**Capitolo 2**

**Descrizione dell’implementazione**

* 1. **Panoramica dell’implementazione**

In questo capitolo tratteremo come è stata svolta l’implementazione dell’applicativo, seguendo principalmente i seguenti obiettivi delineati dai requisiti:

* **Pubblicazione e gestione degli annunci**: Permettere agli utenti di pubblicare annunci sia per noleggi che per scambi, e gestirli efficacemente sia dalla parte del compratore che del venditore.
* **Miglioramento dell’esperienza utente**: Mostrare gli annunci in ordine di distanza nella homepage per facilitare la ricerca degli oggetti desiderati.
* **Servizio di messaggistica**: Garantire un sistema di messaggistica integrato per permettere agli utenti di comunicare e coordinare facilmente scambi e noleggi.
* **Gestione del profilo utente**: consentire agli utenti di visualizzare e modificare le proprie informazioni personali.

Il primo passo è stato identificare le tecnologie necessarie per realizzare l’applicazione. Questo ha comportato la scelta delle API e delle librerie più adatte per le funzionalità richieste, la selezione di servizi cloud affidabili per la gestione dei dati, e la configurazione di un database per memorizzare le informazioni sugli utenti.

Successivamente, è stata definita l’architettura del sistema, che includeva la struttura dell’applicazione e il flusso dei dati tra le varie componenti in modo da garantire scalabilità e facilità di manutenzione.

Con le tecnologie e l’architettura definite, si è proceduto con lo sviluppo vero e proprio dell’applicazione. Il processo di sviluppo è stato guidato dai requisiti prioritari, assicurando che le funzionalità essenziali fossero implementate per prime.

Nei prossimi paragrafi, esploreremo nel dettaglio come sono stati eseguiti tutti questi passi, fornendo una visione approfondita del processo di implementazione e delle scelte tecniche effettuate.

* 1. **Tecnologie utilizzate**

Durante lo sviluppo sono state sfruttate utilizzate diverse tecnologie, in questa sezione verranno descritte le principali tecnologie, come API (Application Programming Interface), framework o librerie utilizzate per sviluppare l’applicazione EcoSwap.

* + 1. **Flutter**

**Flutter** è stato scelto come framework di sviluppo principale per la sua capacità di creare applicazioni cross-platform con una singola base di codice. Flutter utilizza il linguaggio di programmazione **Dart** e offre una vasta gamma di widget per la creazione di interfacce utente. [1]

La scelta di Flutter è stata motivata dalla sua efficienza nello sviluppo e dalla facilità con cui consente di implementare funzionalità complesse.

* + 1. **Firebase**

Firebase è una piattaforma per lo sviluppo di applicazioni mobile e web, è stato integrato nell'applicazione per fornire una serie di servizi backend, essenziali per il funzionamento dell'app, come il salvataggio degli annunci e delle informazioni degli utenti in un cloud database. [2]

Le principali componenti di Firebase utilizzate in questo progetto sono:

* **Firebase Authentication**: Per la gestione dell'autenticazione degli utenti tramite e-mail, password e autenticazione Google.
* **Firebase Realtime Database**: Per l’archiviazione e la sincronizzazione dei dati in tempo reale, permettendo agli utenti di vedere aggiornamenti immediati sugli annunci.
* **Firebase Storage**: Per l’archiviazione sicura di immagini e altri file caricati dagli utenti.
  + 1. **Stripe**

Stripe Checkout è un'API di Stripe scelta per la gestione dei pagamenti all'interno dell'applicazione. Stripe offre una soluzione sicura e affidabile per elaborare pagamenti online, fornendo un'interfaccia di pagamento predefinita. Questa interfaccia garantisce che le transazioni degli utenti, effettuate con carta di credito o altre modalità di pagamento, siano gestite in modo sicuro e conforme agli standard internazionali. [3]

* + 1. **Geolocator e Geocoding**

Per le funzionalità basate sulla posizione geografica, sono state utilizzate le librerie **Geolocator** e **Geocoding**:

* **Geolocator**: Questa libreria è utilizzata per ottenere la posizione geografica attuale dell'utente, fondamentale per mostrare gli annunci vicini.
* **Geocoding**: Utilizzata per convertire indirizzi in coordinate geografiche (e viceversa), migliorando la precisione e l'usabilità delle ricerche geografiche, fondamentale anch’essa nel calcolo della distanza tra un utente e un annuncio.
  + 1. **Other libraries**

Oltre alle tecnologie principali sopra menzionate, l'applicazione utilizza diverse altre librerie per migliorare l'esperienza utente e facilitare lo sviluppo:

