

Appello 13/02/2024

Progetto **Enjoyn**



Gruppo APPenheimer

Marco Ferioli - Matricola 879277

Luca Pincioli - Matricola 885969

Stefano Spinelli - Matricola 887511

Francesco Trolli - Matricola 889039

Giulia Raffaella Vitale - Matricola 885938

Indice

Introduzione	1
Che cos'è Enjoyn?	1
Esempio di utilizzo (1)	1
Esempio di utilizzo (2)	2
Architettura	3
Activity e Fragment	4
Model	5
Adapter	6
Viewmodel	6
Repository	7
Service	7
DataSource	8
Database	9
Util	9
Servizi esterni	10
Come utilizzare l'applicazione	11
Registrazione	11
Conferma e-mail	12
Login	12
Configurazione del profilo	13
Scelta delle categorie	13
Recupero della password	13
Corpo dell'applicativo	14
Scopri - mappa	14
Scopri - Lista degli eventi	15
Todo	16
Nuovo	16
Profilo	17
Sviluppi futuri per l'applicativo	18

Introduzione

Che cos'è Enjoyn?

Enjoyn è un'app nativa sviluppata per dispositivi mobili che utilizzano il sistema operativo Android. L'applicazione ha come obiettivo l'organizzazione di diverse attività tra amici, includendo l'incontro con nuove persone per condividere le proprie esperienze, fare nuove conoscenze e scoprire nuovi interessi.

Le principali funzionalità fornite dall'app sono:

- **Registrazione** tramite e-mail con conseguente creazione del proprio profilo personale;
- **Validazione e verifica** dell'account utente tramite e-mail di controllo;
- **Accesso** tramite login in-app;
- **Configurazione del proprio profilo** tramite immagine, descrizione, nome e cognome (facoltativi);
- **Scelta di un insieme di categorie** che definiscono l'interesse dell'utente nei confronti di determinate attività;
- **Visualizzazione di attività condivise** da altri utenti;
- **Ricerca sulle mappe di luoghi** in cui poter trovare attività proposte da altri utenti;
- **Partecipazione** tramite il proprio profilo personale;
- **Creazione di attività** visibili ad altri utenti;
- **Collegamento con google maps** per percorso fino al posto in cui si svolge l'evento;

Esempio di utilizzo (1)

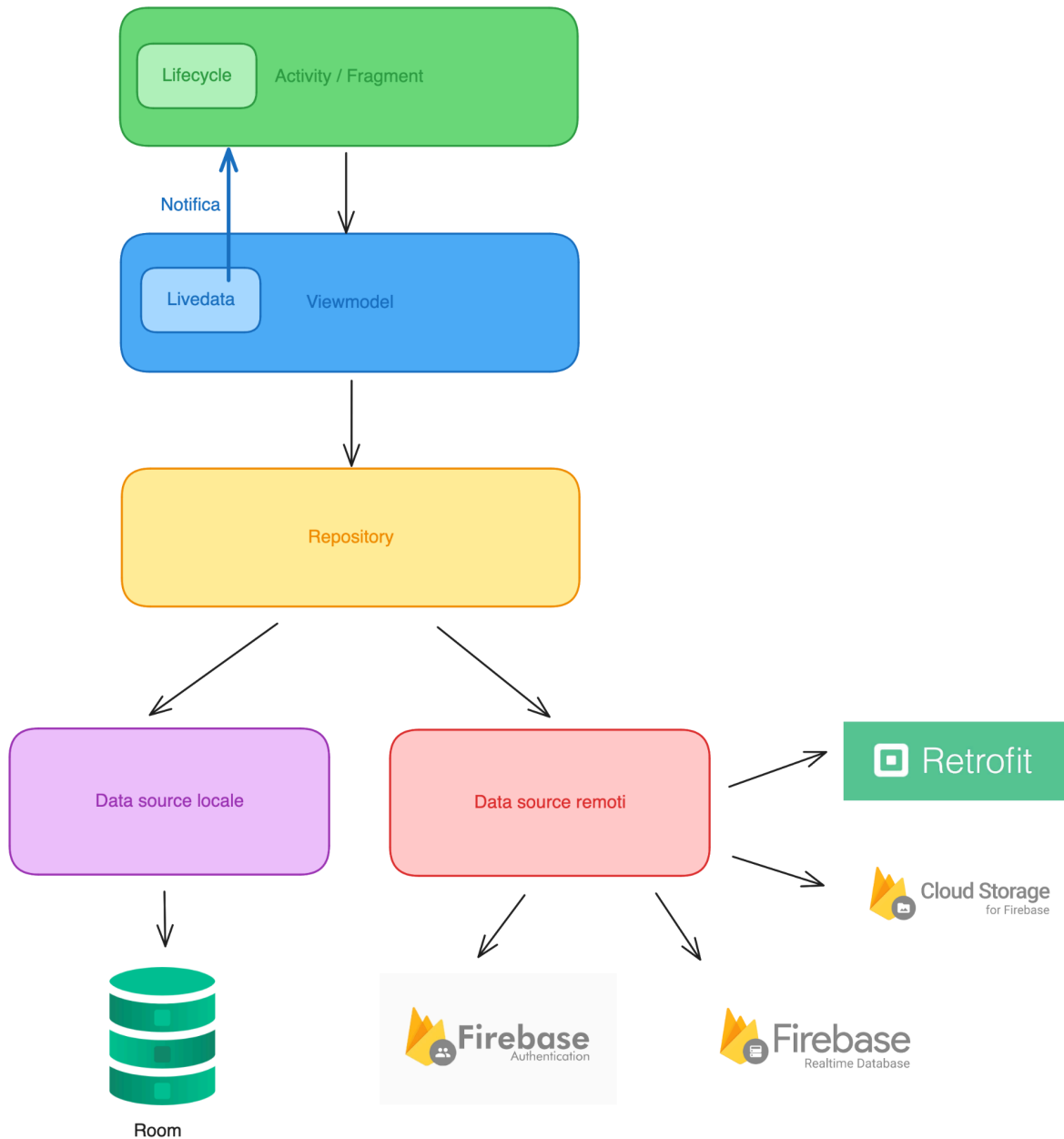
Dopo essermi appassionato al gioco degli scacchi, vorrei imparare a giocare e confrontarmi con altri giocatori dal vivo. Accedo quindi all'app Enjoyn e cerco, tramite il sistema di ricerca, gli eventi legati agli scacchi che ci sono nelle vicinanze

