

TrackItEasy

Documentazione TrackItEasy

Gruppo: Kotlin Enjoyers

Componenti Gruppo:

Marco Sanvito 886493

Emre Tugrul 886027

Sommario

1. Introduzione
2. Funzionalità
3. Librerie
4. Architettura
5. Design
6. Material Design
 - a. Dynamic design
7. Pattern
8. Sviluppi futuri

Introduzione

L'obiettivo del progetto consiste nello sviluppare un'applicazione dedicata al monitoraggio e alla gestione delle spedizioni dei pacchi, offrendo un notevole vantaggio nell'organizzare i pacchi provenienti da diverse compagnie, tra cui Amazon e Poste Italiane. Questa soluzione elimina la necessità di controllare separatamente ogni spedizione, consentendo agli utenti di gestire l'intero processo in un'unica piattaforma. L'idea parte dall'esigenza di avere molteplici pacchi da corrieri diversi che risulta difficile da gestire ordinando molti pacchi.

Funzionalità

Le funzionalità principali dell'applicazione includono l'aggiunta di nuovi pacchi, la loro organizzazione nell'archivio e la possibilità di rimuoverli. Un ulteriore vantaggio è la capacità di ordinare e filtrare i pacchi attraverso un sistema di ricerca.

Si possono inoltre visualizzare i dettagli dei pacchi e vedere a che punto è la spedizione. Si possono salvare i propri dati sul dispositivo o sul Cloud tramite login in base alle proprie esigenze. Nel caso in cui non ci sia la connessione ad internet viene mostrato un simbolo che lo indica. In questo stato sarà comunque possibile effettuare le operazioni che non richiedono una connessione ad internet come visualizzare i pacchi, filtrare, cambiare le impostazioni.

Viene fornita anche la possibilità di collegare più account Amazon così da tenere traccia dei pacchi dei vari account nello stesso posto. Questo può tornare molto utile se ad esempio si hanno più account in famiglia e si può tenere traccia di tutto con l'app.

Le schermate chiave dell'applicazione comprendono la Home Screen, l'Archivio e il Menù, che offre sezioni dedicate all'account dell'utente, alle impostazioni, al feedback e al supporto.

La barra superiore offre funzionalità aggiuntive in base alla pagina in cui ci si trova. Invece la barra di navigazione inferiore permette agli utenti di passare agevolmente da una schermata all'altra, garantendo un'esperienza di utilizzo intuitiva e fluida.

Librerie Utilizzate

UI:

- Jetpack Compose
 - Permette di creare un'interfaccia utente in un modo moderno rendendola più mantenibile e compatibile con il Material Design

Storage:

- Room
- Proto DataStore (La versione moderna delle Shared Preferences)
- Firestore (Cloud Storage)

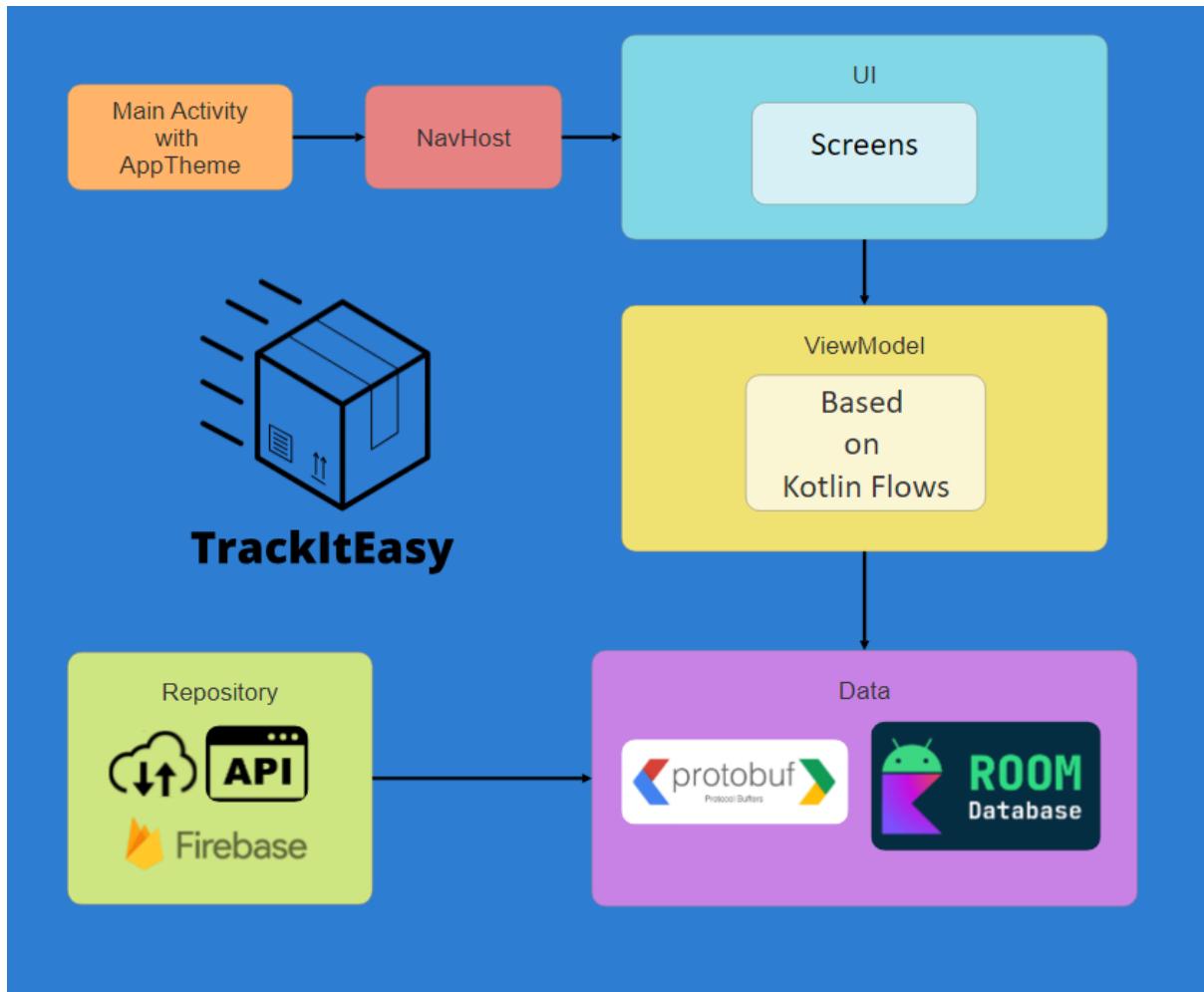
Librerie Esterne:

