

# Moai Planner

## Documentazione



**Elio Gargiulo - 869184**

**Stefano Rigato - 869441**

**Andrea Mocellin - 869218**

**Francesco Lamanna - 869052**

**Nicolle Isavo Benavides Milla - 875386**

# Sommario

Introduzione.....	3
Funzionalità Principali.....	3
Architettura dell'Applicazione.....	4
API utilizzate.....	4
Servizi utilizzati.....	4
Activity e Fragment.....	5
1 - Main.....	5
2 - Account.....	6
3 - Onboarding.....	6
ViewModel.....	6
Utils.....	7
Adapter.....	7
Data e Repository.....	7
Design.....	8
Material 3.....	8
Main Activity.....	8
HomeFragment.....	9
FileFragment.....	10
ToDoListFragment.....	11
TomatoFragment.....	13
NoteFragment.....	14
OptionsFragment.....	16
Welcome Activity.....	17
WelcomeFragment.....	17
RegisterFragment.....	18
SigninFragment.....	19
OnboardingFragment.....	20
Sviluppi Futuri.....	21

# Introduzione

L'applicazione è nata con lo scopo di fornire un'alternativa semplice e intuitiva per la gestione di impegni e tutto ciò che riguarda l'organizzazione di appunti e sessioni di studio di uno studente universitario.

Al momento si è voluto mantenere tutto il più semplice possibile, per poi aggiornare ed estendere in seguito le funzionalità già presenti, fornendo comunque un'applicazione valida all'utilizzo.

La scelta del nome dell'applicazione è nata principalmente per scherzo, ma dopo aver effettuato ricerche sul significato della statua Moai, abbiamo effettivamente scoperto che il messaggio trasmesso (prosperità, determinazione) potesse essere appropriato, oltre ad essere abbastanza intrigante.

L'applicazione richiede almeno Android 8 (SDK 26) per funzionare ed è stata pensata per funzionare al meglio su Android 13 (SDK 33).

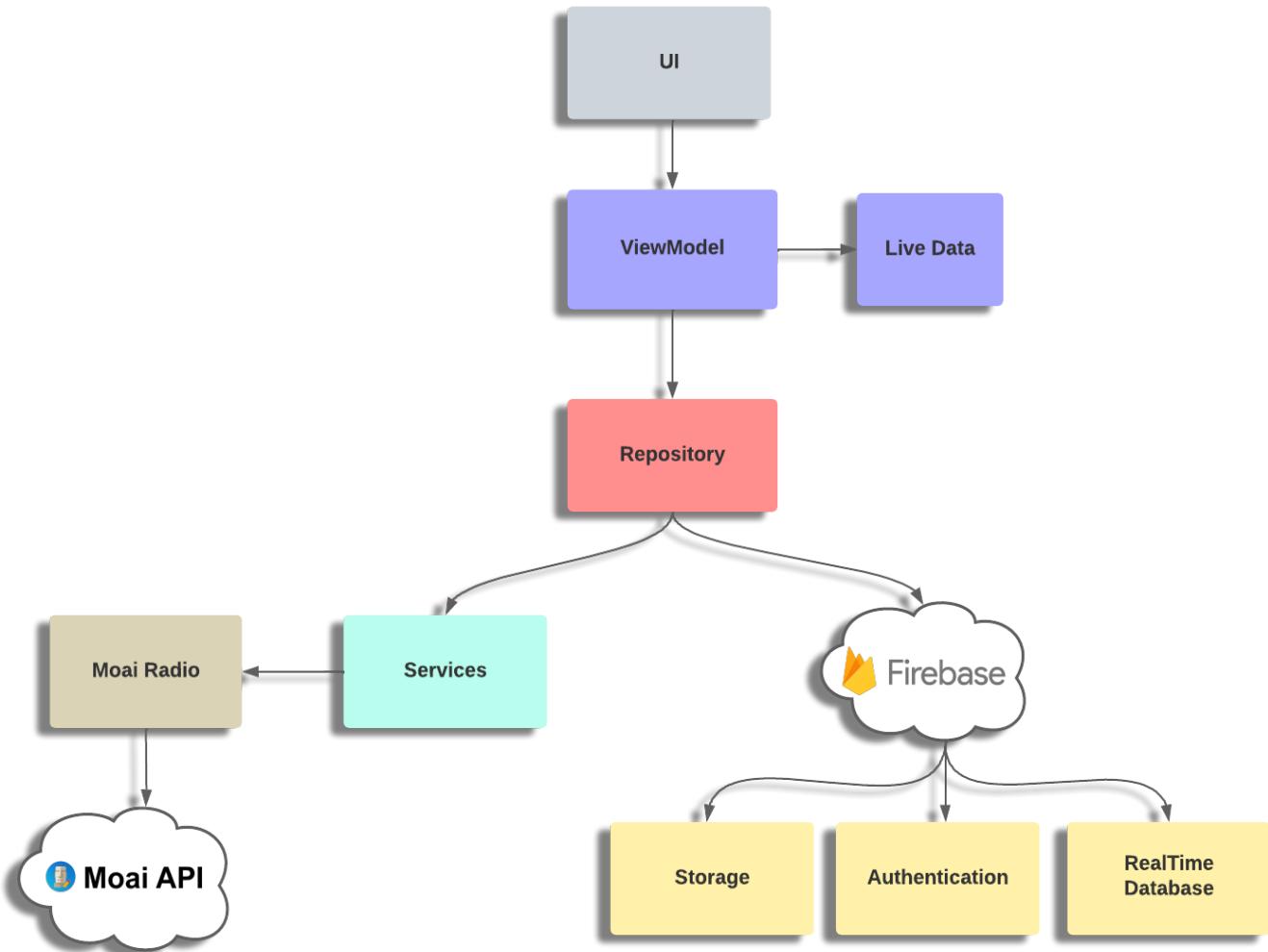
## Funzionalità Principali

Moai Planner offre all'utente le seguenti funzionalità principali:

- 🌶 **Gestione di appunti in formato Markdown**, un formato estremamente utile quando si tratta di prendere appunti. Esso permette di formattare il testo in vari modi, in modo da consentire una raccolta ordinata ed efficace delle proprie note.
- 🌶 **Gestione delle cose da fare**, organizzate tramite liste per giorni. L'utente è in grado di specificare un giorno qualunque e successivamente inserire, modificare o cancellare i propri impegni, con associato l'orario. Gli impegni del giorno verranno mostrati nella Home dell'applicazione.
- 🌶 **Gestione del proprio tempo di lavoro**, introducendo un timer ispirato alla Tecnica del Pomodoro, diviso in sessioni di lavoro e pausa per una quantità scelta di rounds.
- 🌶 **Sincronizzazione sul Cloud** delle Note e dei ToDo tramite un sistema di account proprietario o Google.
- 🌶 **Home** con accesso alle Note attraverso un File Manager, dove è possibile caricare note o creare note/cartelle. Vi è la possibilità di impostare le Note Preferite e di visualizzare solamente queste ultime. Nella Home inoltre sono presenti eventualmente gli impegni del giorno.
- 🌶 **Impostazioni** per modificare il tema delle applicazioni, sistema di notifiche e gestione generale del proprio account (nel caso di account Google vi è una restrizione delle alternative).

L'applicazione pertanto richiede la connessione ad Internet per la maggior parte delle sue funzionalità (in quanto tutto è sincronizzato e accessibile online fatta eccezione per il timer pomodoro e la scrittura di note in locale) e un account.

# Architettura dell'Applicazione



## API utilizzate

📻 **Moai API**: questa API viene utilizzata per ascoltare musica da una radio Lo-Fi che contiene canzoni royalty free. Abbiamo deciso di mettere questa funzionalità all'interno del Fragment del Pomodoro. Per utilizzarla abbiamo usato OkHttp.

## Servizi utilizzati

- 🌐 **Firebase Authentication**: utilizziamo questo servizio di Google per autenticare gli utenti nell'applicazione.
- 🌐 **Cloud Storage for Firebase**: questo servizio di Google ci permette invece di salvare i vari file come le note e gli avatar.
- 🌐 **Firebase Realtime Database**: questo servizio di Google fornisce un database NoSQL che permette di salvare e sincronizzare i dati degli utenti in tempo reale. Lo utilizziamo per salvare le varie note preferite ed i ToDo.
- 🌐 **MoaiRadioService**: questo servizio si occupa di gestire la Moai API e quindi il Music Player dell'applicazione. Esso permette la riproduzione musicale, fornendo un vero e proprio Music Player all'utente.

## Activity e Fragment

### 1 - Main

- 📎 **MainActivity:** è l'activity principale dell'applicazione. Si occupa di creare la toolbar per la navigazione e di gestire quest'ultima. Ogni Fragment principale viene caricato all'interno di questa activity.
- 📎 **NoteFragment:** questo fragment crea un TabLayout con due tab, uno di Edit ed uno di Preview che rispettivamente contengono l>EditFragment ed il PreviewFragment per la modifica e la visualizzazione delle note in markdown. Inoltre gestisce anche i vari salvataggi e aperture di note.
- 📎 **EditFragment:** si occupa della gestione del testo da scrivere delle note e comunica con il MarkdownViewModel.
- 📎 **PreviewFragment:** si occupa di mostrare il markdown formattato in una WebView.
- 📎 **HomeFragment:** questo è il fragment che viene mostrato appena apri l'applicazione, permette di accedere alle note o alle impostazioni e mostra anche i ToDo del giorno corrente.
- 📎 **FileFragment:** permette di vedere tutte le note e cartelle di note salvate sul database dell'utente. È possibile anche interagire con esse per aprire le note sul NoteFragment oppure eliminarle o creare nuove note e cartelle. Inoltre è possibile aggiungere tra i preferiti e filtrare per essi delle note.
- 📎 **OptionsFragment:** comunica con il SettingsViewModel per salvare le varie impostazioni modificabili dall'utente e la possibilità di gestire il proprio account (Come cambio email/password/username).
- 📎 **ToDoListFragment:** mostra un calendario interattivo che permette di selezionare una data dove è possibile aggiungere un ToDo con un testo. Quando viene aggiunto il ToDo è possibile interagire con esso attraverso la checkbox per segnare se il ToDo è stato svolto o meno, e attraverso lo swipe verso destra si può modificare il testo del ToDo, mentre con lo swipe verso sinistra si potrà eliminare il ToDo.
- 📎 **TomatoFragment:** mostra e gestisce il timer per il metodo di studio Pomodoro. Quando il timer della sessione scade, sia per il lavoro che per il riposo, viene inviata una notifica all'utente. È anche presente la Moai Radio per ascoltare musica LoFi durante lo studio.