(rispetto alla mia posizione) e scopro che il circolo degli scacchi di Legnano ha organizzato un torneo e offre lezioni gratuite dal vivo per neofiti. Decido quindi di iscrivermi all'evento e di parteciparvi.

Esempio di utilizzo (2)

È da molto tempo che desidero compiere il cammino della “Via degli dei” da Bologna a Firenze per vedere le bellezze naturali che circondano il territorio degli Appennini. Poichè vorrei condividere la mia esperienza con altre persone, utilizzo l'app Enjoyn e pubblico un evento nella data in cui vorrei compiere l'escursione. Con grande felicità scopro che molte altre persone compiranno le stesse tappe del cammino nel periodo che ho scelto, poiché queste ultime parteciperanno all'evento tramite l'app. Ho quindi trovato dei compagni con cui realizzare la mia avventura!

Architettura



L'architettura è stata realizzata in riferimento a quella descritta nella documentazione Android ed a quella analizzata durante le varie esercitazioni. In questa sezione viene descritta dettagliatamente l'architettura dell'app, i componenti e servizi utilizzati, oltre che le responsabilità delle varie parti che compongono il sistema.

Sono stati applicati i principi suggeriti da Android per produrre un'architettura corretta e scalabile:

- **Separation of Concerns**
Ogni componente del sistema ha una responsabilità.
- **Single Source of truth**
La fonte di provenienza di ciascun tipo di dato è univoca.
- **Unidirectional data flow**
Il flusso di operazioni per ottenere dati dalle varie fonti è unidirezionale, mentre quello per modificare tali dati fluisce nel senso opposto.

Activity e Fragment

Activity e Fragment contengono la logica relativa alla presentazione dei dati. Nella fase di creazione vengono popolati dai vari componenti grafici e vengono definite le operazioni che devono essere eseguite quando l'utente interagisce con tali componenti.

Nell'applicativo sviluppato sono state definite due activity, che ospitano i vari fragment:

1. **AuthActivity**
Gestisce l'intero flusso di accesso all'applicazione, gestendo i fragment relativi al login, alla registrazione, alla validazione della mail e alla prima configurazione del profilo utente.
2. **MainButtonMenuActivity**
Si tratta della principale activity dell'applicazione. Ospita i fragment atti alla visualizzazione del profilo, la creazione di nuovi eventi, la visualizzazione di quelli sottoscritti e la visualizzazione di quelli pubblicati dagli altri utenti.

I fragment sono "collegati" tra loro mediante componenti di **navigation**, come suggerito dalla documentazione Android. I vari fragment vengono descritti singolarmente nella sezione "Come utilizzare l'applicativo".

In alcuni fragment è stata implementata anche la gestione dei **Safe Args**. In particolare sono stati utilizzati in tutti quei casi in cui si è rivelato necessario gestire il passaggio di dati da un fragment all'altro.

Infine per gestire i passaggi di schermata all'interno della MainActivity è stato utilizzato il componente che permette di costruire un **Menu** tramite un bottom navigation.

Model

Nell'applicativo vengono modellati alcuni tipi di dato fondamentali:

- **User**
Rappresenta un utente per l'applicazione.
- **Event**
La classe definisce come è strutturato un evento nel sistema.
- **EventLocation**
Permette di modellare la posizione di un oggetto Event in termini di latitudine e longitudine.
- **Weather**
Tale classe permette di gestire il meteo che caratterizza un oggetto Event sulla base della sua latitudine e longitudine (posizione).
- **Category**
Rappresenta una categoria che può essere associata ad un evento. L'utente sceglie una serie di categorie in fase di registrazione, che prendono il nome di "interessi".

Oltre a queste ultime sono state definite delle classi di tipo **Response**, che permettono di gestire il risultato delle operazioni di recupero dati dalle varie API e dal database locale. Tale risultato viene mappato nei rispettivi Model, quindi queste classi costituiscono delle classi di "supporto" per semplificare le varie operazioni.