* **shared\_preferences**: Per memorizzare dati semplici in modo persistente, utilizzate per salvare localmente le informazioni dell’utente essenziali per riuscire ad effettuare un login automatico.
* **sqflite**: Per il salvataggio dei dati in locale, utilizzato per memorizzare le informazioni dell’utente e altri dati, permettendo così l'accesso ai dati anche offline.
* **image\_picker**: Per consentire agli utenti di selezionare e caricare immagini dai loro dispositivi, utilizzato specialmente per caricare le foto degli annunci o aggiornare l’immagine profilo.
* **flutter\_local\_notifications**: Questa libreria consente di programmare e gestire notifiche che possono avvisare gli utenti di eventi importanti.
* **provider**: Per la gestione dello stato dell'applicazione in modo reattivo e scalabile.
  1. **Architettura del Sistema**
     1. **Architettura generale**

Per lo sviluppo dell’applicazione, avevamo bisogno di un pattern architetturale che ci permettesse di separare in modo pulito la logica di business dall'interfaccia utente. Abbiamo quindi deciso di utilizzare un’architettura Model-View-ViewModel (MVVM), la quale risulta un’ottima architettura soprattutto da un punto di vista di manutenzione per diversi motivi [4]:

* **Separazione delle preoccupazioni**: MVVM ci permette di separare chiaramente la logica di business (gestita dai ViewModel) dalla logica di presentazione (gestita dalle View), facilitando la gestione e la manutenzione del codice.
* **Facilità di test**: Grazie a MVVM, possiamo testare la logica di business senza coinvolgere la UI, migliorando la qualità del codice attraverso test unitari.
* **Riutilizzo del codice**: La separazione delle logiche consente una maggiore riusabilità dei componenti del software, come molteplici ViewModel.
  + 1. **Componenti dell’architettura**

**View**

La View è responsabile della presentazione dell'interfaccia utente e dell'interazione con esso. In EcoSwap, le View sono implementate utilizzando i widget di Flutter. Questi widget osservano i dati forniti dai ViewModel e aggiornano automaticamente l'interfaccia utente in base alle modifiche.

Le principali responsabilità delle View nell'applicazione includono:

* **Visualizzazione degli annunci**: Mostrare gli annunci di scambio e noleggio agli utenti, organizzati in base alla distanza geografica.
* **Interazione con l'utente**: Gestire l'input dell'utente, come la pubblicazione di nuovi annunci e l'invio di messaggi.
* **Aggiornamento dinamico**: Aggiornare l'interfaccia in risposta ai cambiamenti nei dati del ViewModel

**ViewModel**

Il ViewModel agisce come intermediario tra la View e il repository (parte del Model), gestendo così la logica di presentazione e fornendo i dati necessari alla View. Questa separazione consente una maggiore modularità facilitando soprattutto il testing unitario della logica di business in quanto non convolge la UI.

In EcoSwap, i ViewModel sono due:

* **AdViewModel:** Gestisce tutto ciò che riguarda gli annunci, come caricamenti, eliminazioni e avvio dei noleggi, passando i dati dal repository alla View e viceversa.
* **UserViewModel:** Gestisce tutto ciò che riguarda gli utenti, come login, registrazione e aggiornamento della pagina profilo, anch'esso passando i dati dal repository alla View e avviando chiamate dalla View al repository.

**Repository**

Il Repository funge da intermediario tra i ViewModel e i DataSource. In EcoSwap, il Repository si occupa di gestire e indirizzare le chiamate tra i molteplici DataSource, decidendo, ad esempio, se salvare un'informazione localmente o in un database in cloud. Analogamente ai ViewModel, anche i repository sono due:

* **AdRepository:** Gestisce le chiamate tra i diversi DataSource dedicati agli annunci, selezionando il DataSource appropriato in base al tipo di annuncio.
* **UserRepository:** Gestisce le chiamate relative agli utenti, indirizzandole tra database locale o remoto.

**Data Source**

Il Data Source è la componente che interagisce direttamente con le sorgenti dati esterne, come Firebase e database locali[[1]](#footnote-1). Il Data Source si occupa di effettuare operazioni CRUD, gestendo quindi la creazione, lettura, aggiornamento e cancellazione dei dati da database locali o remoti.

In EcoSwap vi sono molteplici Data Source, uno per ogni modello rappresentato nell’applicazione. I principali sono:

* **UserDataSource:** Gestisce tutte le operazioni relative agli utenti, come la registrazione, il login o la modifica delle informazioni dell’utente, salvando tali informazioni in locale o in remoto, a seconda del contesto.
* **RentalDataSource e RentalLocalDataSource:** Gestiscono le operazioni relative ai noleggi, come la creazione, l’eliminazione o l’avvio di un noleggio.
* **ExchangeDataSource e ExchangeLocalDataSource:** Gestiscono le operazioni relative agli scambi, come la creazione e l’eliminazione.
* **ChatDataSource:** Gestisce le operazioni relative alle chat, come la creazione stream per l’inserimento di nuove chat o messaggi nel database.
  + 1. **Interazioni nell’architettura**

Il flusso dei dati nell'architettura MVVM di EcoSwap segue questo schema:

1. **Utente interagisce con la View**: L'utente compie un'azione, come pubblicare un annuncio o avviare un noleggio.
2. **View comunica con il ViewModel**: La View invia l'azione al ViewModel appropriato.
3. **ViewModel interagisce con il Repository**: Il ViewModel richiama le operazioni necessarie nel Repository.
4. **Repository accede ai Data Source**: Il Repository effettua le operazioni sui dati richiamando i DataSource necessari.
5. **DataSource aggiorna i dati**: Il DataSource effettua le operazioni di aggiornamento dei dati.
6. **Aggiornamento dei dati**: Una volta che i dati sono aggiornati, la View viene notificata dei cambiamenti grazie al meccanismo di Future di Flutter[[2]](#footnote-2), aggiornando di conseguenza l’interfaccia utente.

Le interazioni tra le componenti dell'architettura sono illustrate nel diagramma seguente:

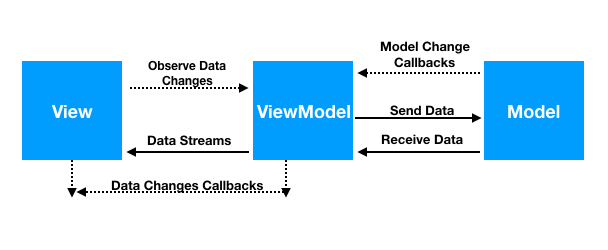


Figura 2.1: Schema rappresentate il funzionamento del Pattern MVVM

* 1. **Implementazione delle pagine principali**

EcoSwap è composta da cinque pagine principali, ognuna accessibile tramite la barra di navigazione inferiore (bottom bar). Ogni pagina svolge un ruolo fondamentale nell’implementazione di diverse funzionalità chiave, garantendo una user experience fluida e intuitiva:

* + 1. **HomePage**

La Home Page è la schermata principale dell'applicazione, dove gli utenti possono visualizzare gli annunci disponibili per scambi e noleggi. Gli annunci sono ordinati per distanza dalla posizione dell'utente per facilitare la ricerca. La HomePage è composta da due sezioni, una dedicata ai noleggi e l’altra agli scambi, accessibili tramite un pulsante in cima alla pagina. Per implementare la visualizzazione degli annunci è stata utilizzata una *GridView* che, insieme a un meccanismo di caricamento progressivo, permette di visualizzare continuamente nuovi annunci in base alla distanza dalla posizione attuale dell’utente.

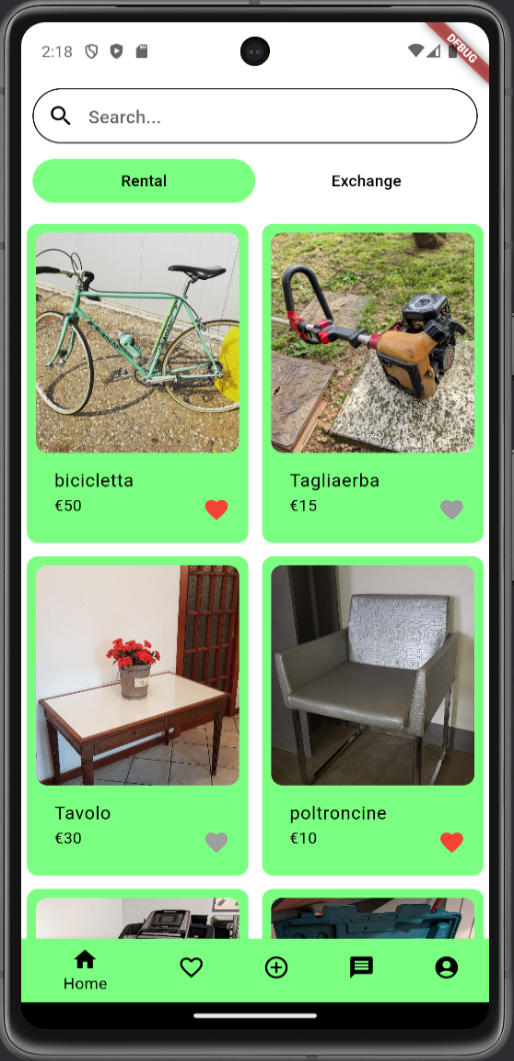
 

Figura 2.2: Immagine rappresentate la HomePage di EcoSwap

In cima alla pagina è presente una barra di ricerca che consente di cercare annunci specifici. Dalla Home Page, gli utenti possono anche salvare gli annunci nei preferiti o cliccare su di essi per visualizzare i dettagli. Da qui, è poi possibile decidere se avviare il noleggio, lo scambio o aprire una chat con il venditore.

* + 1. **Pagina dei preferiti**

Attraverso la pagina dei preferiti è possibile visualizzare gli annunci salvati grazie a una *ListView*, organizzati in modo simile alla Home Page con una divisione tra noleggi e scambi per garantire maggiore chiarezza. Cliccando su uno degli annunci, si può visitare la pagina specifica dell’annuncio, proprio come avviene nella Home Page.