## 2 - Account

- 📎 **WelcomeActivity:** questa activity viene mostrata quando si vuole dare accesso ad un account o crearne uno nuovo. Gestisce quindi tutta la parte che riguarda l'accesso dell'utente nell'applicazione.
- 📎 **RegisterFragment:** è il fragment che gestisce tutta la parte di registrazione richiedendo email password ed un username.
- 📎 **SigninFragment:** gestisce il login all'account chiedendo email e password.
- 📎 **WelcomeFragment:** questo è il fragment principale del WelcomeActivity, vengono mostrati all'utente due button per accedere all'account, uno con email/password classico mentre l'altro attraverso i GoogleServices.

## 3 - Onboarding

- 📎 **OnboardingFragment:** gestisce la preparazione dei vari fragment che presentano, in generale, le varie funzioni dell'applicazione all'utente. Questi fragment vengono mostrati la prima volta che l'utente apre l'app.
- 📎 **OnboardingFirstScreen:** mostra che l'applicazione può essere usata per prendere appunti su delle "Note".
- 📎 **OnboardingSecondScreen:** mostra che l'applicazione può essere usata per segnare i propri "ToDo".
- 📎 **OnboardingThirdScreen:** mostra che l'applicazione può essere usata come timer per il metodo di studio "Pomodoro".

## ViewModel

- ID** **MarkdownViewModel:** questo ViewModel si occupa di gestire tutti i dati necessari per mostrare un file Markdown e gestire i salvataggi.
- ID** **SettingsViewModel:** questo ViewModel si occupa di gestire tutte le impostazioni salvate dall'utente nel fragment OptionsFragment.
- ID** **SettingsViewModelFactory:** componente aggiuntivo necessario alla creazione del SettingsViewModel.
- ID** **TomatoViewModel:** questo ViewModel si occupa di gestire tutti i dati necessari al timer del Pomodoro come le varie durate delle sessioni di studio/pausa e l'aggiornamento del timer.

## Utils

- 🔧 **DisableableViewPager**: gestisce lo swipe del TabLayout presente in NoteFragment.
- 🔧 **FolderItem**: classe che rappresenta un FolderItem, utilizzato in FolderManager.
- 🔧 **NavigationHelper**: serve per semplificare la navigazione fra Fragment ed Activity.
- 🔧 **NetworkUtils**: utilizzato per controllare se l'applicazione è connessa ad internet, in caso contrario nei fragment che richiedono l'accesso ad internet, viene mostrato un dialog che notifica l'errore e riporta l'utente al NoteFragment.
- 🔧 **Quintuple**: classe utilizzata per compattare cinque dati in una quintupla.
- 🔧 **ToDoltem**: classe che rappresenta un ToDoltem, utilizzato in ToDoListFragment.
- 🔧 **ToDoltemListener**: interfaccia utilizzata per monitorare e quindi modificare o eliminare un ToDoltem.
- 🔧 **Utils**: classe che contiene varie funzioni utili all'applicazione, come il download dell'avatar dell'utente, il cambio di tema, snackbars.
- 🔧 **ViewPagerPage**: interfaccia utilizzata dall>EditPageAdapter.

## Adapter

- 📁 **CalendarAdapter**: classe adapter utilizzata per il calendario in ToDoListFragment.
- 📁 **EditPageAdapter**: classe adapter utilizzata per le note in NoteFragment.
- 📁 **FolderViewAdapter**: classe adapter utilizzata per gestire i FolderItem in FileFragment (Apertura, eliminazione).
- 📁 **OnboardingAdapter**: classe adapter utilizzato per gestire le schermate di introduzione all'applicazione.

## Data e Repository

- 💾 **CalendarData**: contiene dei metodi utili per rappresentare una data del calendario.
- 💾 **FolderManager**: gestisce tutta la parte di creazione/upload/eliminazione di note e cartelle.
- 💾 **SettingsRepository**: salva i dati dell'impostazioni utente nelle SharedPreferences.
- 💾 **ToDoFetcher**: prende dal Firebase Realtime Database i ToDo dell'utente.
- 💾 **GoogleSignInHelper**: utilizzato per accedere all'applicazione utilizzando i GoogleServices.
- 💾 **UserAuthentication**: utilizzato per accedere e registrare utenti attraverso Firebase Authentication con email e password.

# Design

Il design dell'applicazione si basa principalmente su una divisione dei compiti in diversi Fragment e due Activity principali per differenziare l'applicazione in fase di accesso e utilizzo vero e proprio.

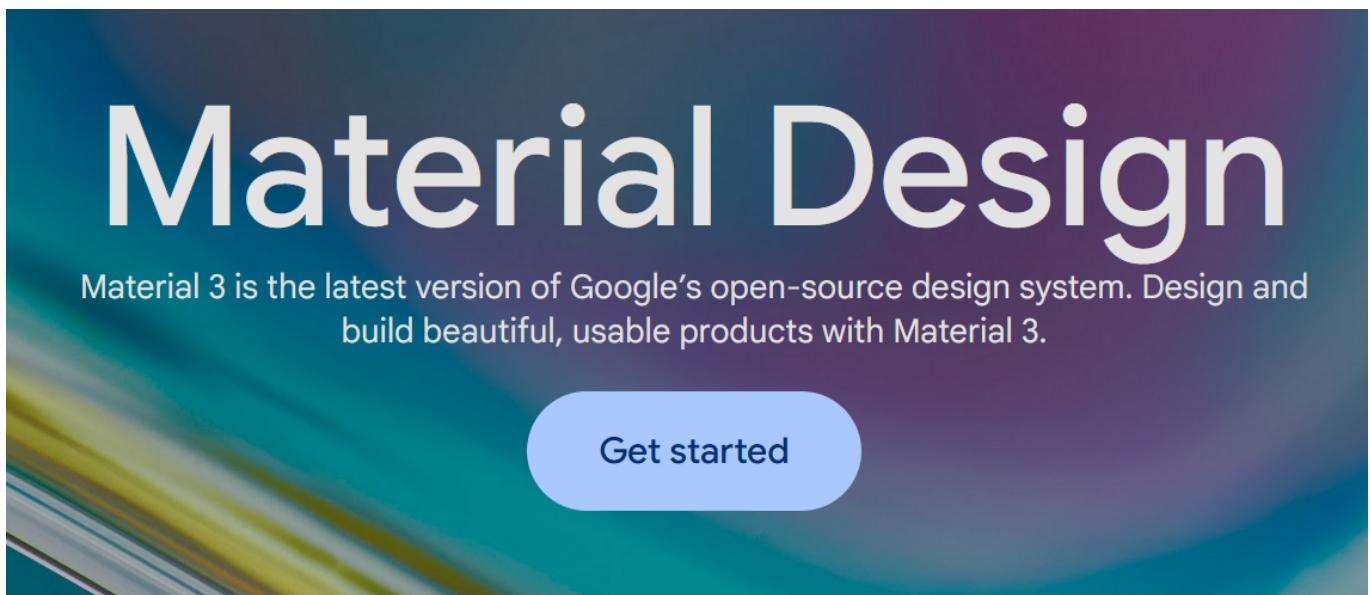
## Material 3

L'applicazione, per quanto riguarda l'interfaccia grafica, si basa su sistema di design [Material 3](#) offerto direttamente da Google.

Esso fornisce componenti grafici minimali, molto piacevoli e facili da utilizzare come Switch, EditText, Buttons, etc.

L'applicazione utilizza come colore principale l'azzurro, un colore che si accosta molto bene sia al nero (tema scuro), sia al bianco (tema chiaro).

I due temi utilizzano variazioni di nero per quanto riguarda il tema scuro, e variazioni di grigio e bianco per quanto riguarda il tema chiaro.