Infine è stata definita una classe astratta chiamata **Result**, che rappresenta un risultato di un'operazione compiuta dal repository. I vari Result vengono inseriti nei livedata, quindi gli observer dichiarati osservano principalmente oggetti di tipo Result. Tale classe sfrutta il meccanismo delle inner-class per definire una serie di risultati.

*Ad esempio: **UserSuccess** è una inner-class definita all'interno della classe **Result**. Essa rappresenta il risultato delle operazioni di basso livello che vanno a buon fine (success) e che restituiscono un utente.*

Al suo interno viene definita anche una inner-class **Error**, che viene invece utilizzata per gestire gli errori prodotti dalle operazioni del repository.

Adapter

Sono delle classi che vengono utilizzate per definire come devono essere strutturati i componenti grafici con layout ripetuti, come i RecyclerView, le ListView e gli Spinner.

Viewmodel

Contengono metodi per interagire con i repository e restituiscono LiveData che possono essere osservati nei vari fragment per produrre operazioni a livello UI. Essi rispettano il principio di coerenza dei dati ed interagiscono con il repository per recuperare o memorizzare dati e non effettuano mai modifiche che rimangono locali. Di conseguenza viene mantenuta coerenza tra i dati presenti sulle Source of Truth e quelli gestiti dall'app.

L'app utilizza i seguenti Viewmodel:

- **CategoryViewModel**
Ha la responsabilità di mantenere i dati relativi alle categorie. Permette di ottenere le categorie, le immagini ed impostare le categorie a cui l'utente è interessato.
- **UserViewModel**
Serve per mantenere i dati relativi all'utente. Inoltre contiene anche i metodi di controllo per i dati inseriti dall'utente nelle varie schermate (ad esempio l'email, la password, ecc).
- **EventViewModel**
Ha la funzione di mantenere le informazioni relative agli eventi.
- **InterestsViewModel**
Mantiene i dati relativi agli interessi dell'utente che ha effettuato l'accesso.

Oltre a questi viewmodel sono stati definiti, se richiesti nel contesto, anche le relative classi **Factory**, che estendono la classe *ViewModelProvider.Factory*. Tali classi sono necessarie per poter istanziare dei viewmodel fornendo dei parametri al suo costruttore.

La classe **CategoriesHolder** è un Singleton che viene utilizzato nella schermata di selezione degli interessi e la sua responsabilità è quella di mantenere una lista contenente tutti gli interessi che sono stati selezionati.

Repository

I repository sono delle classi che permettono di interfacciarsi alle fonti di dati. Le classi che dipendono dai repository “visualizzano” i dati come una collezione. I repository definiscono operazioni CRUD (Create-Read-Update-Delete) per interagire con tale collezione di dati. L'interfaccia utente (Fragment / Activity e Viewmodel) non può dipendere direttamente dalle DataSource, per cui è necessario utilizzare tale sistema.

Sono stati utilizzati i seguenti repository (uno per ciascun tipo di dato):

- **CategoryRepository**
È il repository che permette di accedere alle fonti di dati relative alle categorie.
- **UserRepository**
Gestisce le operazioni di accesso ai dati legati all'utente, tra cui le informazioni acquisite in fase di registrazione, gli interessi e gli eventi a cui partecipa.
- **EventRepository**
Permette di gestire le operazioni di accesso ai dati degli eventi.
- **WeatherRepository**
La responsabilità di questa classe è quella di gestire i dati relativi al meteo e quindi si interfaccia alle API del meteo.
- **MapRepository**
Si occupa di interfacciarsi alle API della mappa per ottenere informazioni riguardo al luogo in cui si svolgono eventi.

Per rendere l'applicativo più scalabile, i repository sono implementati tramite un'interfaccia. Nelle interfacce vengono definiti i metodi che devono essere implementati dal repository.

Service

La classe **WeatherApiService** contiene la query Retrofit per effettuare la chiamata alle API che gestiscono il meteo.

DataSource

Sono classi che contengono i riferimenti alle effettive fonti di dati, che possono essere remoti o locali. Eseguono operazioni per i repository e restituiscono dati tramite callback. L'idea generale è che la callback effettua l'operazione di `postValue()` dei dati ottenuti dalla fonte di dati, in modo tale che gli observer, che osservano i LiveData restituiti dai Viewmodel, possano essere notificati.

Sono state utilizzate le seguenti classi di DataSource:

- **UserRemoteDataSource**
Si interfaccia ai servizi di Firebase Database e Firebase Storage e gestisce la fonte di dati per gli utenti in remoto.
- **UserLocalDataSource**
Permette di accedere ai servizi di Room per memorizzare e prelevare localmente informazioni sull'utente.
- **AuthenticationDataSource**
Il suo obiettivo è quello di fornire un sistema di accesso per il servizio di Firebase Authentication.
- **CategoryRemoteDataSource**
Contiene un riferimento a Firebase Database e Firebase Storage per gestire le informazioni legate alle categorie.
- **InterestRemoteDataSource**
Serve per accedere alle informazioni legate agli interessi dell'utente tramite un riferimento al Firebase Database.
- **InterestLocalDataSource**
La sua responsabilità è quella di fornire un sistema di accesso al database locale Room per interagire con i dati che rappresentano gli interessi dell'utente.
- **EventRemoteDataSource**
Questo data source serve a interfacciare il sistema rispetto a Firebase Database e Firebase Storage e permette di gestire i dati degli eventi in remoto.

