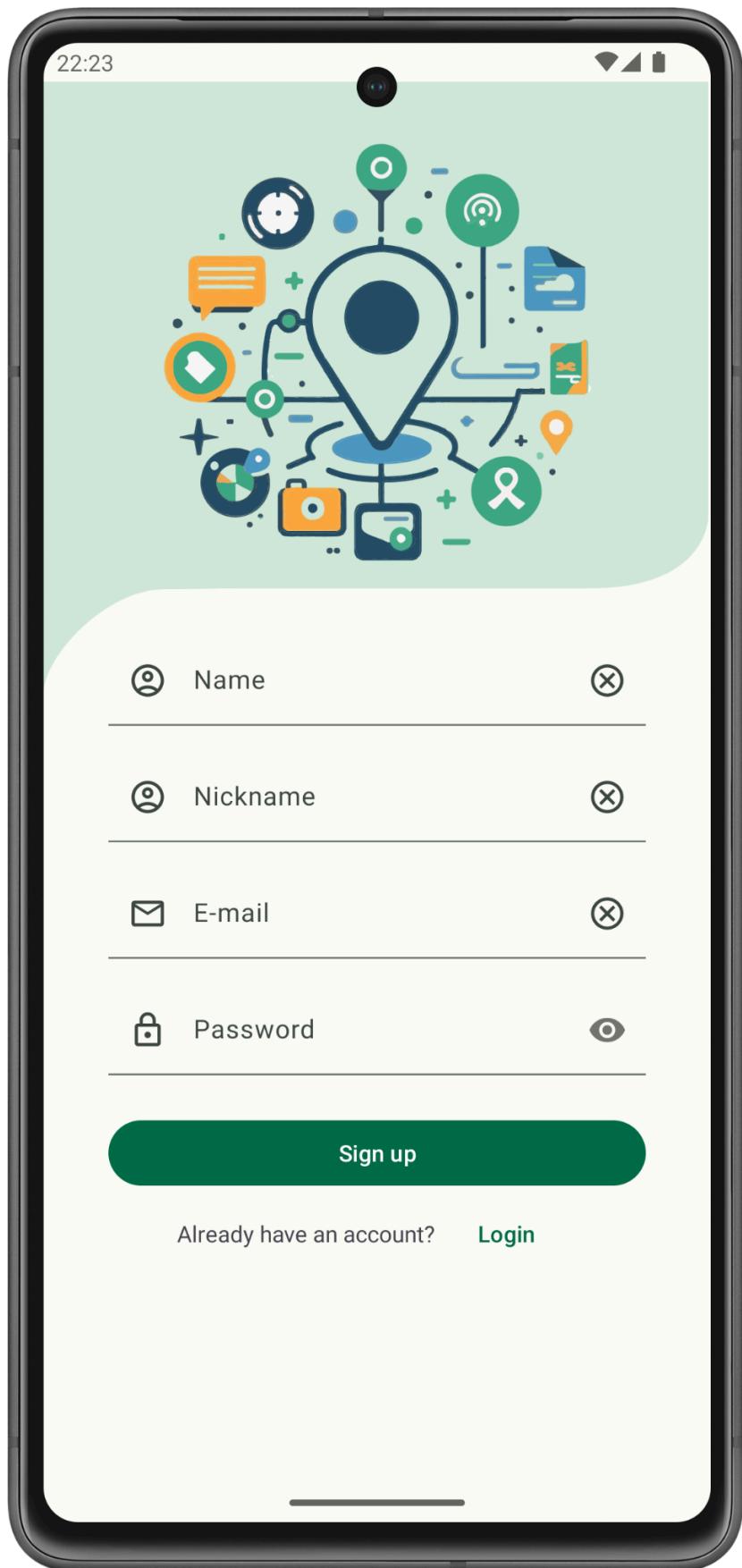
## LoginActivity

Permette all'utente di accedere al proprio account utilizzando le credenziali impostate durante la registrazione, cioè email e password.