- Coil (immagini)
- HtmlUnit (Web Scraping)
- Ksoup (Parsing HTML)
- XmlToJson

Librerie Importanti:

- WebKit
 - WebView per permettere il login con amazon
- WorkManager
 - Gestione del lavoro mentre l'app è in background
- Navigation Compose
 - Permette la navigazione tra le diverse componenti dell'app in un approccio più moderno e efficiente
- Firebase
 - Permette l'autenticazione tramite mail e password e l'utilizzo del Cloud Firestore per salvare i dati dell'utente

Architettura



L'applicazione si compone essenzialmente di queste componenti principali:

Main Activity:

- Questa rappresenta il nucleo dell'applicazione, sviluppato in modo unico utilizzando Jetpack Compose per garantire una UI moderna e reattiva.

NavHost:

- Assume il compito di gestire la navigazione tra le varie schermate dell'app. Funziona come un contenitore dinamico in grado di ospitare i fragment componibili che costituiscono le diverse schermate dell'app. Inoltre, agevola il trasferimento di dati cruciali tra le schermate, garantendo un flusso fluido e intuitivo attraverso l'applicazione. In sostanza, il NavHost facilita la navigazione dell'utente all'interno

dell'applicazione, offrendo un'esperienza di navigazione coerente e ben integrata.

UI:

- è responsabile della presentazione visuale e interattiva dell'applicazione. È composto da una serie di fragment componibili, ognuno dei quali svolge specifiche responsabilità per garantire una gestione chiara e modulare dell'interfaccia utente.

ViewModel:

- Funge da intermediario tra l'interfaccia utente (UI) e i dati sottostanti. Grazie all'utilizzo dei Flow di Kotlin, la ViewModel è in grado di ricevere in modo reattivo le informazioni dai repository o da altre fonti di dati, garantendo un'esperienza utente fluida e aggiornata in tempo reale. Questo approccio elimina la necessità di monitorare direttamente gli oggetti tramite LiveData, offrendo invece un'implementazione più versatile e performante. Inoltre, l'utilizzo dei Flow rende la ViewModel più efficiente e indipendente dalla piattaforma Android, consentendo una migliore separazione delle responsabilità e una maggiore modularità nel codice.

Data:

In questo strato, utilizziamo diverse tecnologie per garantire la sicurezza e l'efficienza delle informazioni trattate:

- Persistenza dei dati con Room: Utilizziamo la libreria Room per la gestione della persistenza dei dati, consentendo di memorizzare e recuperare facilmente le informazioni dei pacchi in un database locale. Questo approccio garantisce una gestione affidabile e performante dei dati all'interno dell'applicazione.
- Gestione sicura delle credenziali con Android Key Store e Cipher: Le informazioni sensibili, come gli account Amazon, vengono criptate utilizzando Android Key Store e la classe Cipher, offrendo un livello aggiuntivo di sicurezza per proteggere i dati dell'utente da accessi non autorizzati.
- Memorizzazione delle impostazioni e dati interni con Proto DataStore: utilizziamo Proto DataStore per memorizzare in modo efficiente le impostazioni dell'applicazione e altri dati interni necessari per il corretto funzionamento dell'applicazione. Proto DataStore offre un'alternativa moderna e performante alle Shared Preferences,

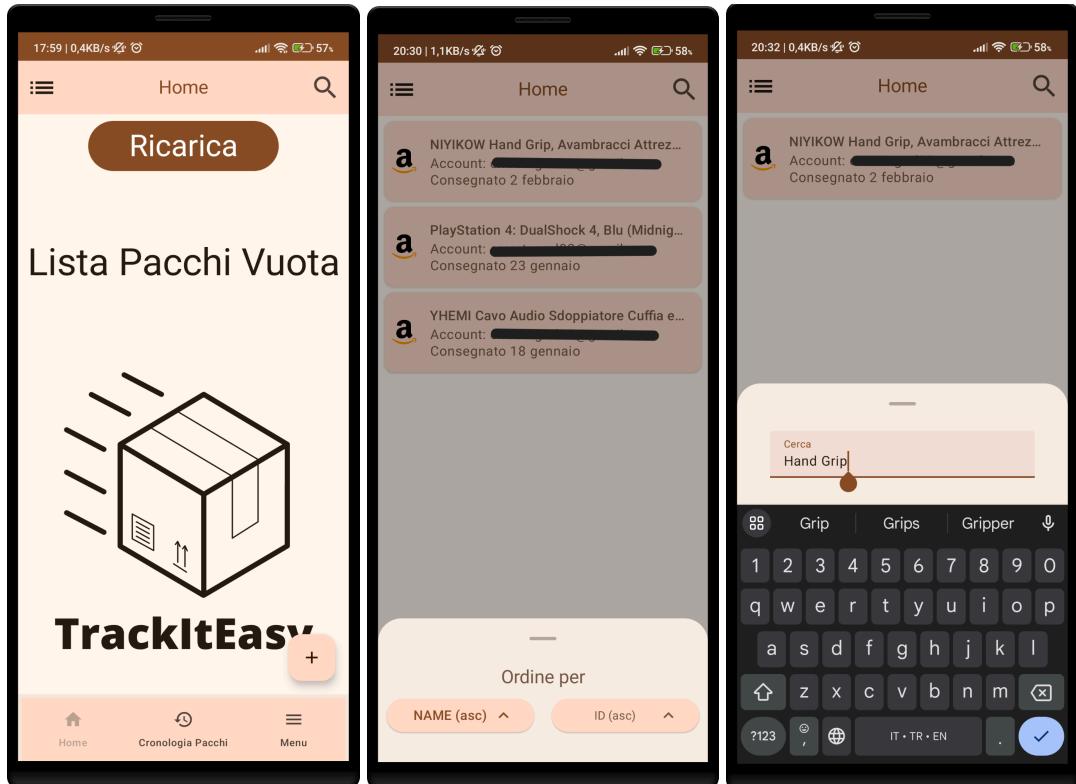
consentendo una gestione più efficiente dei dati interni dell'applicazione.