- **ParticipationRemoteDataSource**

La responsabilità di questo componente è quella di gestire i dati relativi alle partecipazioni di un utente ad un evento tramite il riferimento a Firebase Database.

- **MapRemoteDataSource**

Questo data source rappresenta il collegamento tra l'applicativo e le API che ci permettono di interagire con i dati riguardanti la mappa.

- **WeatherRemoteDataSource**

Si tratta, come nel caso precedente, di un data source che permette di interfacciare il sistema ai servizi messi a disposizione dalle API che riguardano il meteo.

Database

Per salvare i dati localmente, si è deciso di utilizzare il servizio consigliato dalla documentazione Android, ossia Room. Sono stati salvati localmente le tipologie di dati più usati all'interno dell'applicativo, in modo tale da semplificare le operazioni di accesso ai dati in termini di velocità. Per questo motivo, seguendo quanto descritto è stato implementato:

- **LocalRoomDatabase**

È la classe astratta che rappresenta il database locale. Si tratta di una classe singleton, perché deve esistere un solo database locale nel sistema.

- **InterestDao**

L'interfaccia permette di modellare la tabella SQL all'interno di Room relativa agli interessi. Contiene inoltre i metodi per interagire con tali dati.

- **UserDao**

Room viene utilizzato per memorizzare informazioni relative all'utente. Questo sistema viene utilizzato ad esempio in fase di autenticazione e registrazione per controllare la sessione.

Util

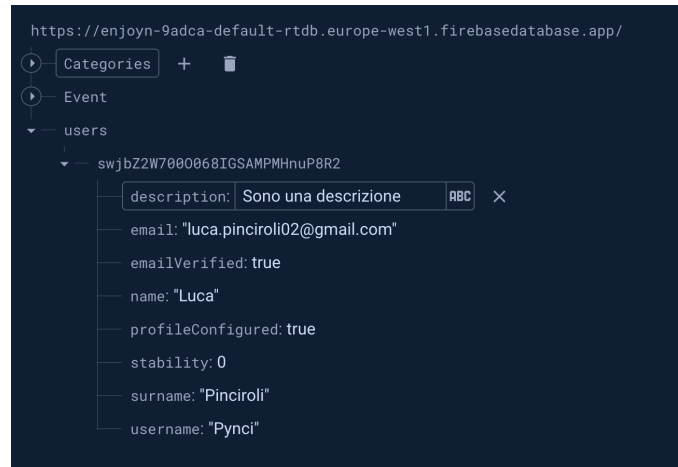
Questo package contiene tutte le classi di supporto per il sistema; ad esempio la classe **Constants**, che permette di definire una serie di costanti da utilizzare nel sistema. La classe principale è **ServiceLocator**, che viene utilizzata per restituire oggetti relativi ai vari servizi utilizzati nell'app, come ad esempio i riferimenti al database ed al repository.

Servizi esterni

L'applicativo utilizza i seguenti servizi esterni per funzionare correttamente:

- **[Firebase Database](#)**

È un database noSQL, che viene utilizzato per memorizzare dati relativi all'applicazione. Si tratta di un file JSON in cui ciascuna tabella SQL è rappresentata invece da un nodo. I records della tabella sono anch'essi dei nodi definiti da una chiave primaria ed un oggetto come valore.



- **[Firebase Authentication](#)**

È un servizio di firebase, che permette di gestire le operazioni di registrazione ed autenticazione. Permette di:

- tenere traccia degli utenti che si sono registrati all'applicazione.
- inviare email di validazione.
- cifrare automaticamente la password dell'utente.
- ripristinare la password dell'utente.

- **[Firebase Storage](#)**

Permette di memorizzare file ed immagini in un sistema strutturato a directory. Questo sistema viene utilizzato per memorizzare, ad esempio, le immagini relative alle categorie e le immagini profilo dei vari utenti.

- **[Retrofit](#)**

È il sistema utilizzato per effettuare le varie chiamate alle API. L'applicazione utilizza due API:

- a. **[Meteo \(Open meteo\)](#)**

Permette di ottenere informazioni riguardanti il meteo in uno specifico luogo, specificando le coordinate, dalla data corrente ai successivi 16 giorni.