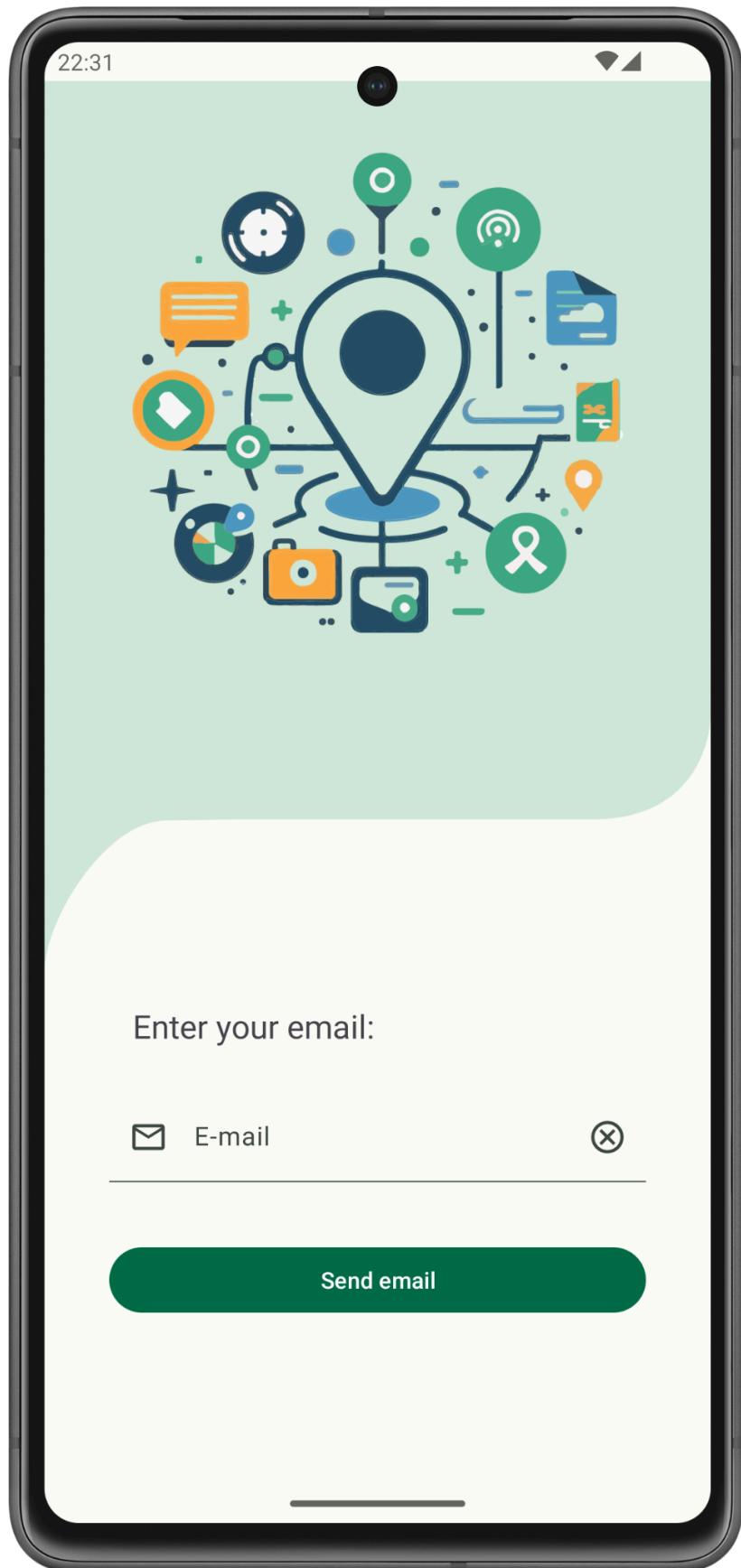
## RegistrationActivity

Consente all'utente di registrarsi all'applicazione. È necessario inserire il proprio nome completo, un nickname, il proprio indirizzo e-mail e una password di almeno sei caratteri.



## ForgottenPasswordActivity

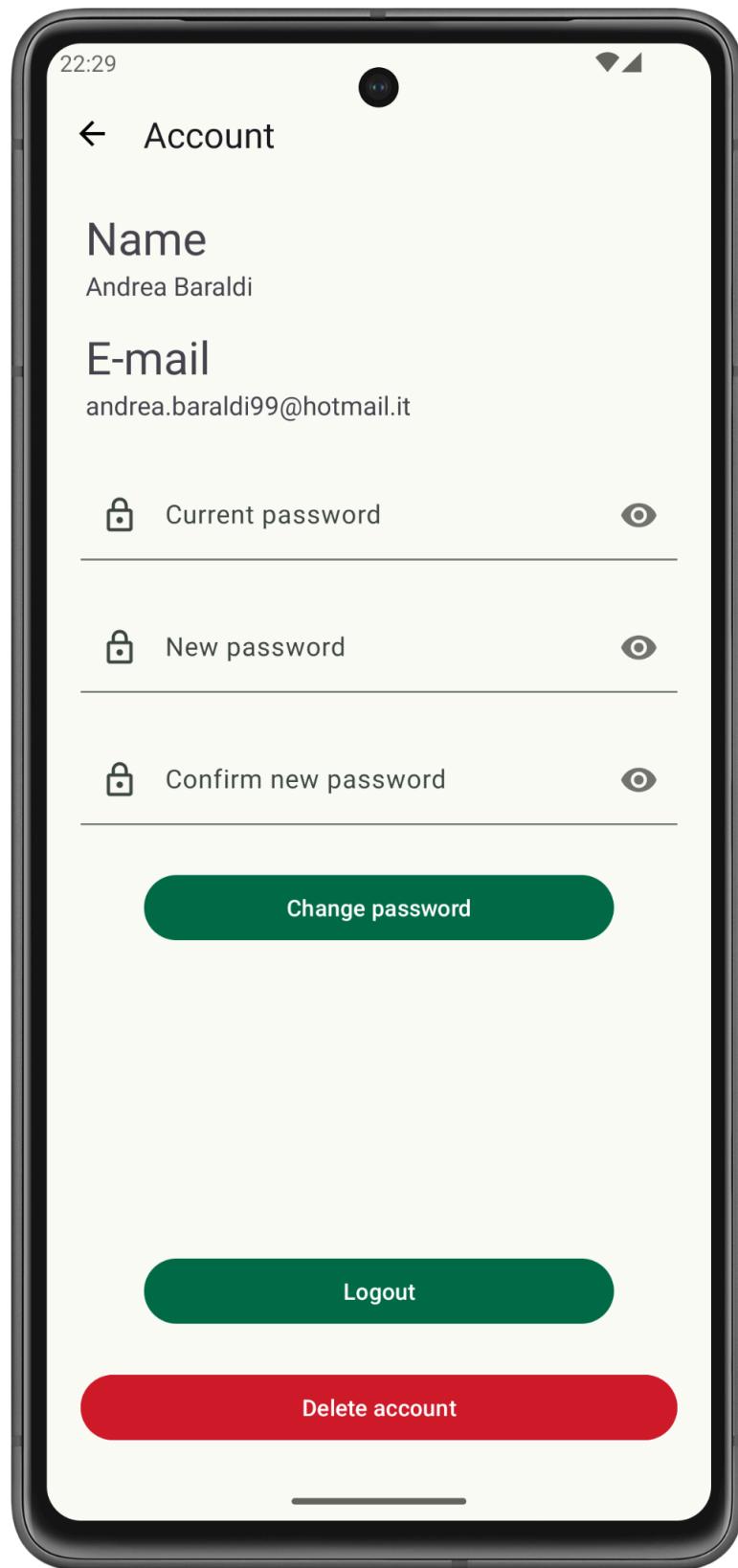
Permette all'utente di reimpostare la propria password. Inserendo l'indirizzo e-mail, riceverà una e-mail contenente un link per il cambio password nel caso esista un account associato a quella e-mail.