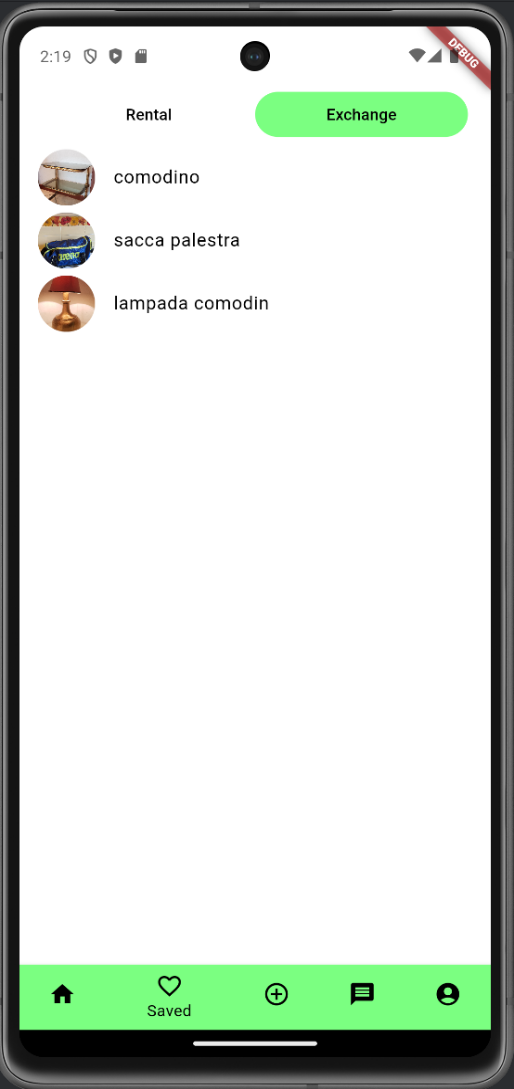
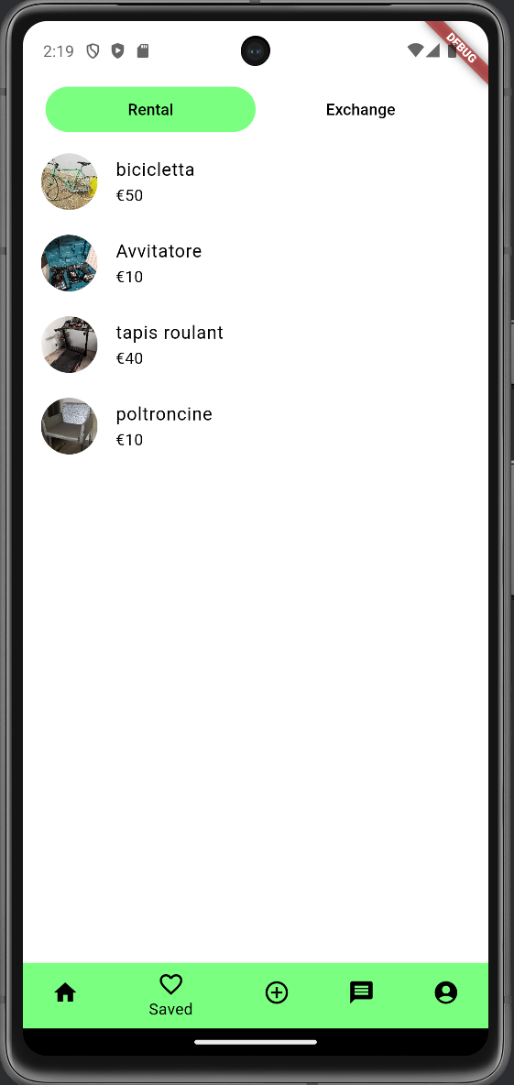


Figura 2.3: Immagine rappresentate la Pagina dei preferiti di EcoSwap

* + 1. **Pagina di caricamento**

Questa pagina implementa l’obiettivo principale dell’applicazione, ossia il caricamento degli annunci. La struttura della pagina è divisa in modo da distinguere tra noleggi e scambi; infatti, in base al tipo di annuncio che si vuole caricare, la pagina presenta campi differenti. I campi comuni a entrambi i tipi di annuncio sono il titolo, la descrizione e la posizione geografica, che può essere indicata sulla mappa. Per facilitare l’inserimento della posizione, è stata integrata una mappa interattiva che consente agli utenti di selezionare facilmente il luogo desiderato. Oltre a questi campi comuni, la pagina offre sezioni specifiche per raccogliere informazioni dettagliate pertinenti a noleggi e scambi, assicurando che tutti i dettagli necessari siano forniti in modo chiaro e organizzato.