- b. **[Mappa \(Mapbox\)](#)**

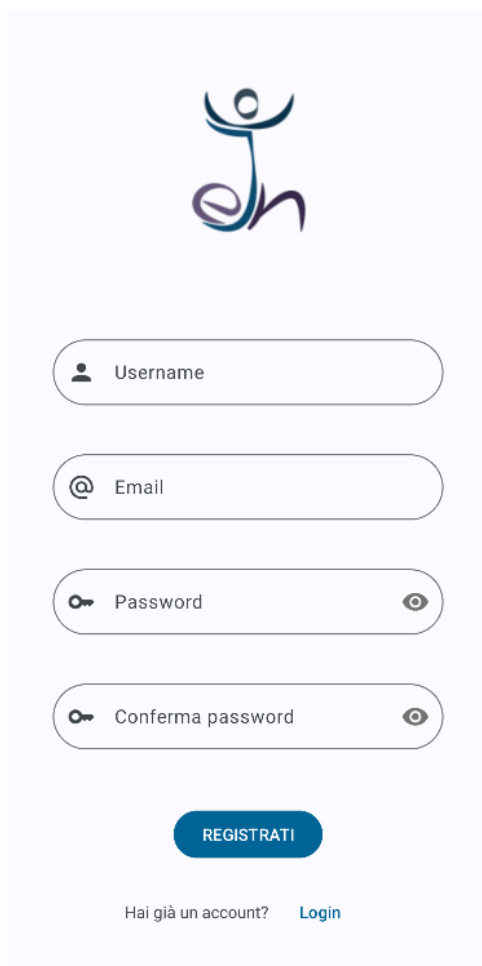
Queste API sono utilizzate per fare in modo che l'utente possa visualizzare su una mappa dove si trovano i vari eventi che gli vengono proposti.

Come utilizzare l'applicazione

In questa sezione vengono descritte le varie schermate che compongono l'applicazione e come esse sono in relazione l'una con l'altra.

Registrazione

La prima cosa che deve fare un utente che non ha mai utilizzato l'applicativo è la procedura di **registrazione**.



Nella schermata di registrazione l'utente inserirà i dati richiesti: username, email e password. La logica sottostante controlla dinamicamente che i valori inseriti dall'utente siano accettabili. Ad esempio viene fatto un controllo per verificare che l'utente non abbia inserito una mail che già esiste nel sistema. In caso tale controllo risulti positivo, viene visualizzato a schermo il relativo messaggio di errore e la pressione del pulsante "Registrati" non produce alcun risultato.

Se tutti i dati inseriti dall'utente sono accettabili, la pressione del tasto "Registrati" effettua la **creazione dell'account** (non ancora validato).

Successivamente viene inviata un'e-mail all'utente, in modo tale che possa validare l'indirizzo di posta elettronica che ha fornito.

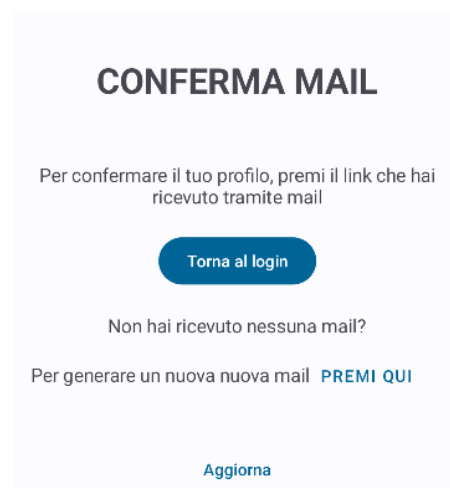
Infine, l'utente viene indirizzato verso una nuova schermata, che contiene semplicemente un messaggio indicante l'invio dell'e-mail di validazione.

Conferma e-mail

L'utente riceve una **e-mail di validazione** sul proprio indirizzo di posta elettronica.

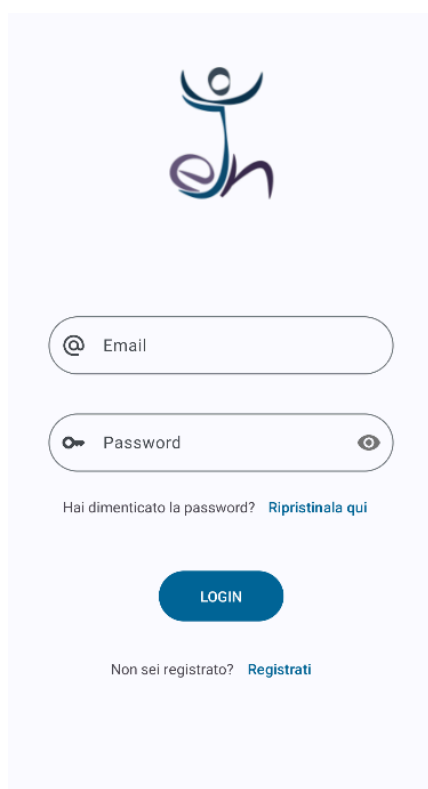
Per effettuare la validazione è sufficiente premere sul link contenuto in tale e-mail. In tal modo è possibile accedere all'app.

Nel caso in cui l'e-mail sia andata perduta, l'utente può generarne una nuova premendo sull'apposito pulsante.



Login

L'utente deve inserire le credenziali con le quali si è registrato e premere sul pulsante "Login". A questo punto, possono verificarsi quattro situazioni differenti:



1. **L'utente non fornisce le credenziali di accesso corrette.** In questo caso viene semplicemente visualizzato un messaggio di errore.

2. **L'utente non ha effettuato la validazione della mail.** In questo caso viene indirizzato sulla schermata contenente il messaggio di invio dell'e-mail di validazione sul proprio indirizzo di posta elettronica specificato (Conferma e-mail).