## AccountSettingsActivity

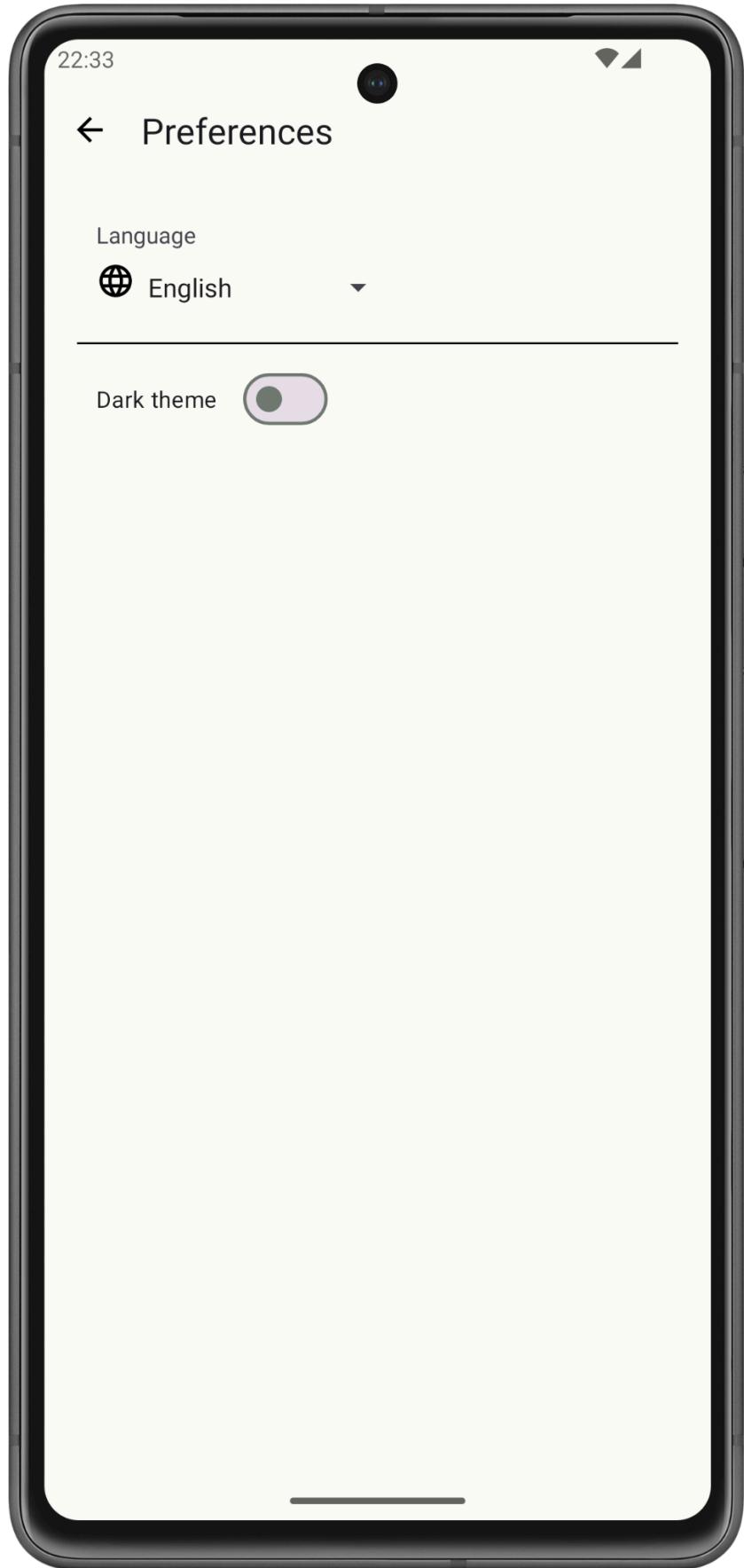
Permette all'utente di:

1. Visualizzare il proprio nome completo e l'indirizzo email;
2. Cambiare la password;
3. Eseguire il logout;
4. Cancellare il proprio account.