Repository:

- Gestisce il recupero e il parsing dei pacchi da vari fornitori come Amazon, Poste Italiane e TNT. Per integrare al meglio con le coroutines di Kotlin e garantire prestazioni ottimali, abbiamo optato per l'utilizzo di HttpURLConnection anziché librerie come Retrofit. Questo approccio ci consente di gestire in modo efficiente le operazioni di rete e di adattarci alle specifiche esigenze dei diversi fornitori, garantendo una migliore esperienza complessiva agli utenti dell'applicazione.
- Inoltre utilizziamo Cloud Firestore, il database NoSQL in tempo reale di Firebase, per gestire e memorizzare la storia dei pacchi degli utenti loggati nel cloud. Ciò consente agli utenti di sincronizzare i loro dati tra dispositivi e di accedere alla loro storia dei pacchi da qualsiasi luogo e in qualsiasi momento.
- Infine questo layer ci permette di gestire il collegamento tra le chiamate dall'interfaccia utente e il data layer separando le responsabilità

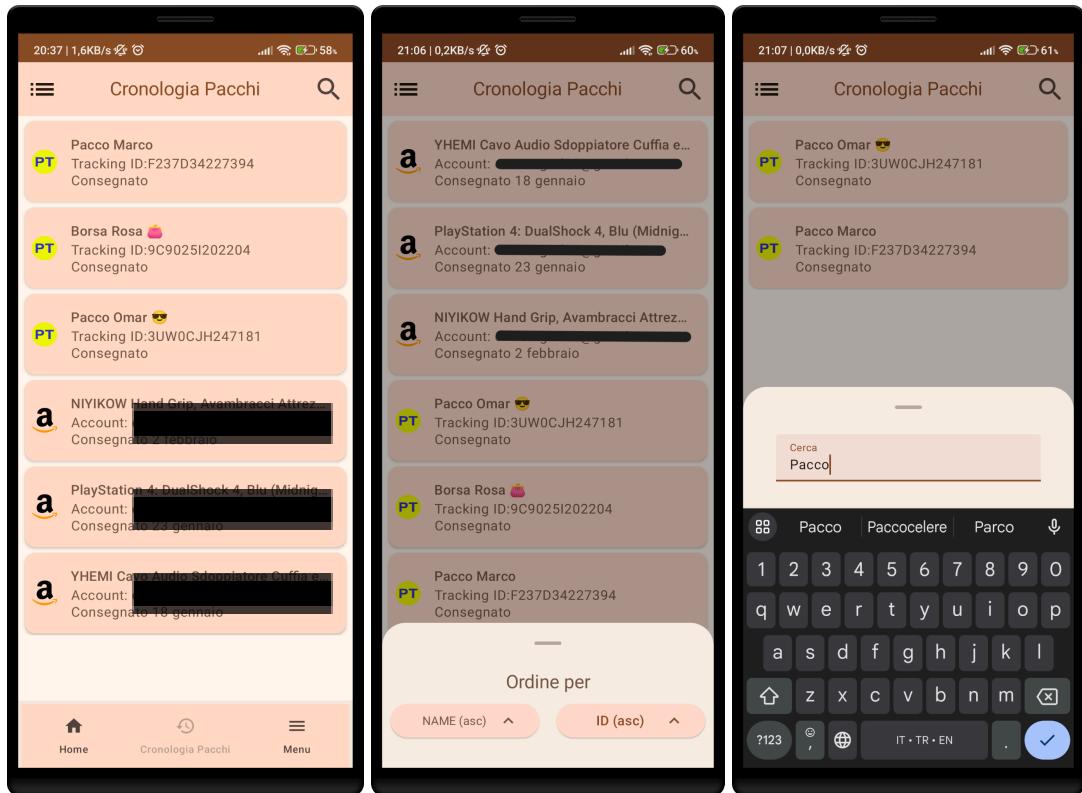
Design

Home Screen



La schermata **Home** costituisce il cuore dell'applicazione, pensata per gestire i pacchi in arrivo o recentemente consegnati. In cima alla schermata, sono posizionati i pulsanti **Ordina** (per organizzare i pacchi in base al nome del pacco o dell'azienda) e **Ricerca** (per individuare pacchi specifici attraverso il nome, l'azienda o criteri simili). Un ulteriore strumento è il pulsante "Ricarica", utile per aggiornare la schermata principale con i nuovi pacchi. Questa operazione si svolge automaticamente in background ogni 15 minuti, garantendo la verifica di eventuali cambiamenti nello stato attuale dei pacchi e l'aggiornamento di conseguenza. Durante il processo di aggiornamento, viene visualizzata una rotella in movimento, indicando all'utente che l'applicazione sta elaborando dati (oppure è possibile aggiornare la pagina tramite uno swipe verso il basso).