3. **L'utente ha effettuato la validazione della propria e-mail ed è la prima volta che accede all'applicativo.** In questo caso verrà mostrata la schermata per configurare il proprio profilo (opzionale).

4. **L'utente ha già effettuato la configurazione del profilo.** In questo caso viene indirizzato alla **schermata principale** dell'app.

Configurazione del profilo

La schermata di configurazione del profilo serve per aggiungere **informazioni personali facoltative** che caratterizzano il proprio account: nome, cognome, una foto-profilo ed una descrizione.

Poiché tali informazioni non sono rilevanti per il corretto funzionamento del sistema, tale sezione (ed anche quella successiva) è totalmente opzionale. Infatti è possibile inserire anche solo determinate informazioni, ad esempio solo la foto-profilo.

Scelta delle categorie

L'utente può scegliere delle **categorie di eventi preferite**, in modo tale che, all'apertura della schermata principale, possa visualizzare principalmente eventi correlati a queste ultime.

Dato che un utente può anche non avere alcun tipo di preferenza, anche in questo caso, la sezione è opzionale.

Recupero della password

Se l'utente ha **perso le proprie credenziali di accesso**, può accedere a tale schermata per effettuare la procedura di **ripristino della password**. Verrà inviata un'email per ripristinare la password all'indirizzo di posta elettronica fornito.

Corpo dell'applicativo

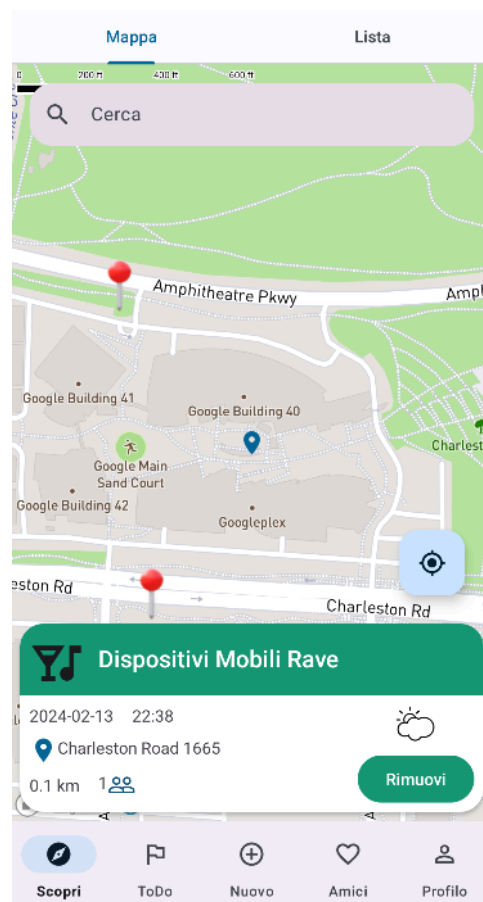
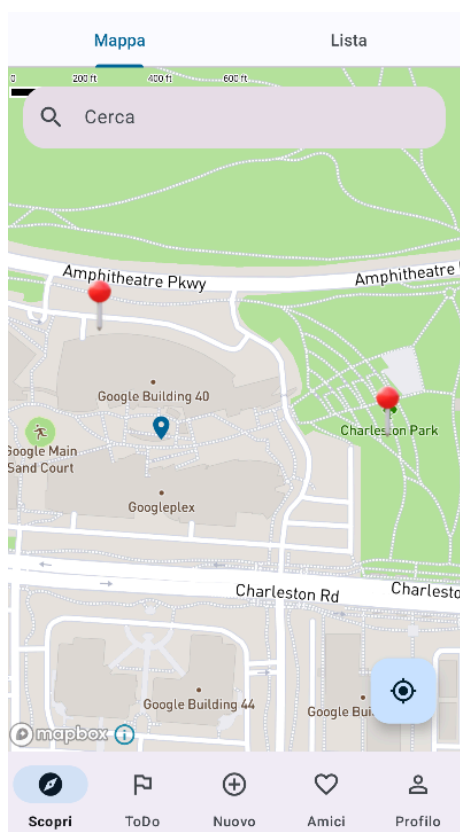
La schermata principale è stata implementata tramite un **menu**, che permette di effettuare la navigazione tra i vari fragment, come consigliato dalla documentazione Android. Il menu è stato strutturato in **cinque** schermate differenti, che vengono descritte nelle sezioni successive.

Scopri - mappa

La schermata “**Scopri**” permette di visualizzare in due modi differenti la lista degli eventi che sono stati registrati nel sistema. Il primo metodo di visualizzazione è attraverso la mappa: la posizione degli eventi viene rappresentata su una mappa tramite un segnaposto, come quello rappresentato nella **figura a destra**.



Il sistema permette di cliccare su un segnaposto per visualizzare i dettagli dell'evento che è stato programmato in quel luogo.