## PreferencesSettingsActivity

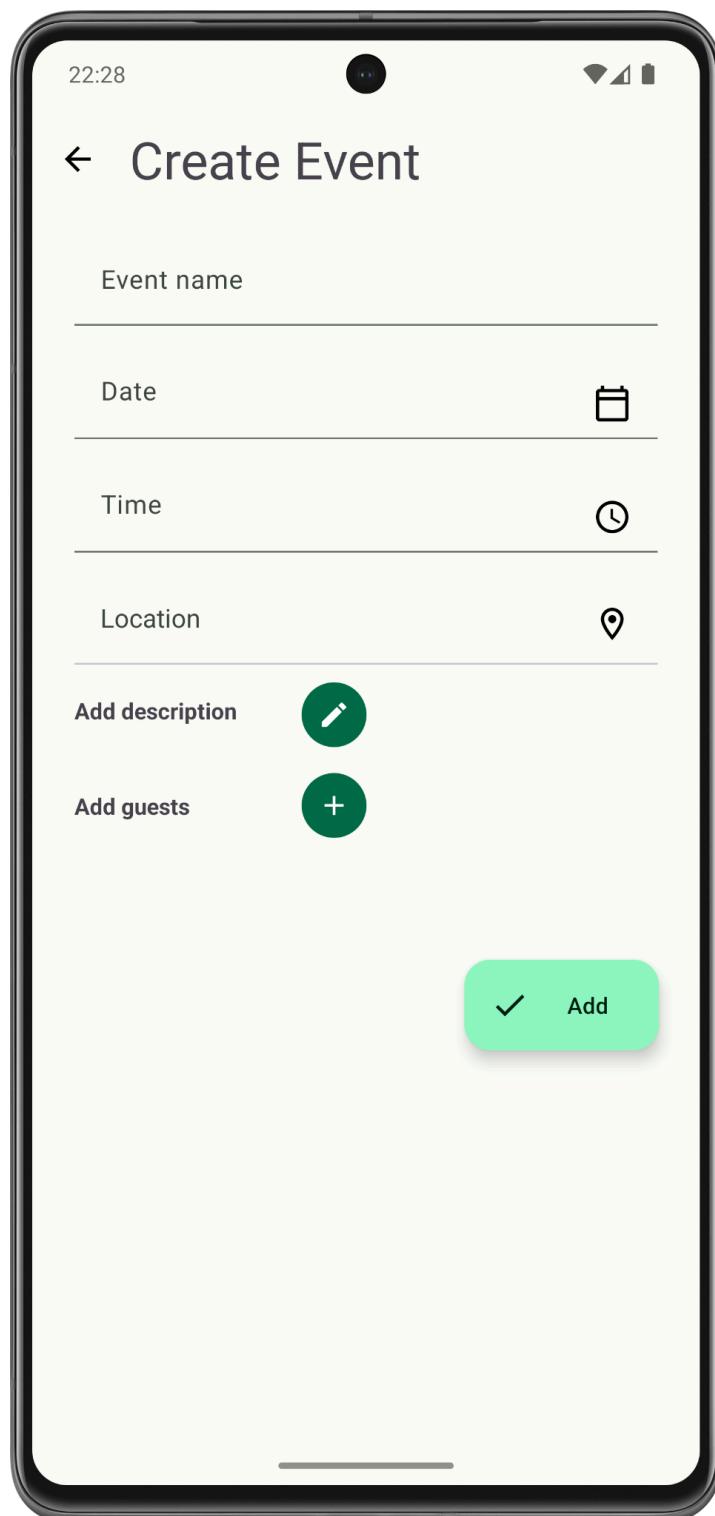
Consente all'utente di selezionare la lingua dell'applicazione, che è disponibile in italiano oppure in inglese, e di attivare o disattivare il tema scuro.



## CreateEventActivity

Permette all'utente di creare un nuovo evento di cui è il manager. Si devono inserire:

1. Nome dell'evento;
2. Data dell'evento;
3. Ora dell'evento;
4. Luogo dell'evento;
5. Descrizione dell'evento;
6. Aggiungere gli amici che si desiderano invitare.



## EventDetailActivity

Permette all'utente di visualizzare i dettagli di un evento:

1. Nome
2. Descrizione
3. Luogo
4. Data e ora
5. Manager
6. Se si clicca sul numero di partecipanti si potrà vedere, tra chi è invitato, chi partecipa e chi non partecipa all'evento
7. Mappa con marker sul luogo dell'evento