History Screen

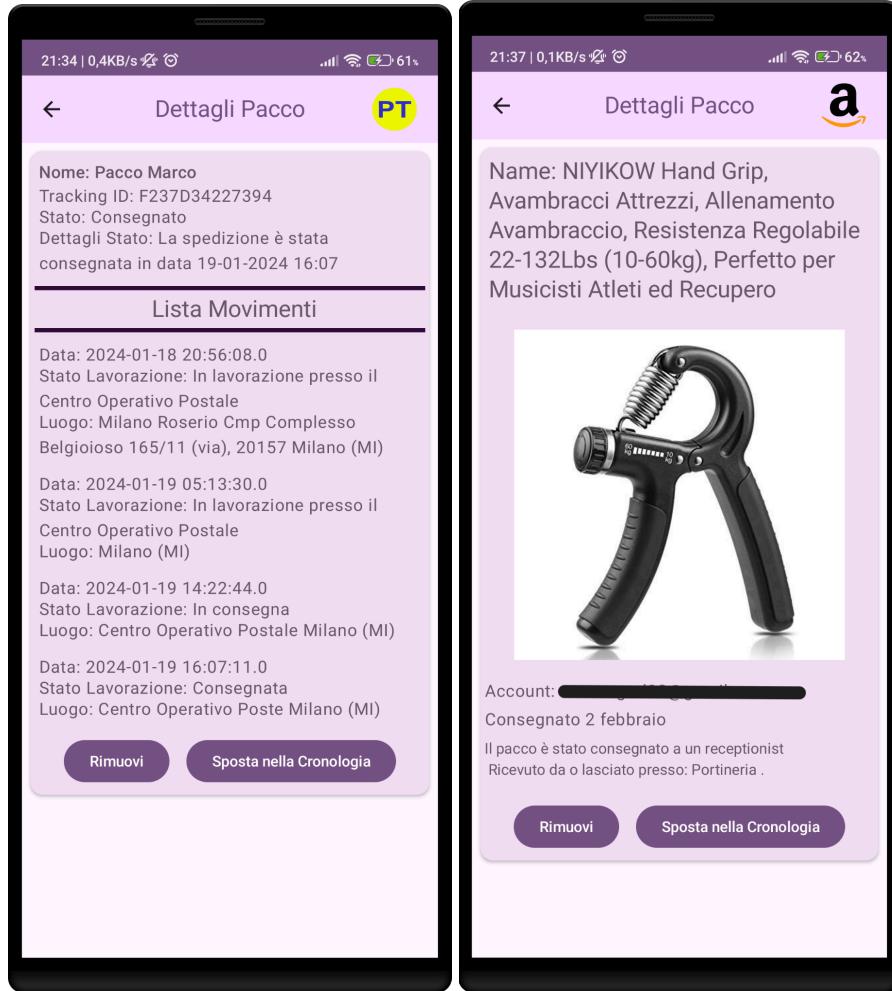


La schermata **History** funge da archivio, progettato per ospitare i pacchi che non è più necessario visualizzare nella schermata principale poiché sono già stati consegnati. Qui, gli utenti possono ordinare i pacchi o effettuare ricerche specifiche.

Analogamente alla schermata **Home**, è possibile cliccare su un pacco per visualizzarne i dettagli e rimuoverlo dall'archivio. Questa disposizione offre un modo efficiente per mantenere traccia storica delle spedizioni, semplificando al contempo la gestione e la consultazione di pacchi passati.

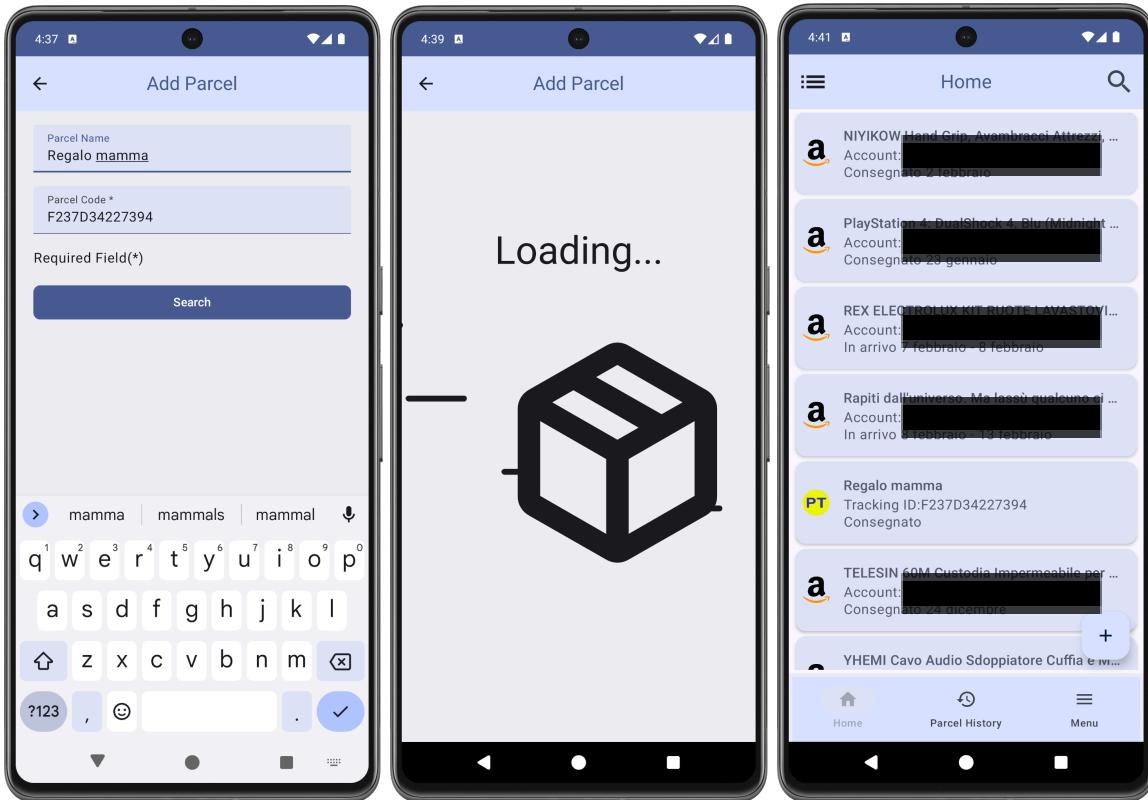
Questa sezione non ha bisogno di connessione ad internet e si può navigare liberamente.

ParcelDetails Screen



Se nella schermata **Home** sono presenti pacchi, è possibile fare clic su uno di essi per accedere ai dettagli del pacco, spostarlo nell'archivio o rimuoverlo. Si può inoltre vedere la **lista movimenti** per monitorare il percorso del pacco. Questo può tornare particolarmente utile quando il pacco arriva da una destinazione estera, in quanto le date sono indicative e non molto precise e può succedere che il pacco arrivi quando non si è disponibili a riceverlo. Avendo a disposizione la lista movimenti si ha un maggiore controllo sulla situazione e si può vedere quando il pacco ha raggiunto il centro operativo della propria città. Su Amazon si viene notificati quando il pacco è in consegna e quindi è stato inserito uno stato che indica lo stato attuale del pacco che tra quanti giorni arriverà oppure “arriverà domani”, “in consegna” e così via. Questa funzionalità offre un controllo immediato e pratico sulla gestione delle spedizioni direttamente dalla schermata principale.