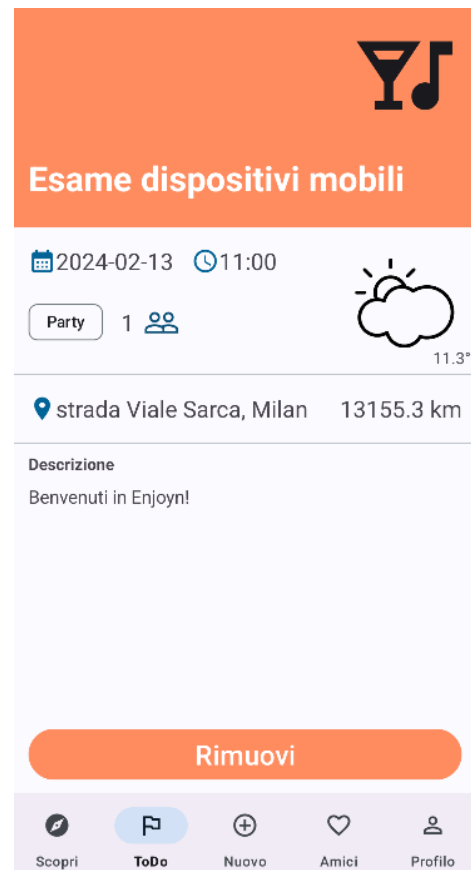
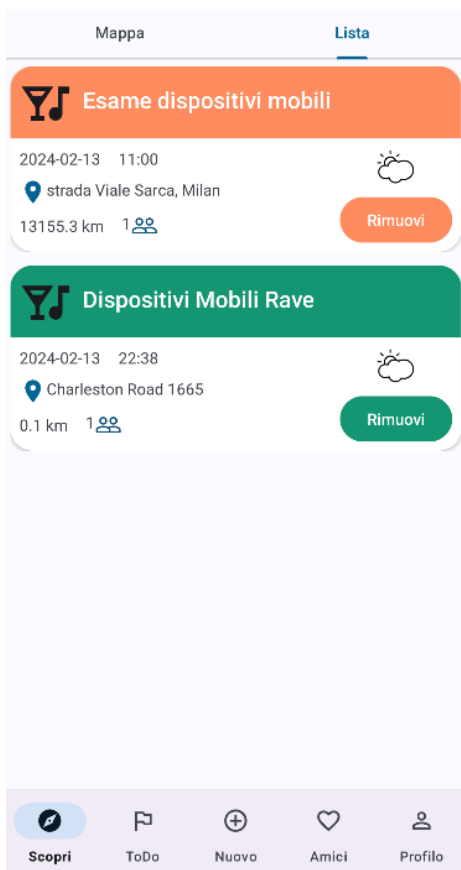
Scopri - Lista degli eventi

Il secondo sistema di visualizzazione degli eventi consiste in una lista, che permette di scoprire gli eventi registrati nel sistema da parte degli altri utenti. In particolare l'utente visualizza gli eventi filtrati secondo **gli interessi che ha registrato**.

Cliccando su un evento è possibile visualizzare l'evento in un formato più dettagliato, come rappresentato nelle immagini sottostanti.

Un evento è caratterizzato da data, orario, categoria, numero di partecipanti e luogo in cui è stato organizzato.

Inoltre, cliccando sul luogo, è possibile ottenere il percorso, dalla propria posizione, sino a quella in cui si svolgerà l'evento, attraverso un collegamento con Google Maps.



Tramite la pressione del tasto **“Unisciti”** l'utente si registra, tramite il proprio profilo personale, all'evento selezionato. L'utente ha la possibilità di visualizzare gli eventi a cui si è registrato nella sezione **ToDo**, descritta successivamente.

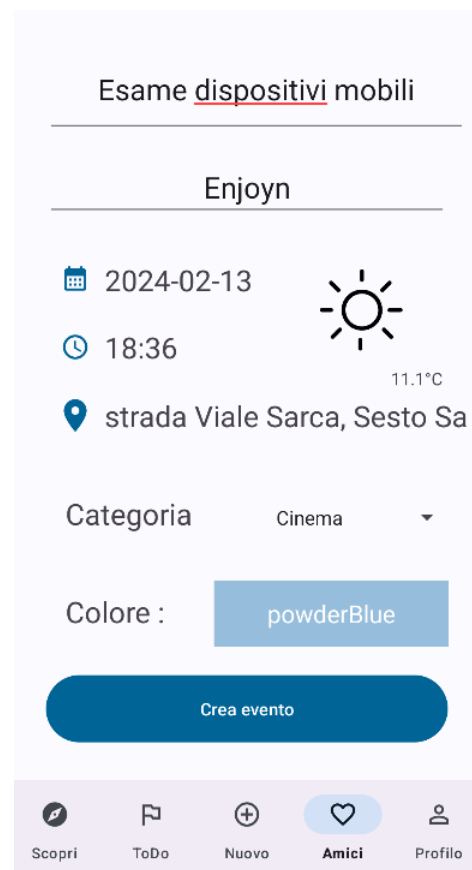
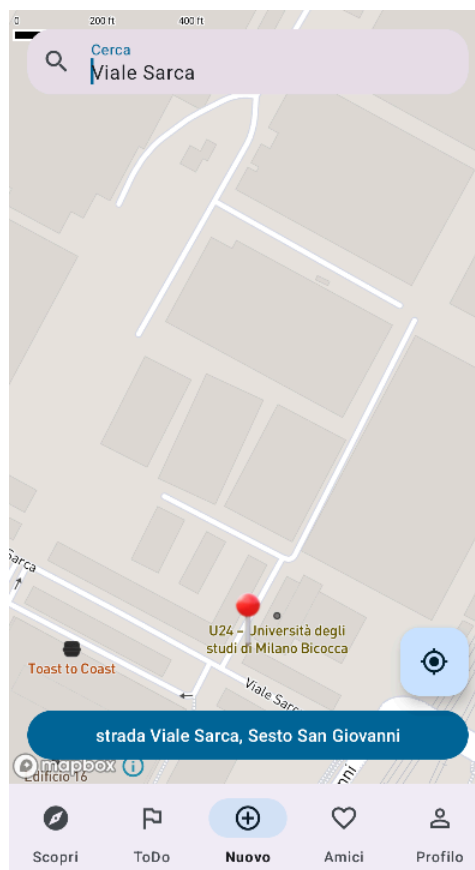
Infine, attraverso l'utilizzo, la gestione e l'interazione con le API riguardanti il meteo, l'utente è in grado di visualizzare un simbolo, che rappresenta le **previsioni meteo** nel giorno e nell'ora indicati.