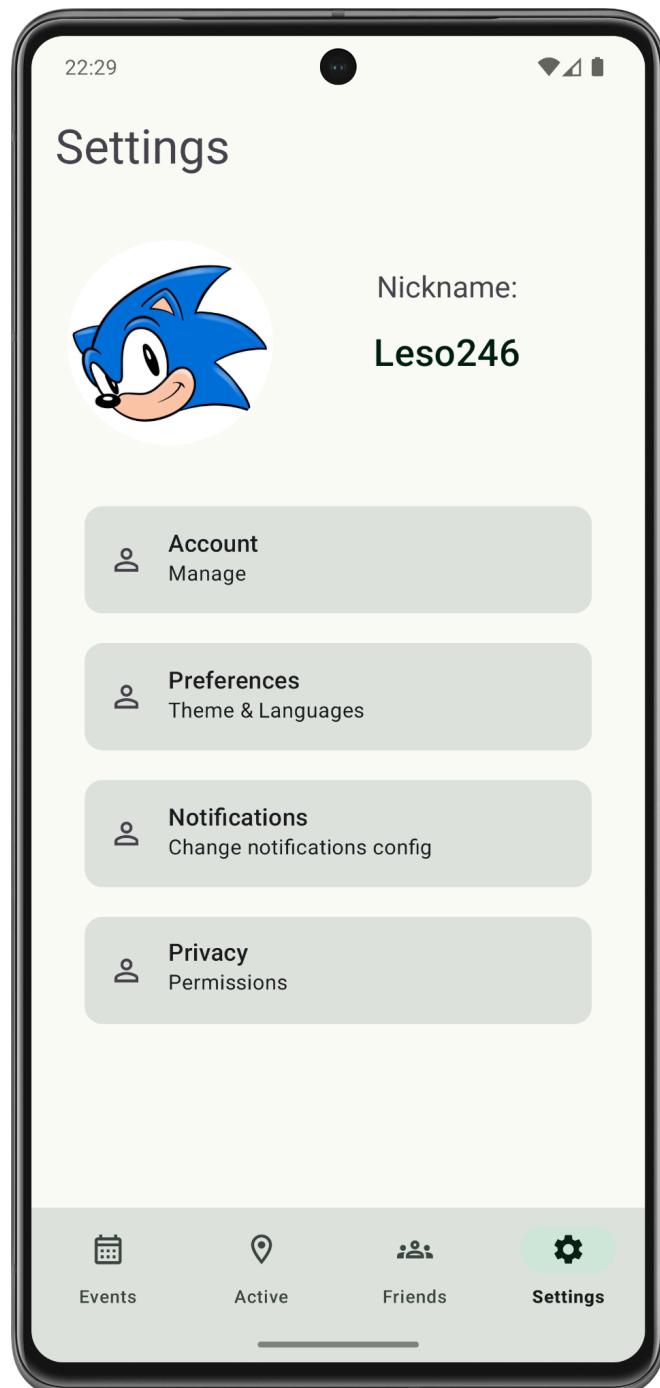


# Fragment

## SettingsFragment

Permette all'utente di:

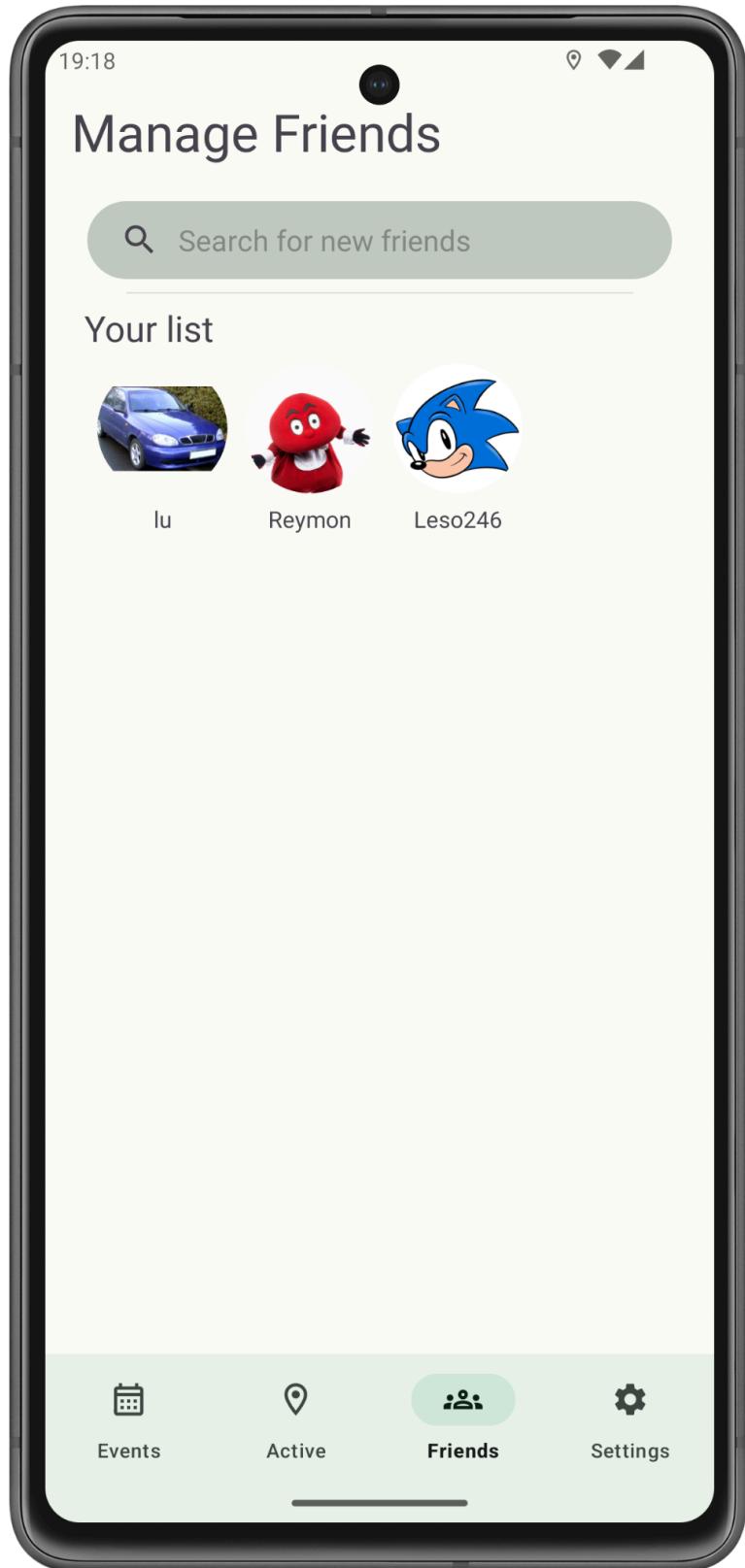
1. Cambiare l'immagine del profilo cliccando su di essa.
2. Visualizzare e modificare il proprio nickname cliccando su di esso.
3. Modificare le impostazioni delle notifiche inviate dall'applicazione.
4. Modificare i permessi dell'applicazione.