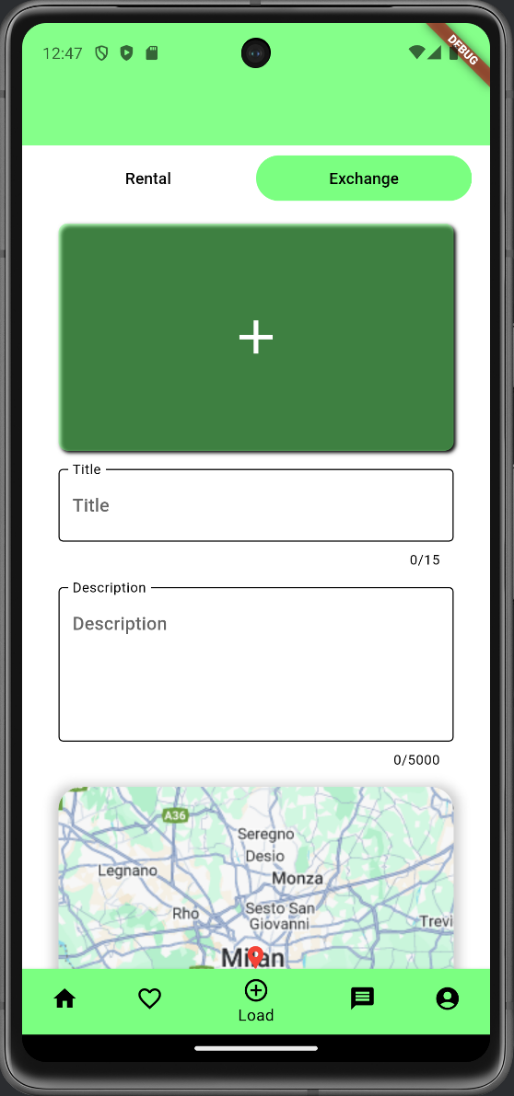
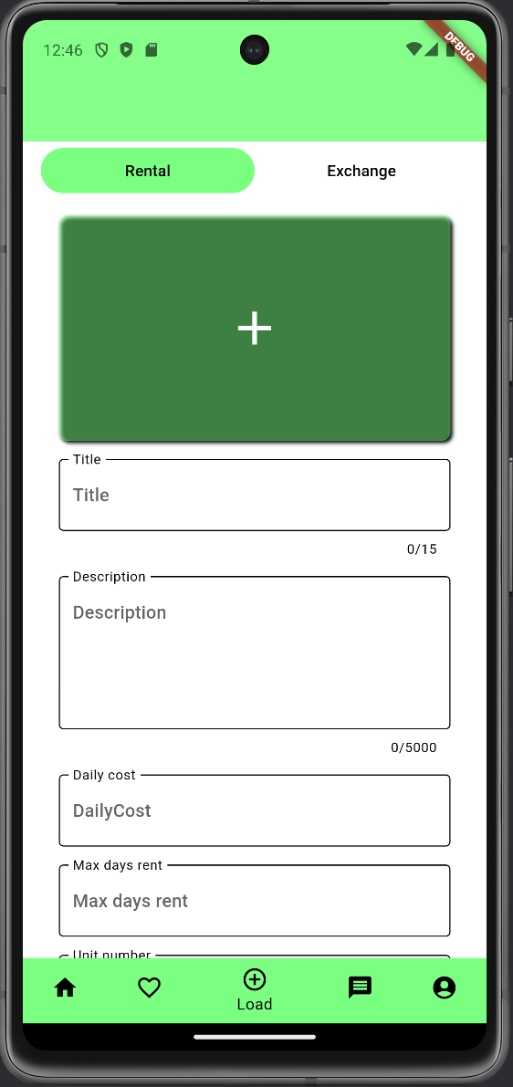


Figura 2.4: Immagine rappresentate la Pagina di caricamento di EcoSwap

Per implementare questa pagina, ci siamo serviti della libreria *ImagePicker* per permettere all’utente di caricare immagini dal proprio telefono direttamente sull’applicazione premendo il tasto (+). Dopo aver caricato l'immagine, essa verrà mostrata immediatamente. Per creare il meccanismo della mappa, abbiamo sviluppato un nuovo widget chiamato *MapWidget*. Utilizzando il widget *FlutterMap* e la posizione attuale dell’utente ottenuta grazie alle librerie di geolocalizzazione, la mappa viene inquadrata su un'area attorno ad esso, permettendogli di selezionare un qualsiasi punto per segnare il luogo dello scambio/noleggio.

* + 1. **Pagina delle chat**

In questa pagina è possibile visualizzare tutti i messaggi scambiati con altri utenti. Le chat sono ordinate temporalmente e sono basate su un annuncio specifico. È quindi possibile avere più chat con un solo utente, ma solo per annunci differenti. Aprendo la chat, è possibile controllare a quale annuncio si sta facendo riferimento. Inoltre, è stato implementato un meccanismo per mostrare se l’ultimo messaggio è stato letto oppure no, grazie a un campo booleano aggiunto ai messaggi.