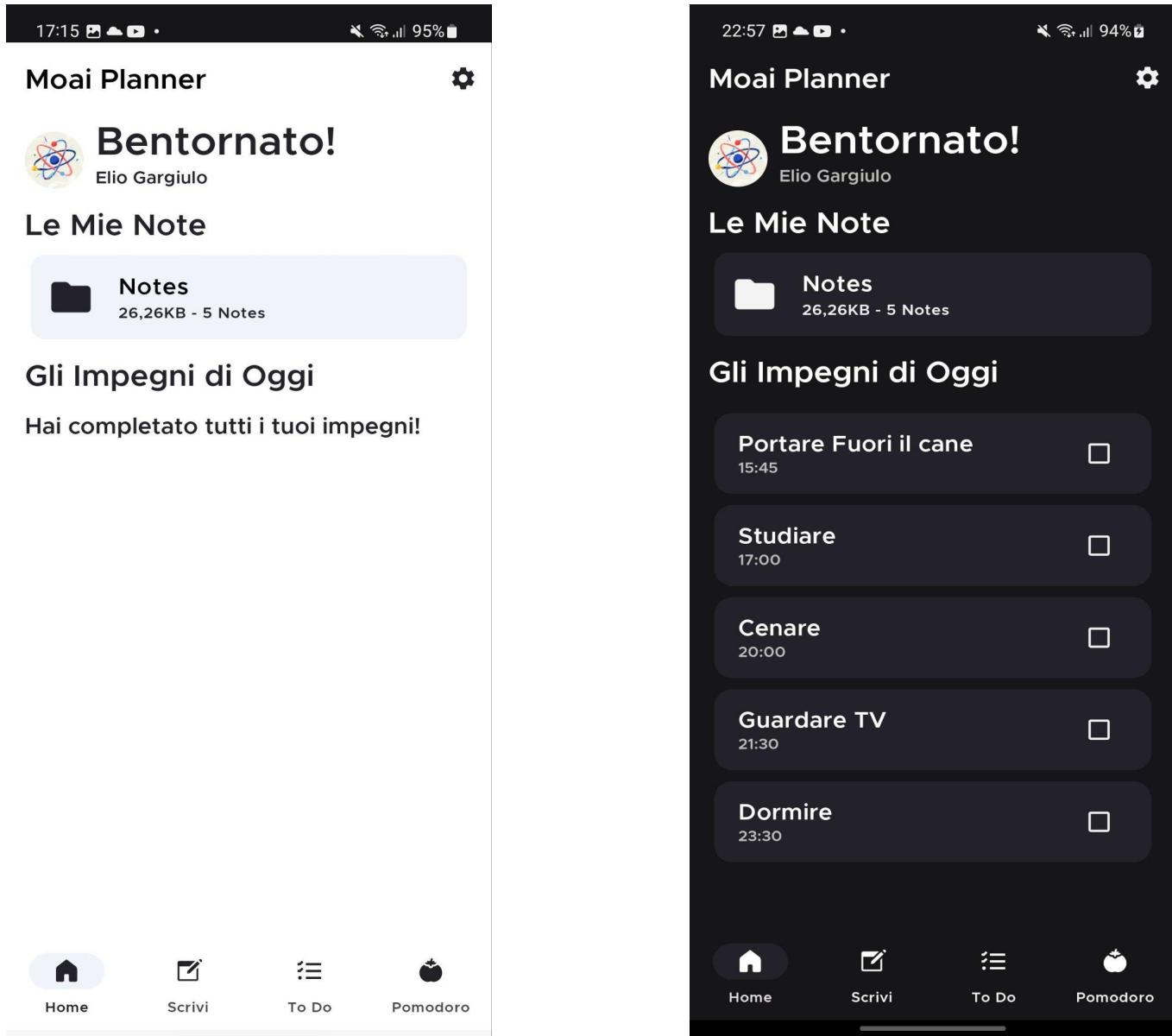


## Main Activity

La main activity è l'attività principale dell'applicazione.

Si occupa di fornire per ogni fragment al suo interno la barra di navigazione e la toolbar dove l'utente può accedere alle impostazioni dell'applicazione, o nel caso del NoteFragment, per gestire i file.

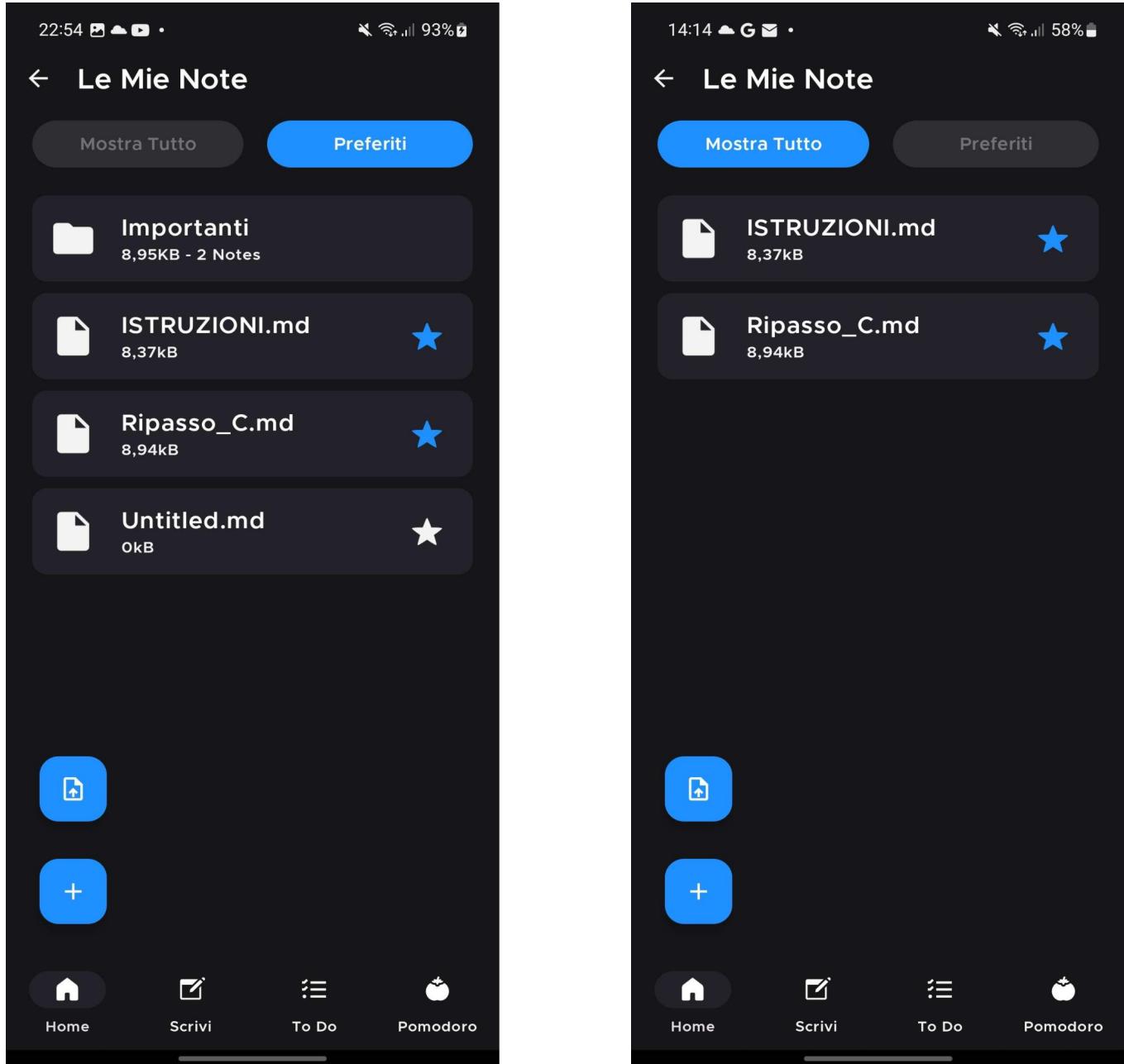
## HomeFragment



Questo fragment è la Home dell'applicazione ed è composta da due *RecyclerView*, una che si occupa tramite gli appositi Adapter di gestire la visualizzazione attraverso delle liste per le note ed una per i ToDo del giorno corrente.

Premendo su “ Notes” verrà aperto il *FileFragment*, che generalmente permette di vedere tutte le note salvate dall'utente sul cloud attraverso il File Manager. È possibile anche interagire con i ToDo del giorno corrente dove, attivando la checkbox, il ToDo verrà segnato come completato e rimosso dalla vista nella Home. L'interazione vera e propria avviene attraverso la comunicazione dei cambiamenti al RealTime Database. La lista dei ToDo è ordinata in modo crescente per l'ora specificata. Questo fragment presenta, come ogni altro fragment, la versione “Light” di seguito riportata. Il download dell'avatar viene effettuato tramite la libreria “Picasso”, con l'apposita funzione collocata nella classe *Utils*.