Todo

La schermata “Todo” permette di visualizzare la lista degli eventi a cui l’utente partecipa ed eventualmente di cancellare la propria partecipazione all’evento.

Nuovo evento

La schermata “**Nuovo evento**” permette all’utente, tramite il proprio profilo personale, di creare un nuovo evento e di pubblicarlo nel sistema, in modo tale da renderlo visibile agli altri utenti. L’utente deve scegliere, come prima cosa, il luogo dove si svolgerà l’evento tramite la mappa che viene visualizzata schermo. Successivamente l’utente deve inserire i dettagli che caratterizzano l’evento. Inoltre può scegliere un colore con il quale l’evento verrà visualizzato dagli altri utenti nella schermata “Scopri”.



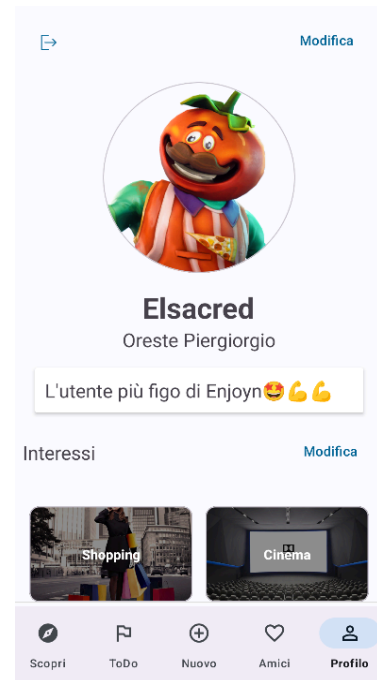
Utilizzando le API del meteo viene fatto in modo che anche l’utente possa visualizzare il meteo nel giorno ed orario stabilito entro 16 giorni dalla data corrente.

Profilo

La schermata “**Profilo**” permette all’utente di visualizzare il proprio profilo personale e, se necessario, può modificare alcuni dei dati inseriti in fase di registrazione, tra cui:

- il nome
- il cognome
- la foto profilo
- la descrizione
- gli interessi

Infine l’utente ha la possibilità di effettuare il Logout. Il risultato di tale operazione consiste nel distruggere la sessione corrente e riportare l’utente sulla pagina di Login.



Sviluppi futuri per l'applicativo

Quest'ultima sezione descrive gli sviluppi futuri per l'applicativo. Pensiamo che l'applicazione sviluppata si possa prestare facilmente ad estensioni e nuove funzionalità:

- **Account premium**

Vorremmo implementare un sistema che permette ad un utente di estendere il proprio profilo con un account premium e certificato. In questo modo un utente avrebbe la possibilità di mettere in rilievo i propri eventi nella schermata Discover. Questo metodo gioverebbe a coloro che, ad esempio, dispongono di un'attività e vorrebbero pubblicizzare gli eventi che organizzano.

- **Miglioramento dell'aspetto grafico**

Abbiamo cercato di definire fin da subito uno stile che desse rilievo all'applicativo. Siamo sufficientemente soddisfatti del risultato, ma pensiamo che si possano applicare delle modifiche che rendano più interessante l'applicazione dal punto di vista visivo.

- **Creazione di una lista amici**

L'utente potrebbe vedere la lista dei partecipanti ad un evento ed aggiungerli alla lista propria lista amici.

- **Eventi privati**

Un utente avrebbe la possibilità di organizzare degli eventi visibili solo ed esclusivamente ai propri amici.

- **Chat di gruppo nell'evento**

Implementare una chat di gruppo che permette agli utenti che partecipano ad un determinato evento di interagire tra di loro.

- **Creazione di un account Admin**

Al momento la gestione delle categorie (ad esempio l'aggiunta e la rimozione) è disponibile solo dal terminale di firebase. Un account Admin dovrebbe avere la possibilità di interagire globalmente con l'applicativo tramite permessi speciali.