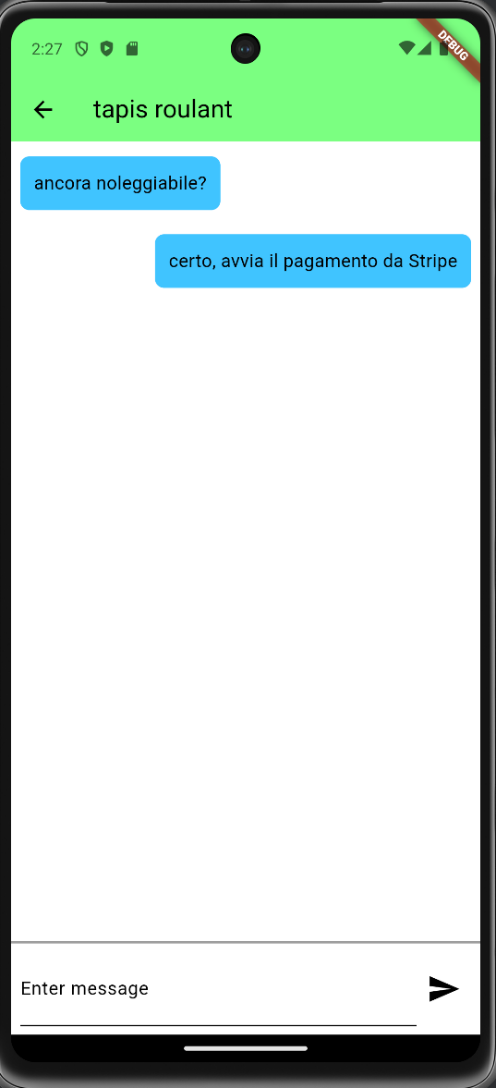
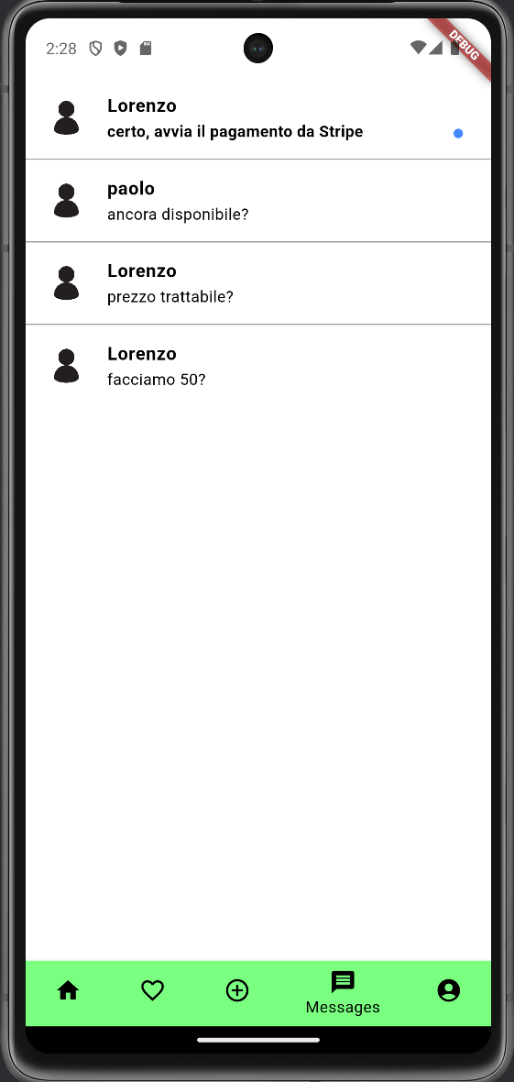


Figura 2.5: Immagine rappresentate la Pagina delle chat di EcoSwap

Per avere una chat con messaggi in tempo reale, abbiamo utilizzato uno stream di *DatabaseEvent* connesso a Firebase, che consente di visualizzare un nuovo messaggio ogni volta che esso viene salvato su Firebase.

* + 1. **Pagina del profilo**

Nella pagina del profilo, è possibile visualizzare le informazioni dell’utente loggato, come le recensioni premendo l’apposito bottone o le varie liste degli annunci, quali i noleggi attivi venduti, i noleggi attivi acquistati, i noleggi pubblicati e gli scambi pubblicati. È inoltre possibile accedere ai precedenti ordini effettuati, modificare le informazioni del proprio profilo, oppure effettuare il logout premendo i tre puntini nella parte superiore dello schermo.