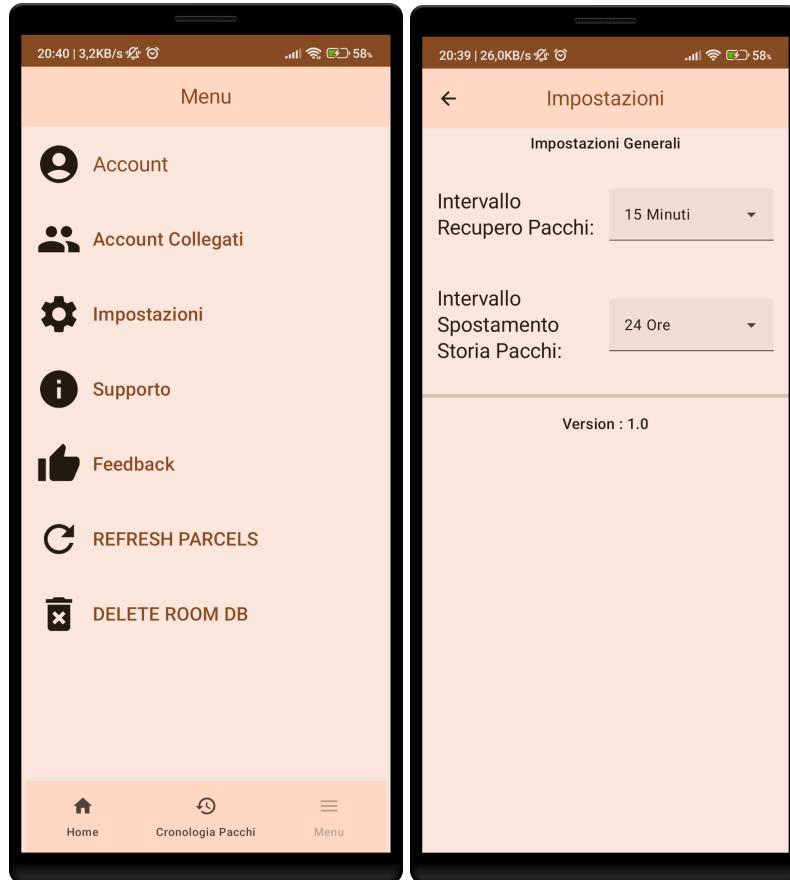
ParcelEntry Screen



Quando si seleziona l'opzione per aggiungere un nuovo pacco, si accede alla schermata **ParcelEntry**. In questa sezione, gli utenti hanno la possibilità di inserire manualmente i dati del pacco. Il campo **Nome** è opzionale e concepito per assegnare un nome personalizzato al pacco, rendendolo facilmente identificabile. Nel campo **Codice**, invece, si inserisce il codice spedizione fornito dal corriere.

Dopo aver inserito i dati, al momento della pressione su **Cerca**, viene avviata un'animazione che indica l'effettuazione di un tentativo di aggiunta del pacco. Subito dopo, si visualizza un resoconto dell'operazione. È importante notare che questa sezione richiede l'utilizzo della connessione, poiché viene verificato che il codice inserito sia valido ed esistente. Successivamente, il pacco inserito sarà visibile nella schermata principale **Home**, semplificando così la gestione centralizzata di tutti i pacchi.

Settings Screen

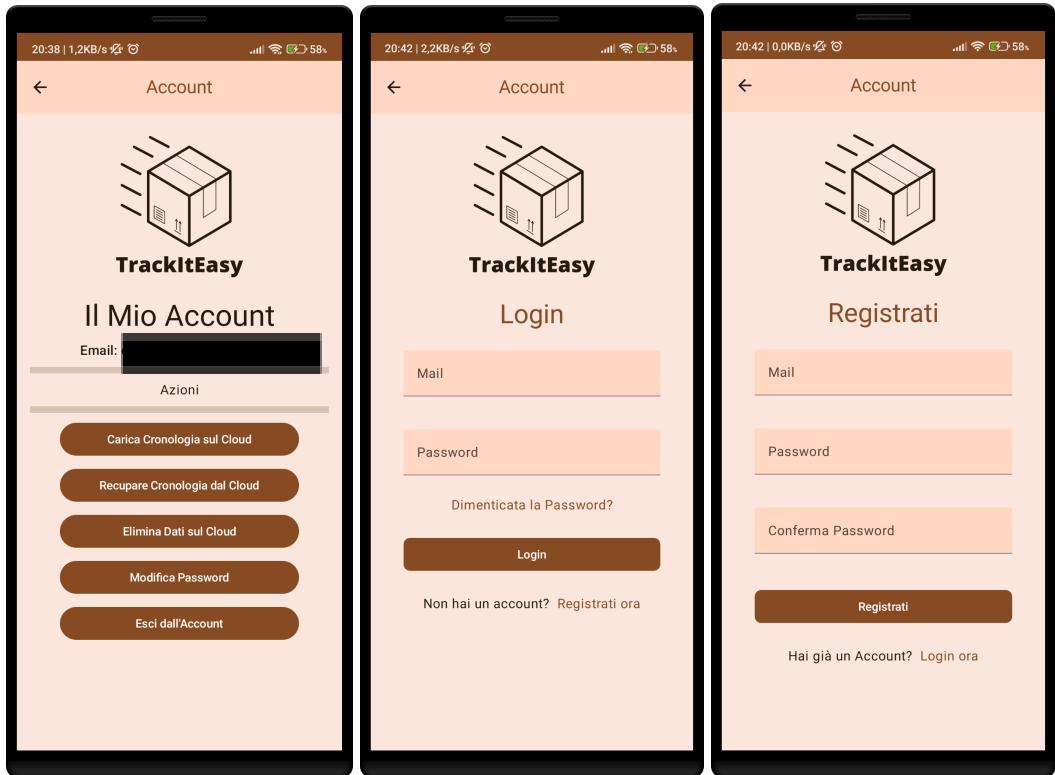


La schermata del **Menu** offre diverse funzionalità supplementari, tra cui le Impostazioni, la gestione dell'account, il collegamento con l'account Amazon, la possibilità di fornire feedback e l'accesso al supporto tecnico.

Nella sezione **Impostazioni**, gli utenti possono personalizzare l'esperienza regolando l'intervallo di tempo tra le sincronizzazioni e decidendo dopo quanti giorni un pacco viene automaticamente spostato nell'archivio. Questa flessibilità consente agli utenti di adattare l'app alle proprie esigenze.