## FriendsFragment

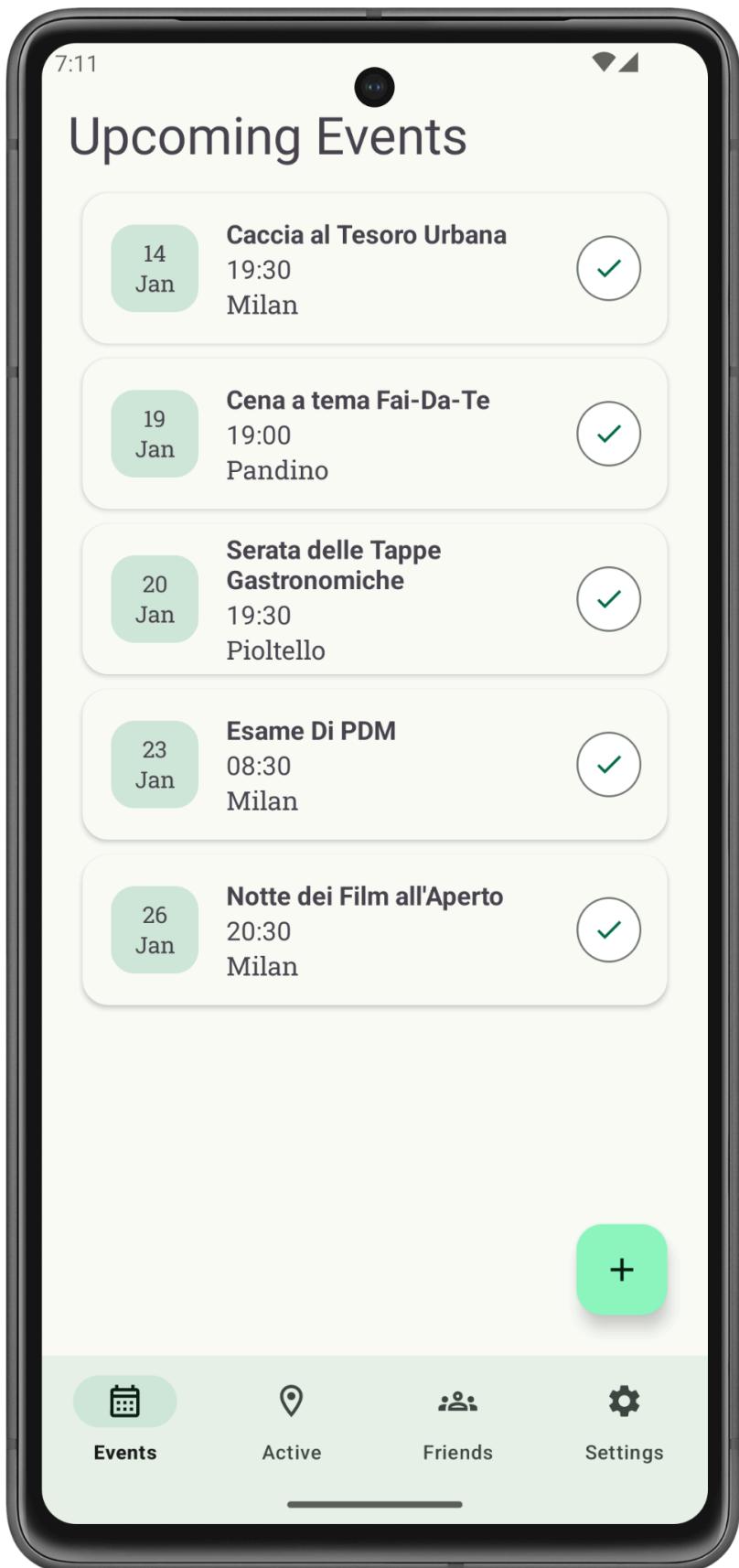
Consente all'utente di:

1. Visualizzare la lista degli amici.
2. Rimuovere amici dalla propria lista cliccando sull'immagine dell'utente desiderato.
3. Aggiungere nuovi amici cercandoli tramite la barra di ricerca.