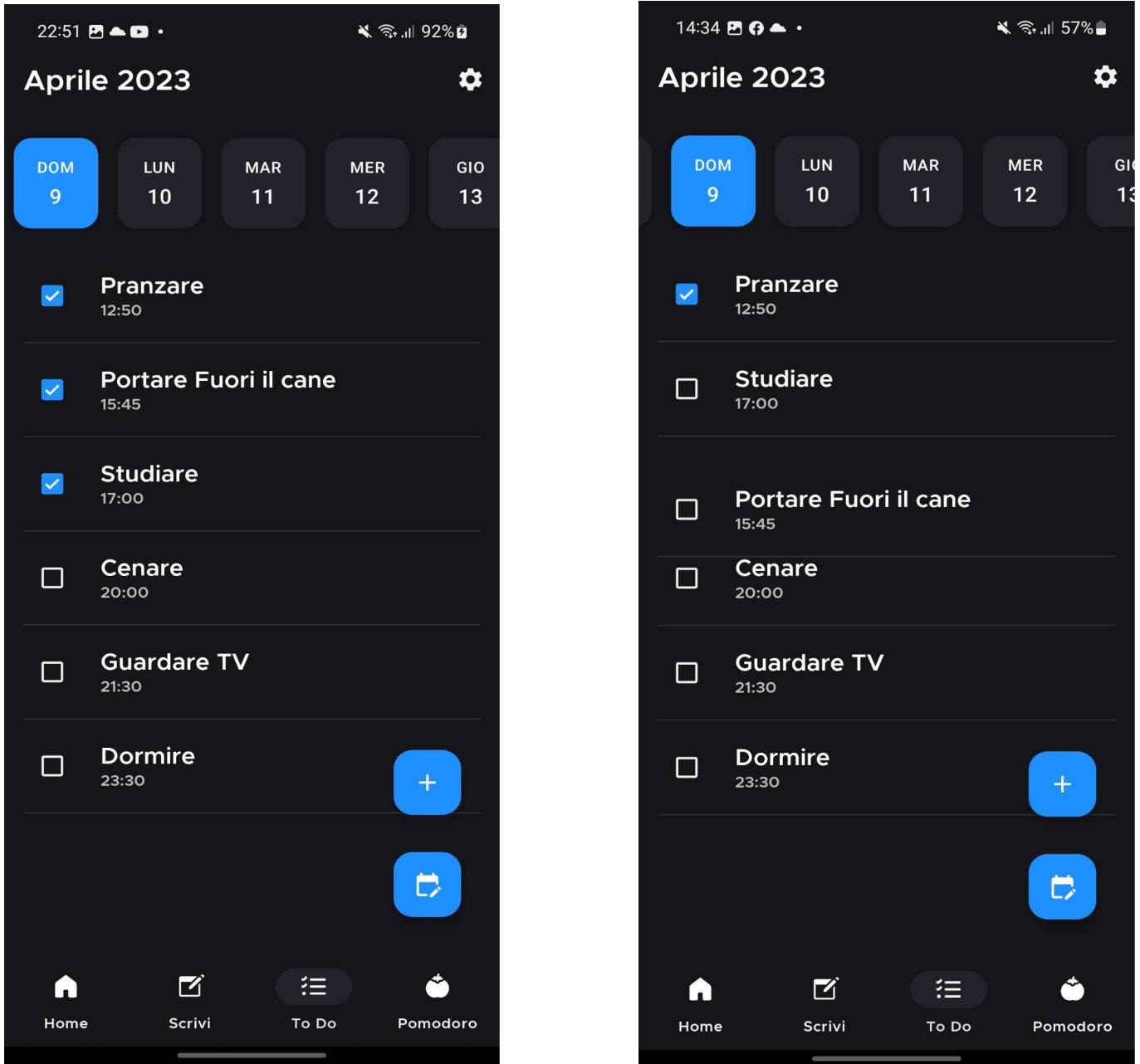
## FileFragment



Questo fragment utilizza un *RecyclerView* con relativo Adapter, ViewHolder e classe per descrivere un item della lista. Vengono mostrate tutte le cartelle e note salvate dall'utente sul cloud. Attraverso i due *floating action buttons* è possibile creare nuovi file o cartelle oppure caricare un file già esistente dalla memoria del dispositivo attraverso un specifico *dialog*. Inoltre è possibile tenere premuto su cartelle o note per aprire un *dialog* che permette di eliminare l'elemento selezionato.

Ogni file ha una stellina che può essere attivata premendoci sopra per aggiungere la nota ai preferiti e, grazie al bottone sopra il *RecyclerView*, è possibile cambiare dalla vista dei preferiti a quella di tutte le note. Il sistema di preferiti viene sincronizzato attraverso l'utilizzo del RealTime Database.

## ToDoListFragment



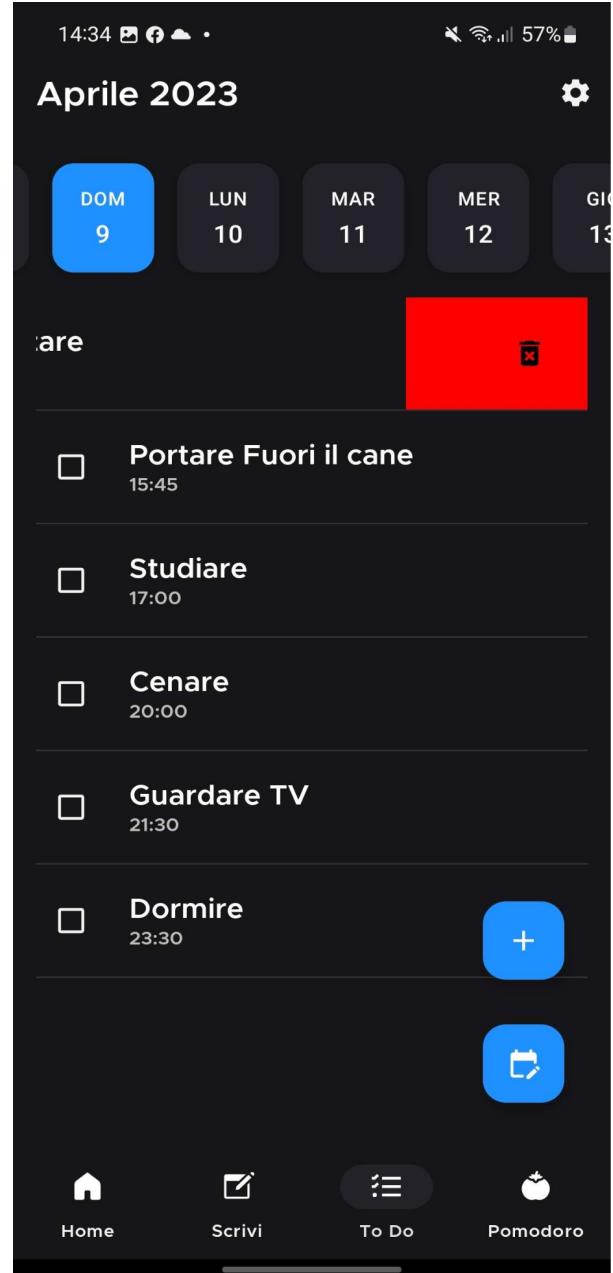
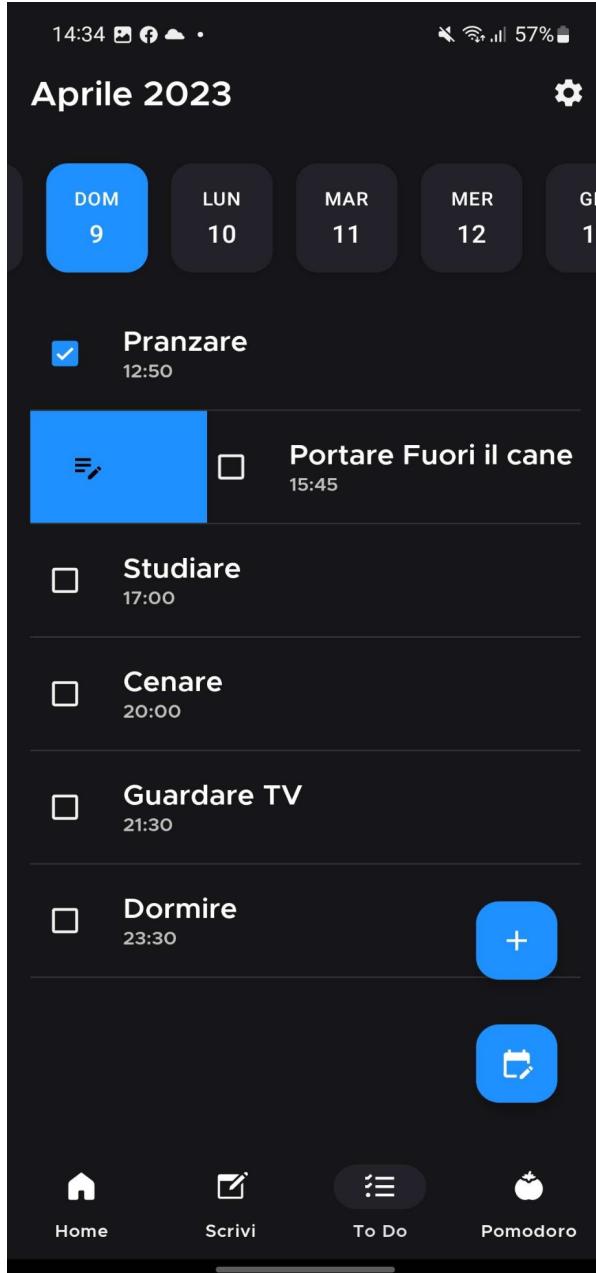
Il **ToDoListFragment** presenta due *RecyclerView* e relativi *Adapters*, *ViewHolders* e classi per descrivere gli item delle liste , tuttavia quello superiore è orizzontale e mostra il calendario del mese corrente.

È possibile cambiare la data premendo gli elementi (le date del calendario) di questo *RecyclerView* oppure è possibile cambiare data attraverso il *floating action button* 17, che attraverso un dialog permette all'utente di scegliere una data, un mese ed anche l'anno.

Per aggiungere un ToDo si utilizza il *floating action button* , anche questo aprirà un dialog dove l'utente potrà inserire una descrizione dell'impegno ed un orario.

Gli elementi della lista possiedono alcune proprietà, tra cui il supporto di alcune gestioni come swipe e hold.

Tenendo premuto un elemento della lista si potrà ordinare la sua posizione in quest'ultima.