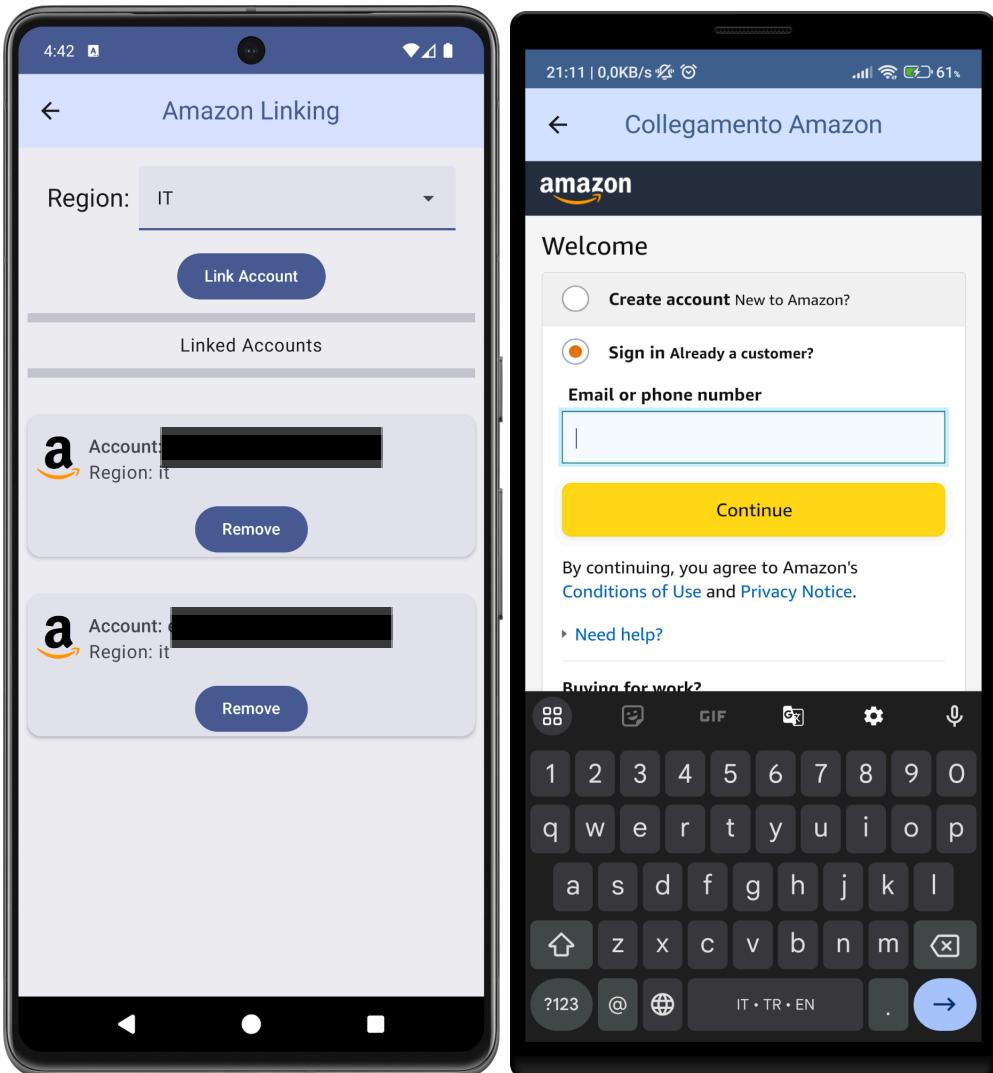
Le sezioni **Feedback** e **Supporto** offrono agli utenti l'opportunità di contribuire all'applicazione. La sezione **Feedback** è dedicata alle opinioni generali sull'esperienza dell'utente, consentendo agli utenti di condividere i loro pensieri e suggerimenti per migliorare l'applicazione. D'altra parte, la sezione **Supporto** è progettata per segnalare eventuali bug o problemi tecnici, facilitando così la manutenzione e il continuo perfezionamento dell'applicazione. Questa divisione delle sezioni consente una partecipazione attiva degli utenti e torna utile per ottimizzare l'efficacia dell'app.

Account Screen



Nella sezione **Account**, gli utenti possono autenticarsi o registrarsi. Una volta effettuato l'accesso, si aprono nuove opzioni, consentendo di cambiare le proprie credenziali, salvare, recuperare o rimuovere pacchi dal cloud. Questa caratteristica risulta preziosa quando è necessario resettare o cambiare dispositivo, in quanto i dati possono essere archiviati nel cloud e recuperati successivamente mediante autenticazione. Se non si ricorda la password si può sempre recuperare nella sezione login e inserendo la mail verrà fornito un link in cui è possibile cambiarla. In aggiunta, è possibile eliminare i pacchi salvati nel cloud o effettuare il logout dall'account. Questa sezione è dipendente dalla connessione a Internet, inclusa la gestione delle operazioni con il Cloud. L'operazione di **Logout**, tuttavia, è gestita client-side e non richiede interazioni con il server, permettendo agli utenti di eseguirla anche in assenza di connessione.