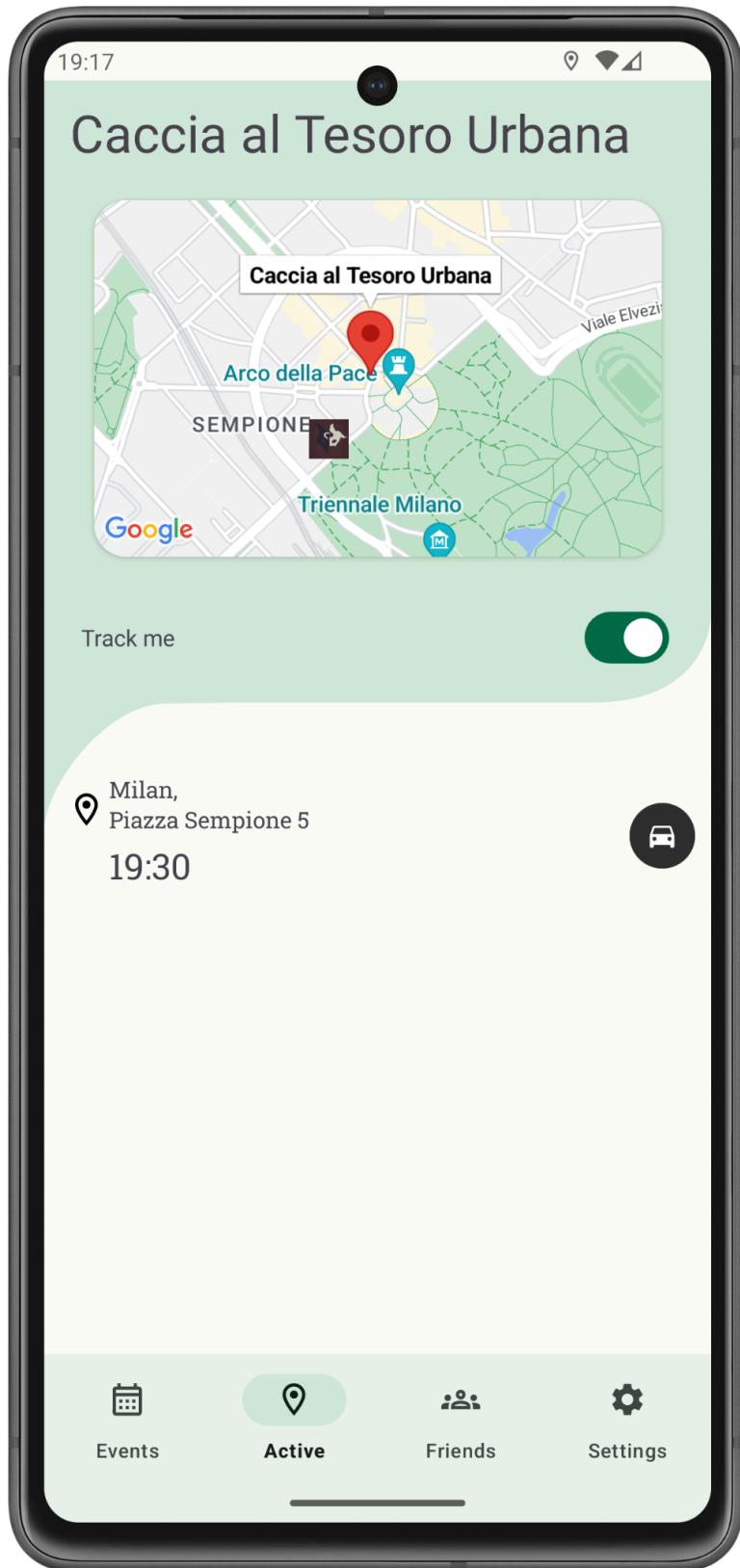
## EventFragment

In questo Fragment è possibile visualizzare i propri eventi e quelli creati dagli amici a cui si è stati invitati e a cui si è deciso di partecipare. Cliccando sull'evento è possibile visualizzarne i dettagli e decidere se partecipare o meno, cliccando sulla spunta a destra.



## ActiveFragment

In questo Fragment viene mostrato l'evento più recente il cui inizio è previsto entro mezz'ora dall'orario attuale. Sulla mappa vengono visualizzati in tempo reale gli spostamenti di tutti gli utenti che partecipano all'evento, a condizione che abbiano attivato l'opzione "tracciami". Cliccando sul pulsante sulla destra, si sarà reindirizzati alla posizione dell'evento su Google Maps.



# Descrizione delle classi

## UI Layer

### ViewModel

- **EventViewModel**: si occupa di recuperare gli eventi dalla repository EventRepository e di renderli disponibili alle Activity o Fragment che lo istanziano.
- **FriendsViewModel**: si occupa di recuperare gli utenti dalla repository UserRepository e di renderli disponibili alle Activity o Fragment che lo istanziano.
- **UserViewModel**: si occupa di gestire l'autenticazione dell'utente interfacciandosi con lo UserRepository
- **LocationViewModel**: si collega al repository LocationRepository e rende disponibili le informazioni relative le posizioni in tempo reale degli utenti all'Active Fragment
- **CreateEventViewModel**: si collega al repository SuggestionRepository permettendo la gestione dei dati riguardanti le location all'interno della CreateEventActivity.

## Data layer

### Repository

Le repositories collegano i ViewModel al DataSource

- **EventRepository**: si occupa di interfacciarsi con i DataSource per la gestione degli eventi
- **LocationRepository**: si occupa di interfacciarsi con i DataSource per la gestione delle posizioni per il tracciamento in tempo reale
- **SuggestionsRepository**: si occupa di interfacciarsi con i DataSource per l'autocompletamento degli indirizzi quando si inserisce il luogo nella creazione di un evento
- **UserRepository**: si occupa di interfacciarsi con il DataSource per la gestione delle credenziali dell'utente

### Worker

- **UpdateEventsWorker**: si occupa di controllare in background se ci sono nuovi eventi da visualizzare

### Services

Abbiamo utilizzato delle classi di servizio sia nel caso fossimo obbligati dalle librerie (come Retrofit) sia quando i data source, per una maggiore flessibilità, utilizzavano più volte le stesse funzioni dalle stesse sorgente di dati

- **CloudDBService**: si occupa di gestire le operazioni CRUD non specializzate del database cloud firestore
- **MapboxService**: si occupa di chiamare le API di Mapbox tramite Retrofit per gestire l'autocompletamento degli indirizzi degli eventi