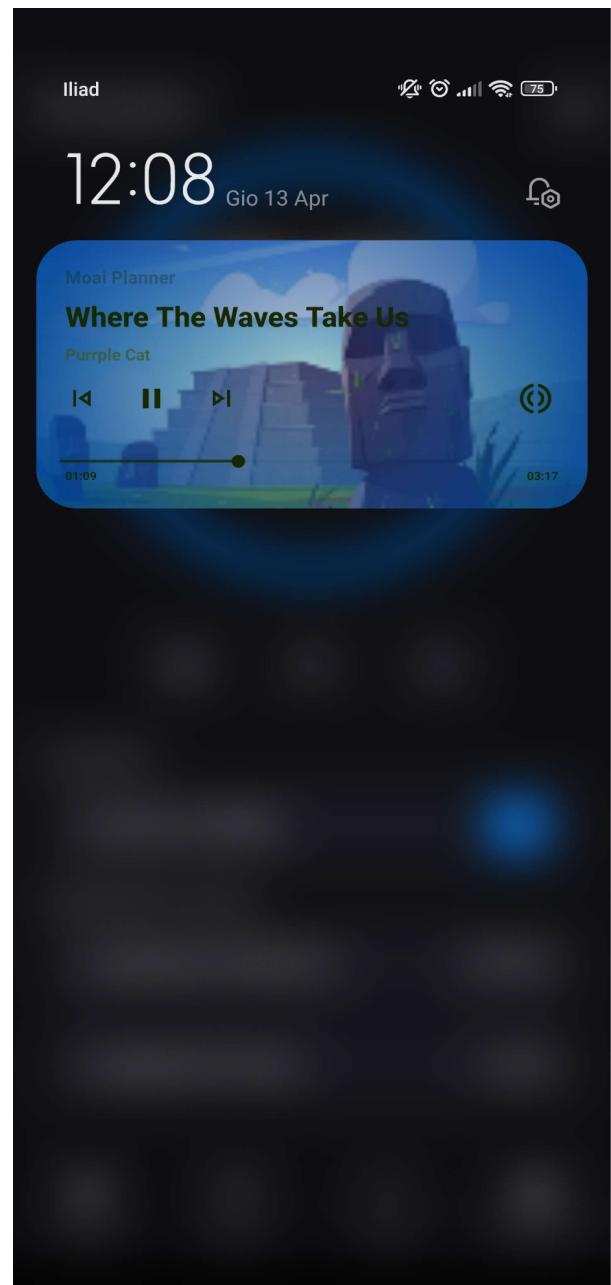
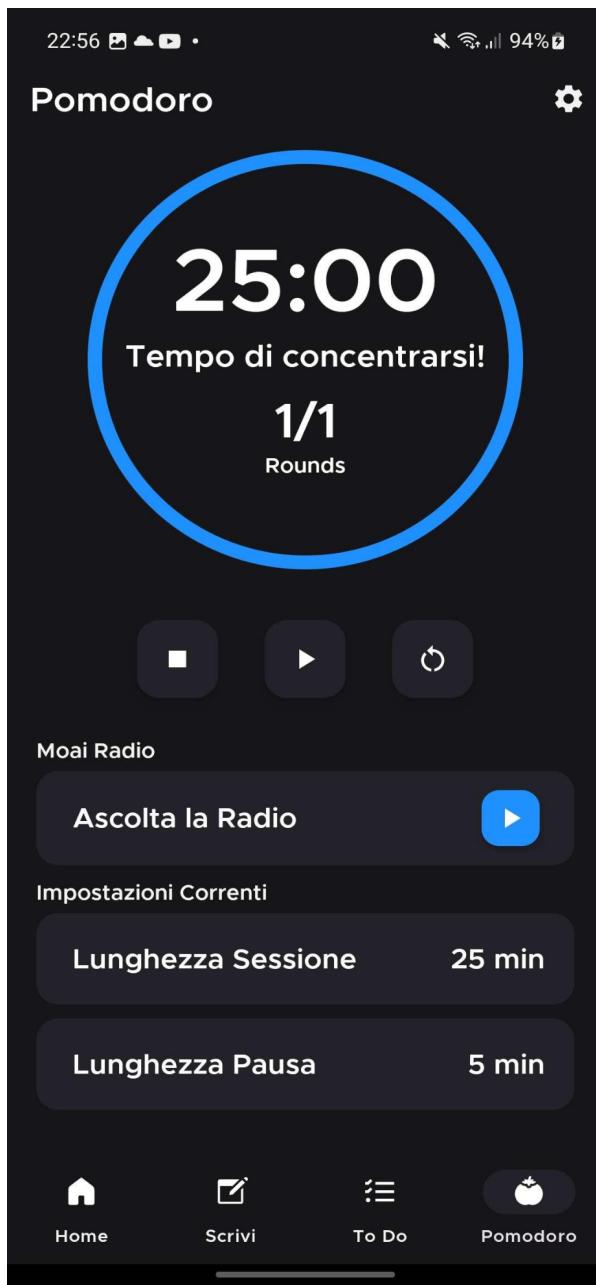


È possibile modificare il ToDo facendo swipe verso destra su di esso o anche eliminarlo facendo lo swipe verso sinistra.

Tutto viene quindi gestito attraverso un *ItemTouchHelper*.

Gli elementi della ToDoList sono automaticamente sincronizzati attraverso il RealTime Database, che ad ogni modifica viene aggiornato.

## TomatoFragment



N.B: Screenshot di notifica con music player su Android 13 con MIUI 14, lo stile della notifica con music player può variare a seconda delle varie skin utilizzate dai vendor dello smartphone.

La tecnica del Pomodoro è un metodo per la gestione del tempo nato alla fine degli anni ottanta. La tecnica utilizza un timer per suddividere il lavoro in intervalli tradizionalmente lunghi 25 minuti, separati da brevi pause. Il **TomatoFragment** si occupa quindi di mostrare un timer con la sessione corrente (Tempo di concentrarsi oppure pausa) e 3 bottoni per interagire con il timer. Per modificare i tempi della sessione è possibile andare nelle impostazioni attraverso il bottone .

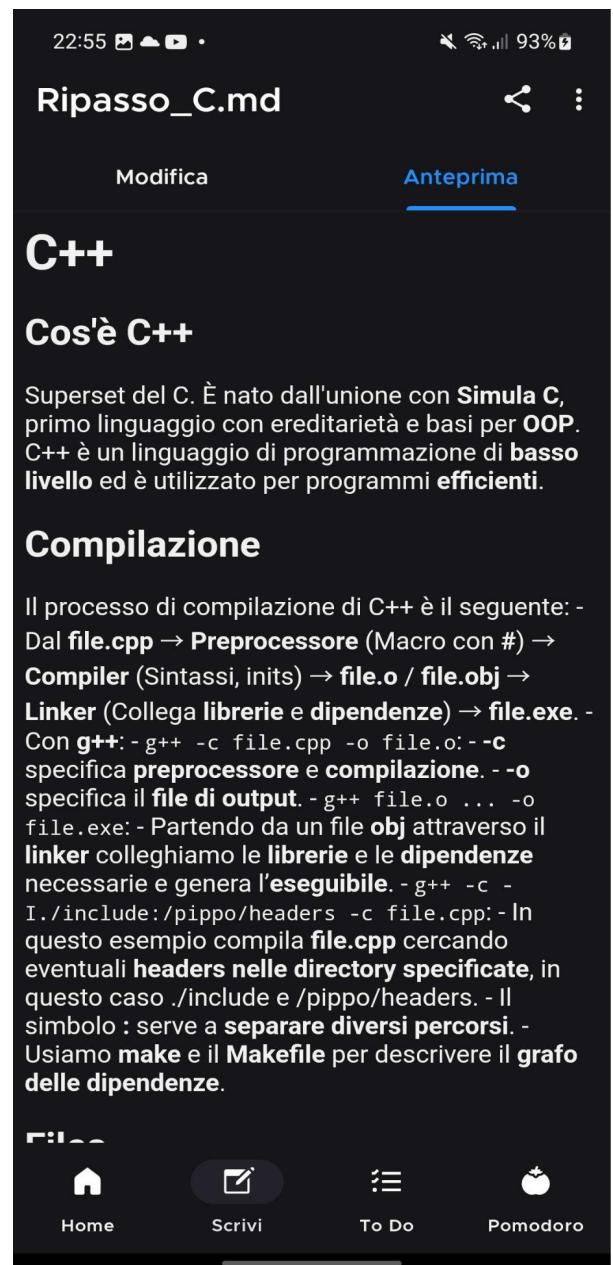
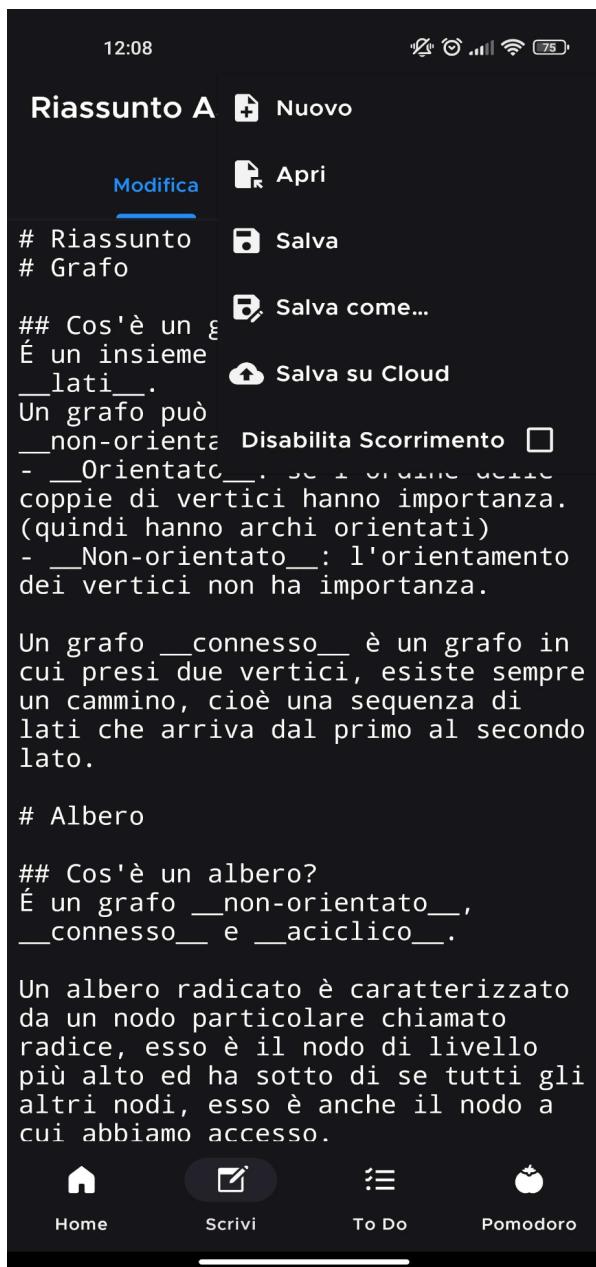
Il timer è composto da una *ProgressBar* circolare e gestito attraverso l'utilizzo del *TomatoViewModel*, che contiene informazioni utili al timer (come minuti, rounds, etc.)