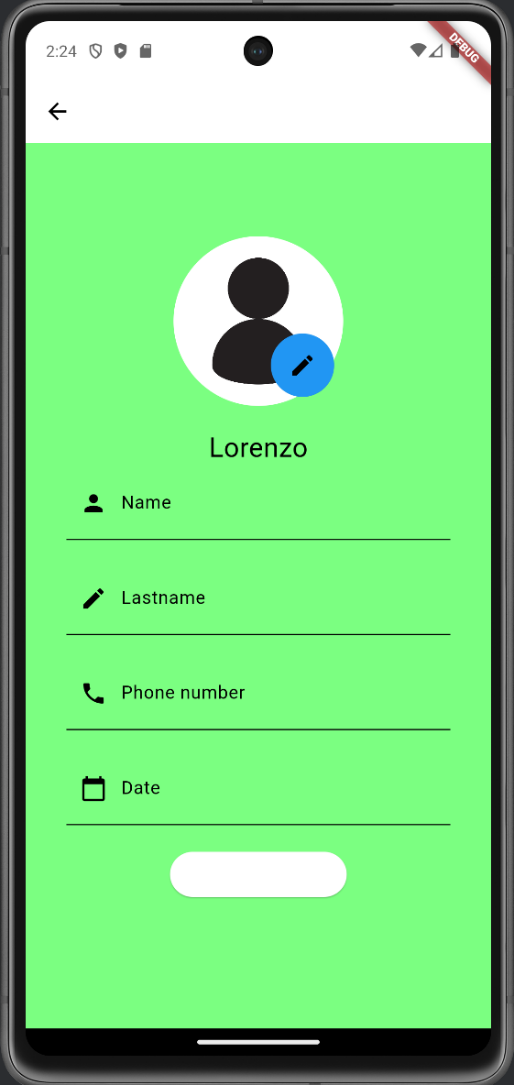
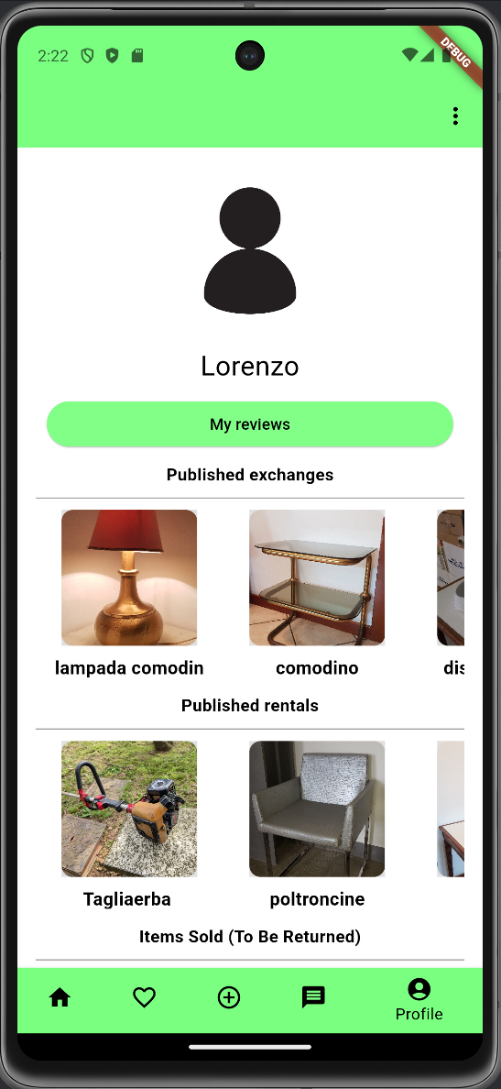


Figura 2.6: Immagine rappresentate la Pagina del profilo di EcoSwap

Inoltre, è possibile cliccare su un annuncio nelle varie liste per effettuare operazioni specifiche. Ad esempio, è possibile avviare il processo di reclamo o restituzione per un noleggio, gestire le interazioni con gli altri utenti o eliminare un annuncio dalla lista degli annunci pubblicati.

* + 1. **Pagina di Login e Registrazione**

Altre pagine molto importanti sono quelle di login e registrazione dato che attraverso queste pagine è possibile registrarsi o accedere all’app. Grazie all'utilizzo della libreria *SharedPreferences*, il login diventerà automatico fino a che non si rieffettuerà un logout attraverso la pagina del profilo. Vi è anche la possibilità di accedere tramite un account Google, rendendo il processo di login più rapido e semplice. Inoltre, nel caso si sia dimenticata la password, è possibile recuperarla facilmente inserendo il proprio indirizzo e-mail. Una volta inserito l'indirizzo e-mail, verrà inviata una mail con le istruzioni per reimpostare la password, garantendo così un processo di recupero intuitivo per l'utente.