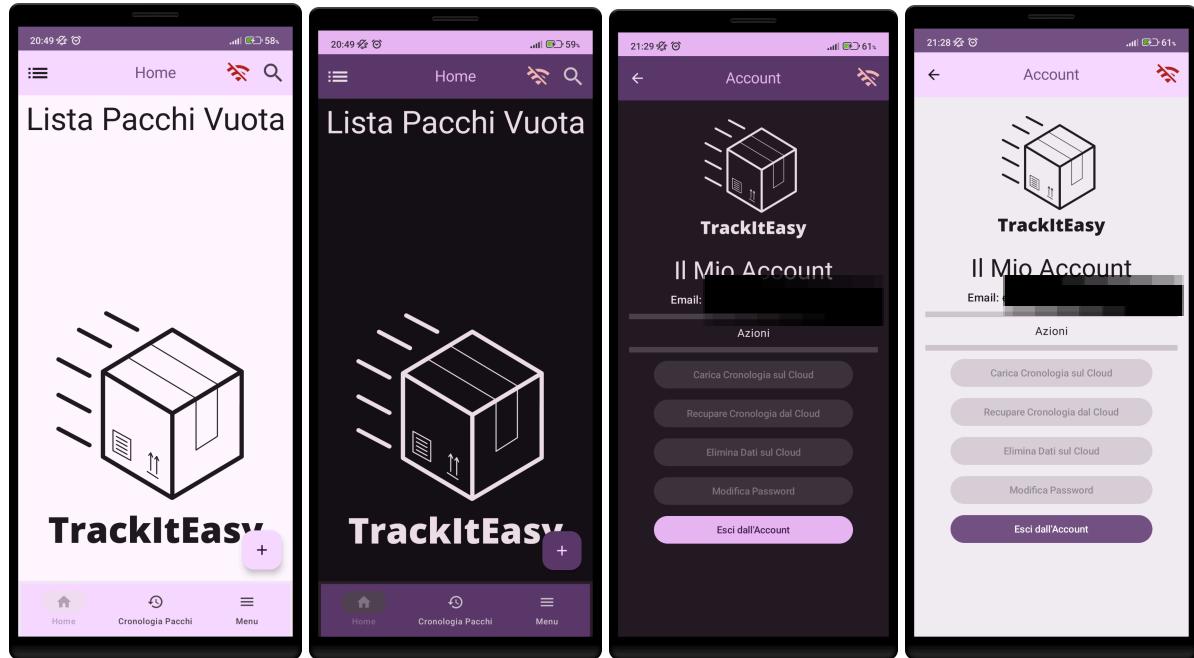
Amazon Account Screen



Il [login con Amazon](#) rappresenta un'altra funzionalità chiave che semplifica il recupero dei pacchi direttamente dall'account Amazon dell'utente. Una volta effettuato l'accesso, l'account Amazon viene collegato, consentendo il recupero dei pacchi e la ricezione degli aggiornamenti futuri direttamente dall'app.

Nella sezione [Linking Account](#), l'utente può rimuovere l'account Amazon, eliminando così i pacchi associati. È importante notare che, nonostante la rimozione, i pacchi nell'archivio rimarranno disponibili come storico, garantendo agli utenti di conservare un registro completo delle loro spedizioni, anche dopo la disconnessione dall'account Amazon.

Connessione ad Internet



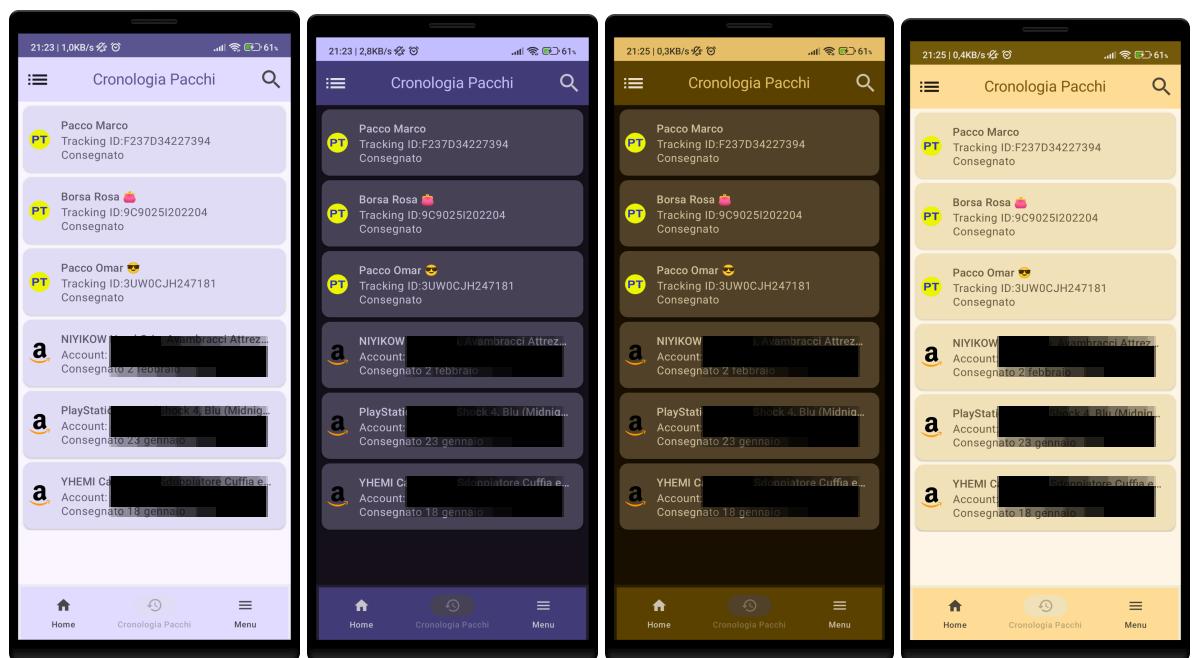
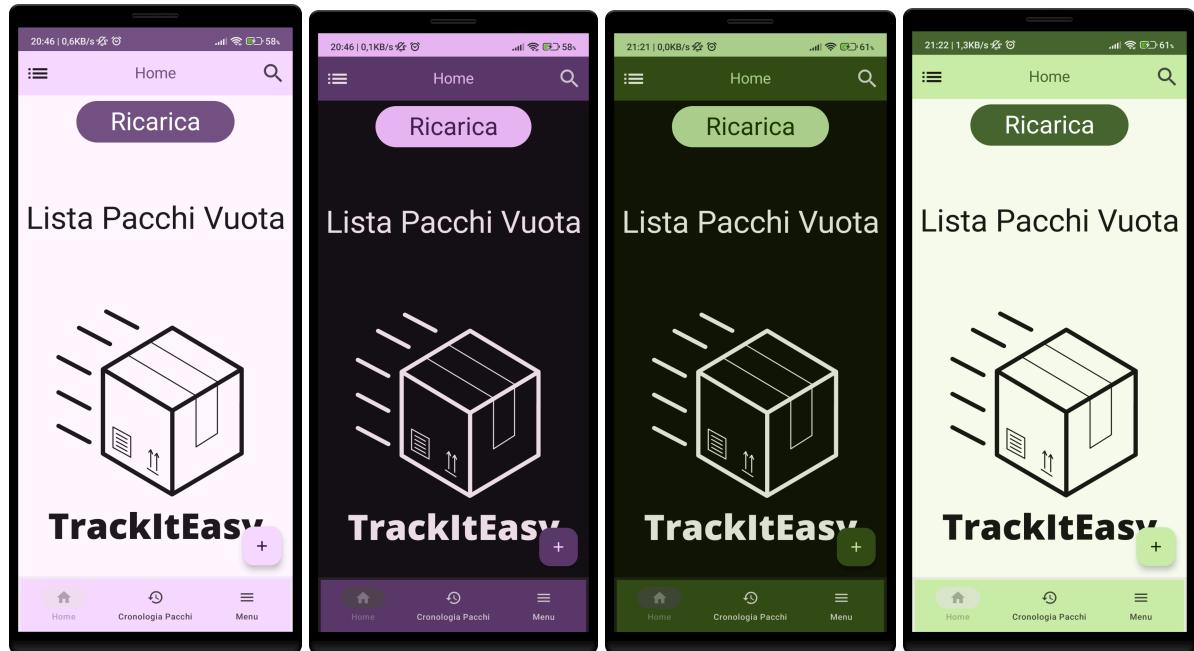
Quando l'utente si trova **senza connessione** a Internet, alcune operazioni all'interno dell'app vengono limitate, poiché richiedono interazioni esterne con un server. In questo contesto, viene visualizzato un **simbolo** che segnala all'utente l'assenza di accesso a Internet, indicando che alcune funzionalità potrebbero essere limitate.