È possibile anche ascoltare la "Moai Radio" premendo il bottone di play , la musica viene fornita attraverso la "Moai API" e componendo una playlist casuale di canzoni LoFi.

Attraverso la notifica è possibile interagire con la musica attraverso i bottoni di skip o pausa, ed è anche possibile "navigare" sulla traccia audio.

Per smettere di ascoltare la radio basterà premere sul bottone di stop  sotto il timer del Pomodoro.

## NoteFragment

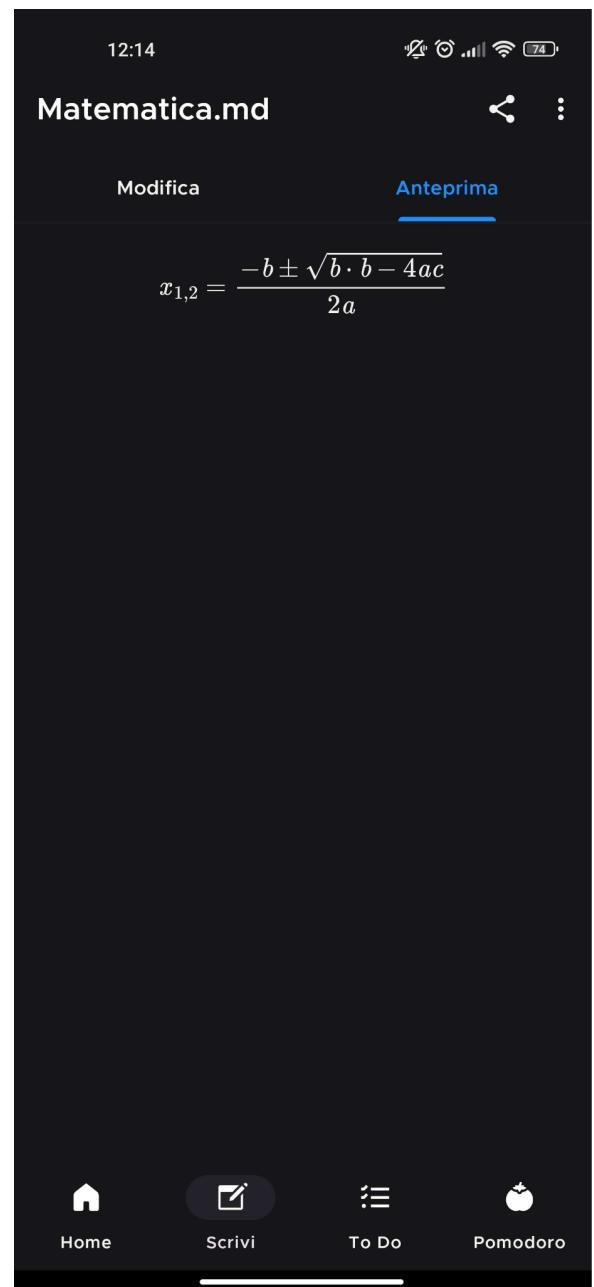
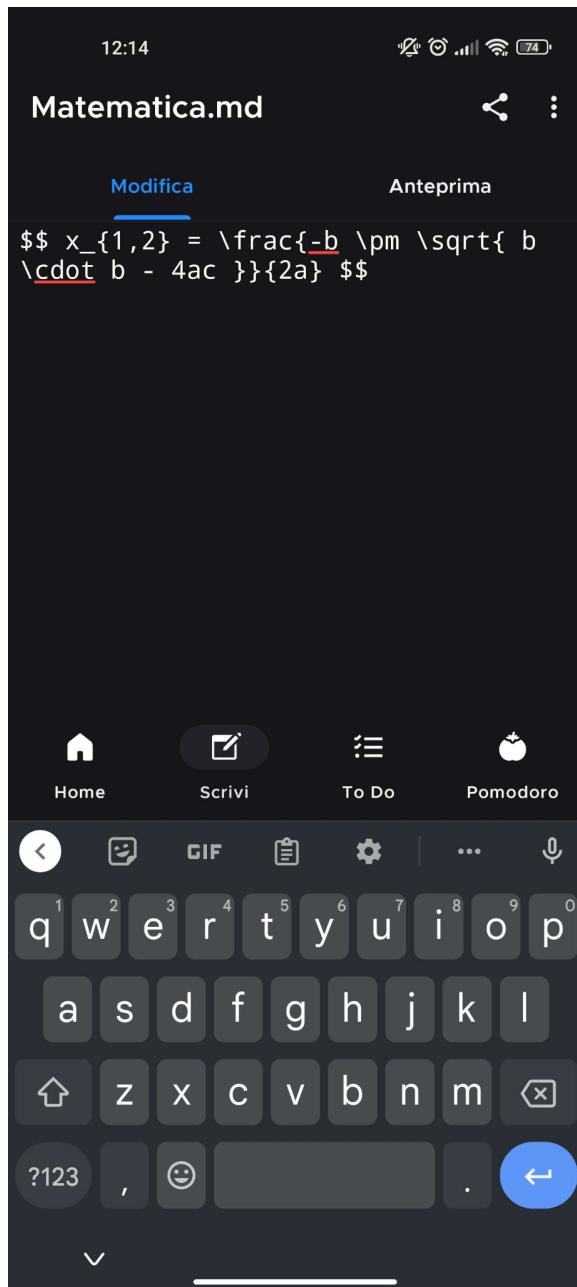


Per prendere note abbiamo scelto di utilizzare il linguaggio di markup "Markdown". Il fragment è formato da un *TabLayout* composto da due tab, uno di "Modifica" ed uno di "Anteprima".

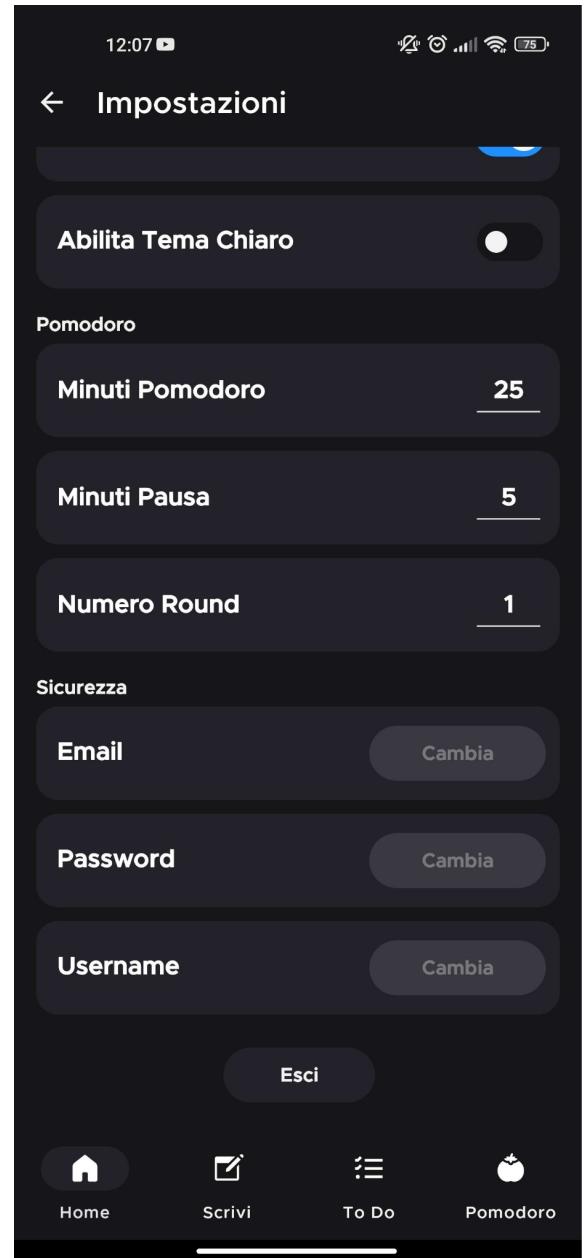
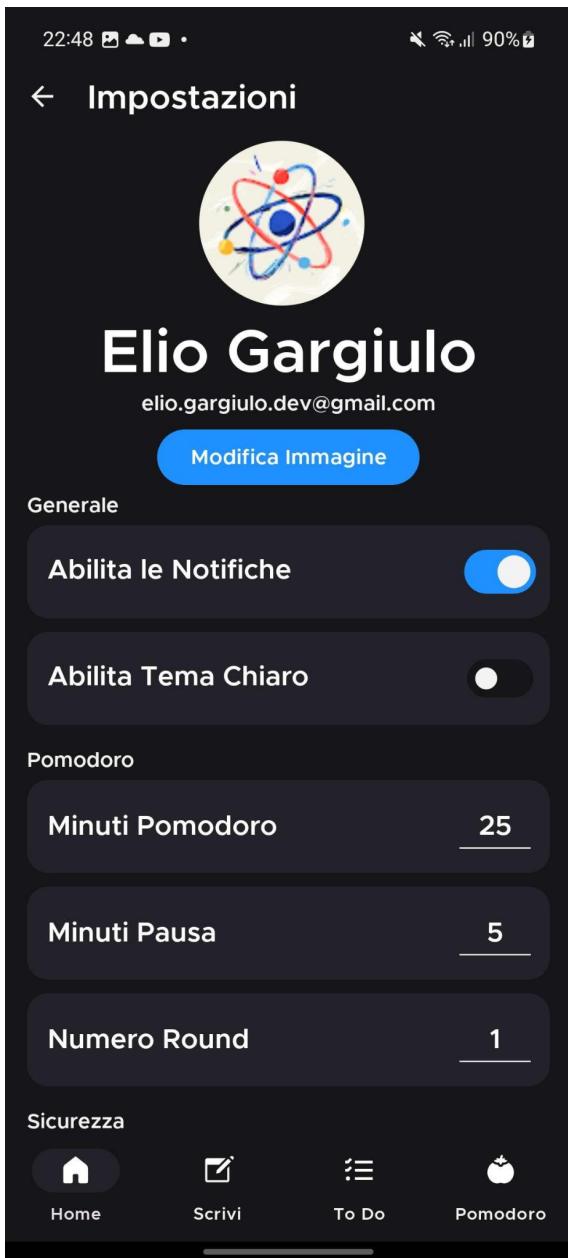
Nel tab di "Modifica", che carica l'**EditFragment**, è possibile scrivere in Markdown e successivamente visualizzare il risultato nel tab di "Anteprima", il quale carica il **PreviewFragment**. I fragment si occupano di comunicare con il *MarkdownViewModel* e relativi Adapter per gestire i file.