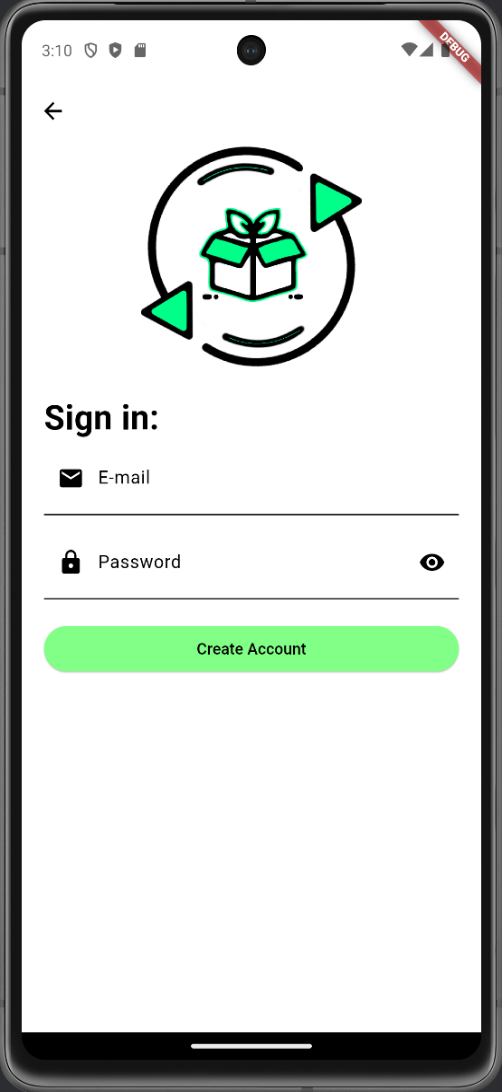
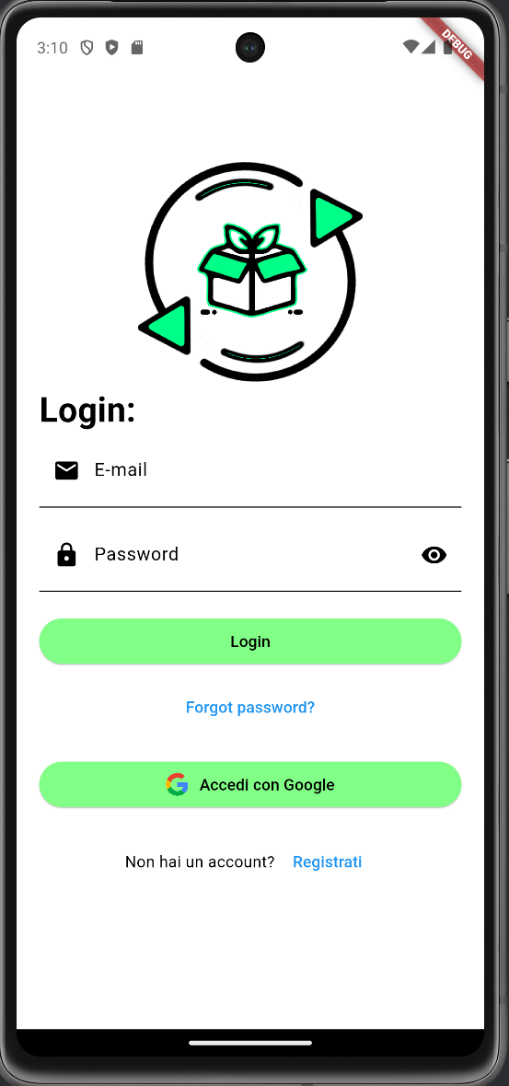


Figura 2.6: Immagine rappresentate la Pagina di Login di EcoSwap

* 1. **Conclusioni**

Nello sviluppo di EcoSwap, il nostro obiettivo era offrire un'alternativa sostenibile alle tradizionali piattaforme di consumo. Le funzionalità principali dell'applicazione, come la gestione degli annunci, le chat in tempo reale e il profilo utente, sono state implementate con successo per promuovere scambi eco-sostenibili. Tuttavia, riteniamo che ci siano opportunità per migliorare ulteriormente l'esperienza degli utenti. Un'interfaccia con un design più pulito e un sistema di notifiche più avanzato, che avvisi gli utenti quando un noleggio sta per scadere o altre informazioni rilevanti, potrebbe essere considerato per migliorare l'usabilità e l’esperienza degli utenti.

**Bibliografia capitolo**

[1] Flutter. *Flutter development documentation.* Url: <https://flutter.dev/development>

[2] Google Firebase. *Google Firebase documentation.* URL:

<https://firebase.google.com/docs?hl=it>

[3] Stripe. *Online Payment Processing for Internet Businesses.* URL: <https://docs.stripe.com/>

[4] Pattern MVVM. *Model-view-viewmodel Design Pattern.* Url: <https://it.wikipedia.org/wiki/Model-view-viewmodel>

1. Il salvataggio in database locali può risultare utile in caso di mancanza di connessione, permettendo comunque un utilizzo limitato dell'applicazione, nel nostro caso, è possibile solamente visualizzare le informazioni personali. [↑](#footnote-ref-1)
2. In Flutter, un Future è un oggetto che rappresenta un’operazione asincrona. Quando l’operazione è completa, il Future notifica i suoi ascoltatori, permettendo la gestione reattiva degli aggiornamenti dello stato. [↑](#footnote-ref-2)