## Data Sources

- **UserDataSource**: si occupa di gestire la registrazione, l'autenticazione e la modifica dell'utente tramite Firebase Authentication
- **ImageRemoteDataSource**: si occupa di gestire l'upload delle immagini di profilo degli utenti sul Firebase Storage
- **EventsRemoteDataSource**: si occupa di scrivere, leggere e aggiornare gli eventi all'interno del Cloud Firestore
- **UserRemoteDataSource**: si occupa di scrivere, leggere e aggiornare gli utenti e i loro amici all'interno del Cloud Firestore
- **LocationRemoteDataSource**: si occupa di mandare la posizione dell'utente e di notificare l'app con le posizioni degli altri utenti che partecipano ad un evento tramite il Realtime Database
- **EventsLocalDataSource**: si occupa di scrivere, leggere e aggiornare gli eventi all'interno del database locale
- **UserLocalDataSource**: si occupa di scrivere, leggere e aggiornare gli utenti e i loro amici all'interno del database locale
- **AutoCompleteRemoteDataSource**: si occupa di gestire le richieste per l'autocompletamento degli indirizzi tramite il MapboxService

## Dao

- **EventDao**: si occupa di effettuare le query al database locale per quanto riguarda gli eventi
- **UserDao**: si occupa di effettuare le query al database locale per quanto riguarda gli utenti

NOTA: la creazione del database è affidata alla classe **EventsRoomDatabase**

## Models

- **Event**: rappresenta un evento
- **EventsCloudResponse**: rappresenta un evento richiesto online
- **EventsWithusersFromCloudResponse**: rappresenta l'evento con gli utenti che vi partecipano richiesto online
- **EventWithUsers**: rappresenta l'evento con gli utenti che vi partecipano
- **Location**: rappresenta la posizione di un utente
- **Result**: rappresenta il risultato di una chiamata online/locale
- **User**: rappresenta un utente
- **UserEventCrossRef**: rappresenta la relazione molti a molti tra gli utenti e gli eventi
- **UserFromRemote**: rappresenta un utente richiesto online
- **UserWithEvents**: rappresenta l'utente e i suoi eventi
- **Package mapbox**: Modelli generati per ricevere i dati dall'API di Mapbox

## Util

- **Constants**: contiene tutte le costanti utilizzate
- **DataEncryptionUtil**: utilizzata per scrivere i dati dell'utente in un file criptato
- **DateTimeComparator**: utilizzata per ordinare gli eventi in EventFragment
- **LocationService/ServiceLocator/DefaultLocationClient/LocationApp**: si occupa di richiedere un aggiornamento della posizione dell'utente periodicamente, la quale poi viene sfruttata nell'activeFragment
- **MyAppGlideModule**: necessario per il funzionamento di Glide
- **Parser**: contiene dei parser per la formattazione dei dati in modo da poter essere visualizzati correttamente nelle varie activity/fragment
- **SharedPreferencesUtil**: classe che permette di scrivere e leggere valori tramite le SharedPreferences

## Design

Lo stile dell'applicazione EventBuddy è stato realizzato seguendo le linee guida di Material Design 3 e sono disponibili due temi: chiaro e scuro. I colori dei rispettivi temi sono stati scelti utilizzando il sito <https://m3.material.io/theme-builder#/custom> e scegliendo come colore primario #509873, il quale viene ripreso anche nel logo dell'applicazione stessa. Per quanto riguarda quest'ultimo, abbiamo pensato che poiché si tratta di un'app di creazione, gestione, e tracciamento di eventi l'icona della posizione sia quella che rispecchia di più questo tema.



Prima di procedere con la programmazione effettiva dell'app, abbiamo utilizzato Figma, un sito per la creazione di design, per realizzare la prima versione dei layout delle varie schermate su cui basarci poi durante la scrittura del codice. Questo primo design dell'applicazione è visibile al seguente indirizzo Web: [https://www.figma.com/file/3AssHLWV0CuKslnkvOSdb2/Material-3-Design-Kit-\(Community\)?type=design&node-id=54495-66&mode=design&t=B1tsljCH2610NBrm-0](https://www.figma.com/file/3AssHLWV0CuKslnkvOSdb2/Material-3-Design-Kit-(Community)?type=design&node-id=54495-66&mode=design&t=B1tsljCH2610NBrm-0).

## Sviluppi futuri

- Dynamic Colors: vorremmo permettere di adattare il tema in base allo sfondo utilizzato dall'utente, in modo da rendere la sua esperienza ancora migliore.
- Implementare la possibilità di creare gruppi di amici per facilitare l'aggiunta degli invitati agli eventi, in modo tale che selezionando il gruppo da invitare si invitano in automatico tutti gli utenti che ne fanno parte.
- Chat di gruppo dedicata per ogni evento, per facilitare la comunicazione tra i partecipanti.
- Condivisione di foto e messaggi per rendere l'esperienza più interattiva e coinvolgente.
- Statistiche riguardanti la partecipazione di un utente: a quanti eventi ha preso parte, quante volte è arrivato in orario/ritardo...
- Implementare la cancellazione degli eventi sia nel caso in cui si vogliano volutamente annullare sia nel caso in cui ormai siano passati. (Questa funzionalità era stata già considerata ma è stata messa in secondo piano in quanto è necessario costruire un gestore del database online esterno all'applicazione, per esempio tramite Firebase Functions che però risulta essere a pagamento).
- Preparazione dell'applicazione per il deploy nell'app store: è necessario oscurare i dati sensibili all'interno dei database online.