È stato anche aggiunto il supporto a LaTeX scrivendo all'interno di `$$<inserire LaTeX formula qui>$$`.

È anche possibile condividere il file di markdown attraverso il bottone condividi, oppure si possono visualizzare le varie opzioni per aprire file, crearli o salvarli attraverso i 3 puntini. Se viene abilitata l'opzione di "Disabilita Scorrimento" per cambiare da un tab all'altro bisognerà premere sul nome del tab e lo swipe per cambiare viene disabilitato.



## OptionsFragment



Il fragment utilizza una semplice ScrollView con vari Button, Switch ed EditText per cambiare le impostazioni.

Le impostazioni disponibili che possono essere cambiate sono le notifiche, il tema, i timer/round del Pomodoro e anche gestire il proprio account con la possibilità di cambiare l'immagine dell'utente, email, password o username. Il pulsante in fondo "Esci" permette di uscire dall'account corrente. Nel caso venga utilizzato un account Google (come nelle immagini riportate), le funzionalità di sicurezza saranno disabilitate in quanto gestite da Google.

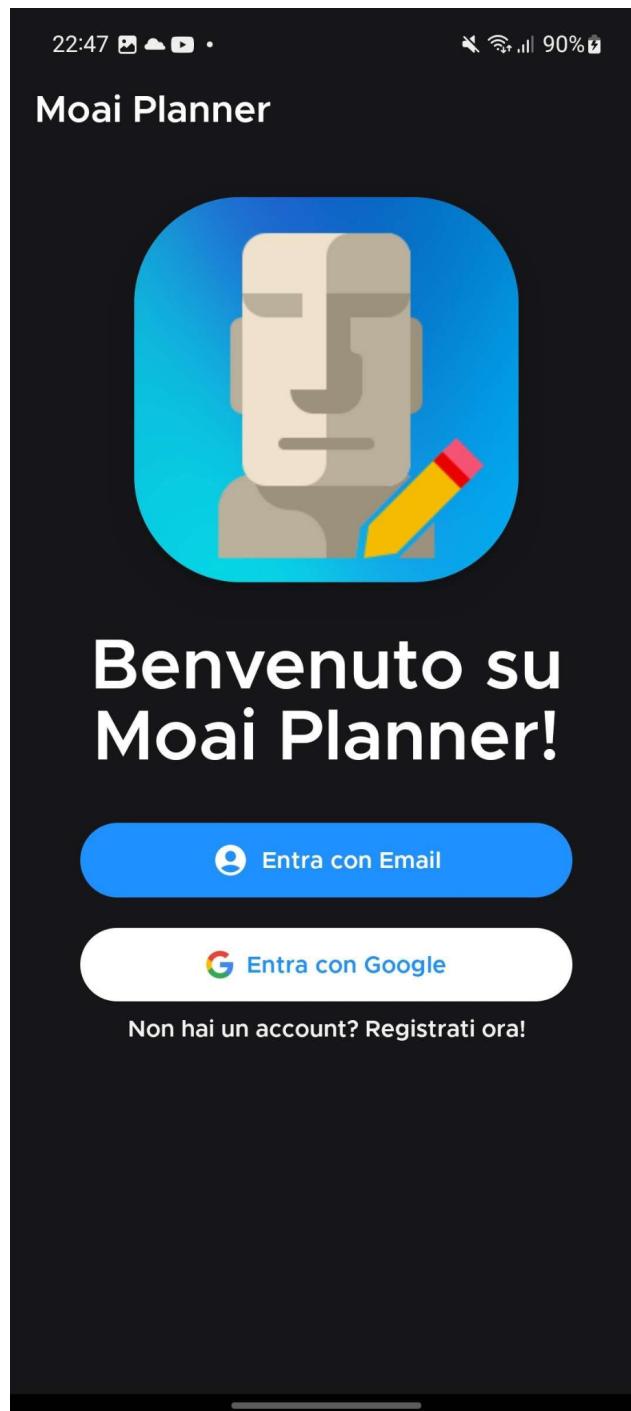
Il fragment comunica i cambiamenti delle impostazioni attraverso il *SettingsViewModel* e salvate tramite il *SettingsRepository*.

Il download delle immagini viene effettuato tramite la libreria "*Picasso*".

## Welcome Activity

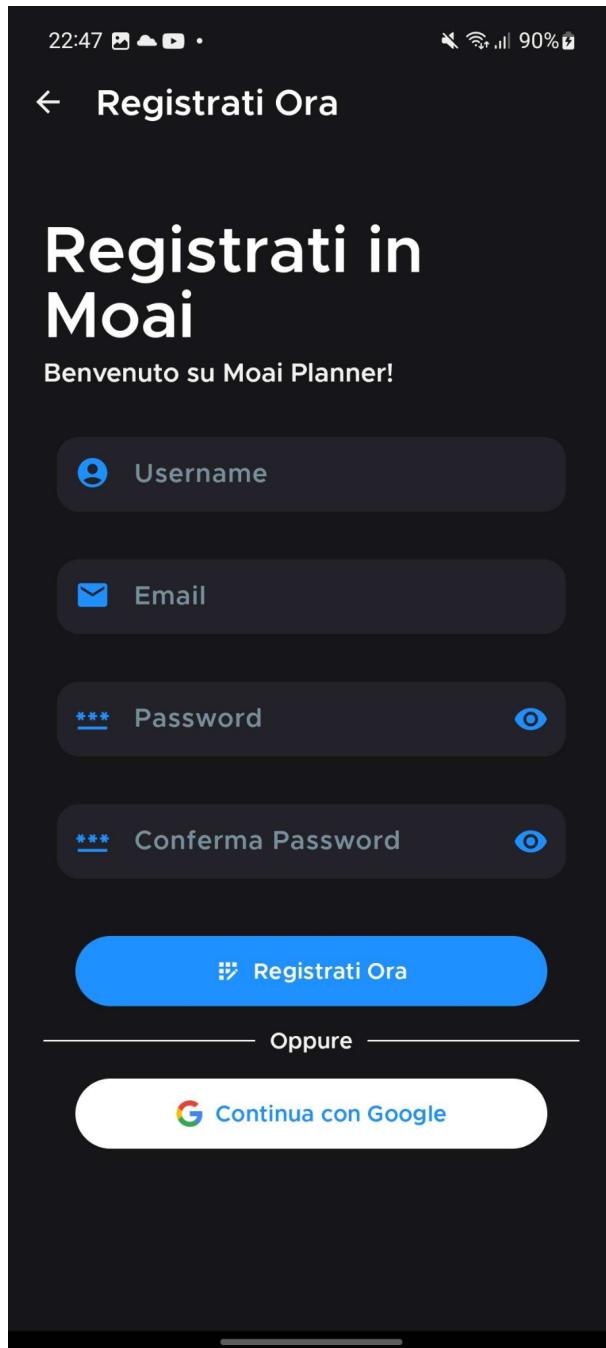
Questa activity si occupa di gestire la parte riguardante l'accesso da parte di un utente all'applicazione e di caricare le impostazioni della *SettingsRepository* dalle *SharedPreferences*.

## WelcomeFragment



Fragment con interfaccia molto semplice che fornisce due bottoni per permettere due modalità di login, ed un testo interattivo per passare alla schermata di registrazione. La registrazione tramite Email o Google viene effettuata tramite le classi *UserAuthentication* e *GoogleSignInHelper*

## RegisterFragment



Fragment che gestisce la registrazione di un utente nell'applicazione.