Tuttavia, teniamo in considerazione che gli utenti potrebbero decidere di utilizzare l'applicazione anche in assenza di connessione. Pertanto, è fondamentale gestire questa situazione in modo fluido. In caso di mancanza di connessione, l'applicazione monitora costantemente lo stato della connessione dell'utente e lo notifica chiaramente, in modo che l'utente ne sia consapevole.

Inoltre, considerando che la connessione potrebbe interrompersi in qualsiasi momento durante l'utilizzo dell'app, vengono effettuati controlli durante l'inserimento, l'aggiornamento e ogni altra funzionalità che richiede una connessione. In questo modo, l'applicazione può gestire la situazione in modo appropriato, garantendo un'esperienza utente coerente anche in condizioni di connessione variabili.

Material Design

Utilizzando le componenti del Material Design 3, consentiamo all'utente di personalizzare dinamicamente i colori dell'applicazione secondo le proprie preferenze, offrendo la libertà di scegliere le combinazioni cromatiche desiderate.



Pattern Utilizzati

Model-View-ViewModel (MVVM) pattern:

Il pattern MVVM separa la logica della UI dalla logica dell'applicazione, permettendo una gestione più efficiente e modulare del codice. La View si occupa di visualizzare i dati forniti dal ViewModel e di reagire alle interazioni dell'utente, mentre il ViewModel gestisce lo stato e la logica di business dell'applicazione. Questo approccio favorisce la separazione delle responsabilità e migliora la manutenibilità del codice.

Single-activity architecture:

L'architettura basata su singola attività e più fragment consente di sviluppare un'applicazione con un'unica attività principale e molteplici frammenti per gestire le diverse schermate dell'app. Questo approccio riduce i cambi di contesto e semplifica la gestione dello stato dell'applicazione, migliorando la navigazione e l'esperienza utente complessiva.

Navigation component:

Il componente di navigazione fornisce uno standard per la gestione della navigazione all'interno dell'applicazione, semplificando la definizione e la gestione delle rotte di navigazione. Attraverso la creazione di un grafo di navigazione, è possibile collegare le diverse destinazioni dell'app, facilitando la navigazione sia in avanti che all'indietro, oltre alla navigazione innestata.

State hoisting:

Il concetto di state hoisting permette di gestire lo stato dell'applicazione attraverso la comunicazione tra componenti figlie e componenti genitore. Questo approccio elimina la necessità di passare interfacce o callback alle componenti figlie, garantendo un codice più pulito e compatto. Inoltre, favorisce la gestione centralizzata dello stato, migliorando la coerenza e la manutenibilità del codice.

Dependency Injection

La Dependency Injection (DI) è una tecnica ampiamente utilizzata nella programmazione e particolarmente adatta allo sviluppo di applicazioni Android. Seguendo i principi della DI, si gettano le basi per una buona architettura delle app.

Fornendo questi vantaggi:

- Riusabilità del codice
- Facilità di refactoring
- Semplificazione dei test

Singleton

Il Singleton Pattern è un design pattern creazionale che garantisce che una classe abbia una sola istanza e fornisce un punto globale di accesso a tale istanza. Ciò significa che, indipendentemente da quante volte viene istanziata la classe, viene restituita sempre la stessa istanza.

Chain of Responsibility

Il pattern Chain of Responsibility (Catena di Responsabilità) è un design pattern comportamentale che consente di passare una richiesta lungo una catena di gestori. In questo pattern, più oggetti gestori possono elaborare una richiesta, ognuno con la possibilità di gestirla, trasmetterla a quello successivo nella catena o interrompere il processo di trasmissione.

Usiamo questo pattern per cercare il gestore di un pacco senza far specificare all'utente il gestore del pacco.

Sviluppi futuri

In prospettiva futura, ci impegniamo a trasformare l'applicazione in un unico centro di controllo per monitorare lo stato dei pacchi da diversi spedizionieri in modo semplice ed efficace. Questo obiettivo richiede l'espansione della nostra rete di spedizionieri supportati, includendo aziende come DHL, GLS e BRT, oltre a una migliore visualizzazione delle informazioni dei pacchi.

Oltre agli spedizionieri tradizionali, stiamo lavorando per integrare la possibilità di tracciare pacchi da negozi online come Amazon, e miriamo ad aggiungere altri come AliExpress. Vogliamo fornire agli utenti una visione completa della loro esperienza di spedizione, indipendentemente dall'origine del pacco.

Tra le funzionalità future pianificate, c'è l'introduzione di notifiche per fornire agli utenti aggiornamenti in tempo reale sui loro pacchi. Inoltre, stiamo cercando di migliorare l'integrazione dell'autenticazione utilizzando la libreria AndroidX Credential Manager. Tuttavia, riconosciamo che questa libreria è ancora in una fase di sviluppo e non sono pronte delle guide esaustive, rendendo difficile la sua integrazione con Jetpack Compose.

Infine, ci stiamo concentrando sull'ottimizzazione dell'efficienza del recupero dei pacchi da gestori come Amazon, dove le operazioni possono essere complesse e pesanti. L'obiettivo è garantire un'esperienza utente fluida e reattiva, indipendentemente dalla complessità delle operazioni di recupero dei pacchi da parte dei vari spedizionieri.