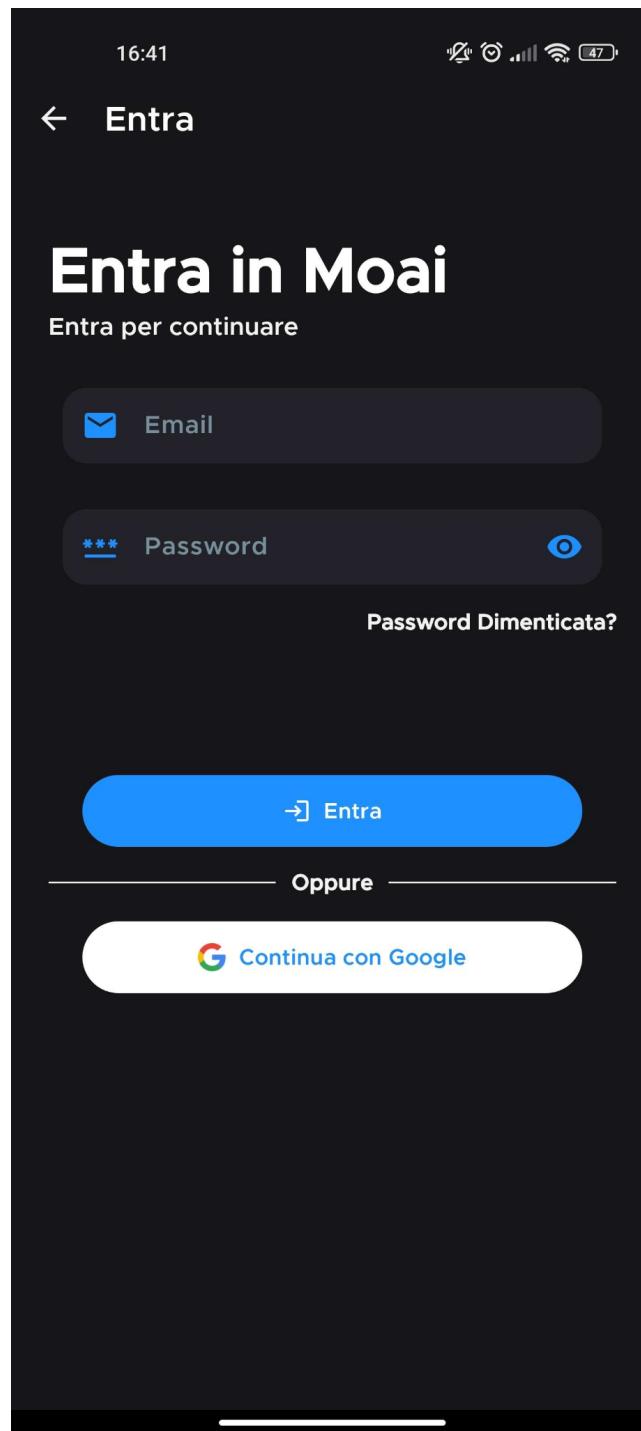
L'utente può inserire nei vari *EditText* i dati necessari per la registrazione.

Questi dati vengono validati e se tutto va a buon fine (ovvero l'utente attraverso i metodi forniti dalla classe *UserAuthentication* viene registrato su Firebase) verrà inviata una mail di verifica della registrazione.

Dopo aver confermato la registrazione è possibile eseguire il login/sign in.

In alternativa è possibile effettuare il login/registrazione con Google.

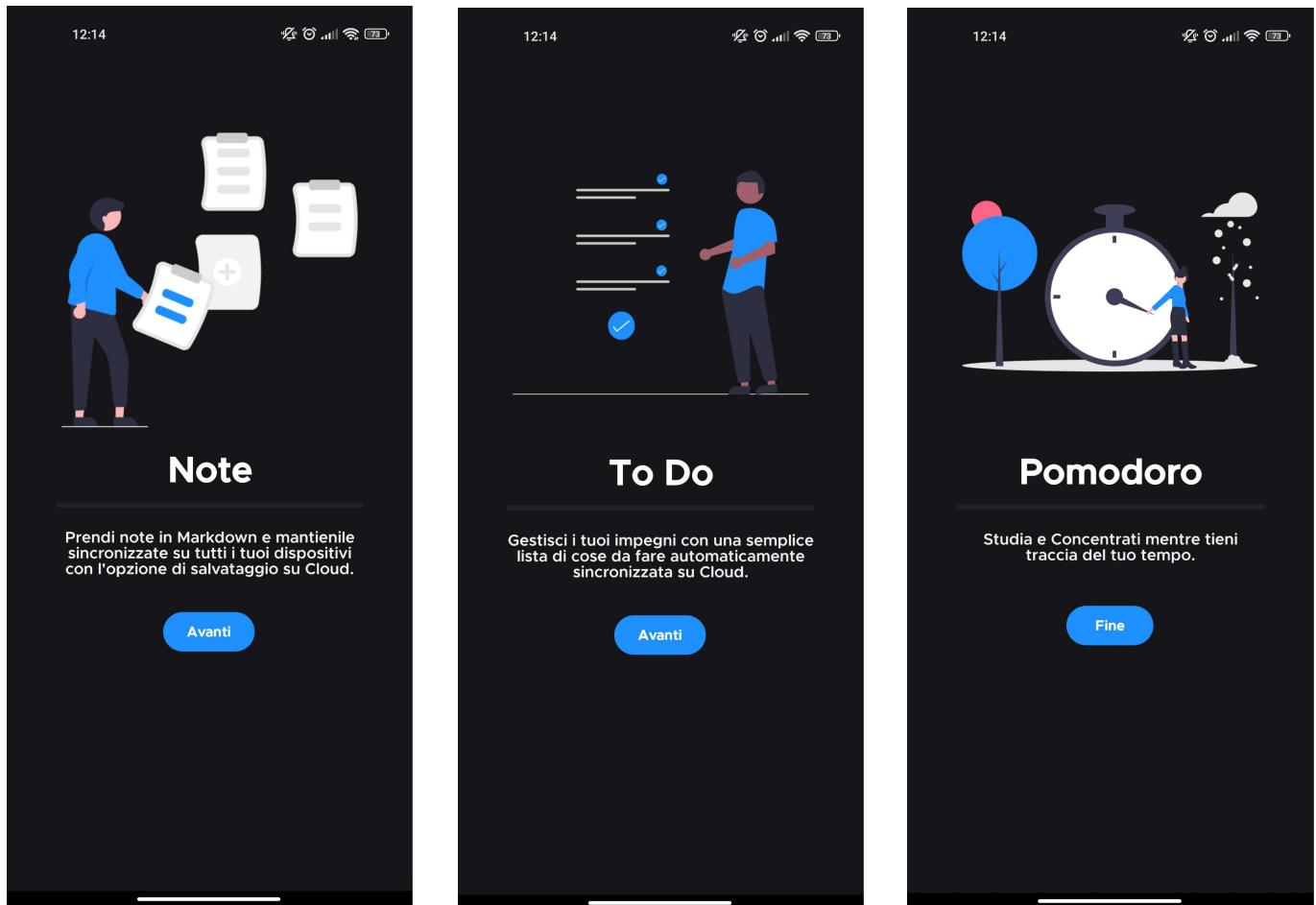
## SigninFragment



Fragment che permette di eseguire l'accesso all'account con email e password oppure sempre con Google.

In caso di password dimenticata è possibile premere su "Password Dimenticata?" per reimpostarla, dove verrà mostrato un dialog dove verrà richiesta la email per il recupero.

## OnboardingFragment



Fragment che mostra una piccola presentazione generale delle varie funzioni dell'applicazione all'utente. Queste schermate vengono mostrate la prima volta che l'utente apre l'app.

Nello specifico viene utilizzato un *ViewPager* con relativo *Adapter* per gestire lo scorrimento tra gli screen e alla fine della presentazione viene impostato nelle *SharedPreferences* un flag che viene utilizzato per non mostrare nuovamente la presentazione.

## Sviluppi Futuri

L'applicazione è stata pensata per essere aggiornata costantemente nel futuro, introducendo funzionalità ulteriori e modifiche apprezzabili da tutti gli utenti.

Nello specifico, in futuro verrà introdotto:

- 🌿 **Sistema di Amicizia:** un utente potrà richiedere l'amicizia ad altri utenti e avrà la possibilità di condividere le note con questi ultimi attraverso una cartella di "Note Condivise".
- 🌿 **Sistema di Statistiche:** l'utente potrà consultare alcune informazioni riguardanti l'utilizzo dell'applicazione, tempo speso, etc.
- 🌿 **Google Calendar API:** verrà data la possibilità di sincronizzare gli impegni specificati nelle ToDoList sul Calendario di Google attraverso l'utilizzo della API fornita da questi ultimi nel caso l'utente abbia un account Google.
- 🌿 **Aggiornamenti QOL:** questi aggiornamenti sono mirati a migliorare l'esperienza utente attuale. In particolare verranno introdotte alcune funzionalità mancanti nella attuale versione:
  - ⌚ Possibilità di rinominare le Note senza quindi doverle salvare in locale.
  - ⌚ Possibilità di scegliere dove salvare le Note sul Cloud. Al momento vengono tutte salvate nella directory principale se la Nota non è stata creata manualmente dal File Manager ma attraverso il tasto New del NoteFragment.
  - ⌚ Possibilità di ordinare le Note come più si preferisce (ordine alfabetico o personalizzato).

In generale quindi, gli aggiornamenti si concentreranno sull'estensione delle funzionalità già presenti, come la parte della ToDoList, il Metodo Pomodoro e altro, rendendo il tutto più funzionale e stabile.

Infine sarebbe interessante poter trasferire l'applicazione anche su altre piattaforme (in particolare iOS), magari spostandosi su soluzioni multiplattforma come Flutter, React Native o provare Kotlin Cross-Platform che al momento è in Beta.