

Rapporto sul Progetto di Laboratorio di Programmazione per Sistemi Mobili e Tablet (A.A 2023/2024)



Nome progetto: *Shelfy, la tua libreria a portata di mano!*

Studenti: Riccardo Fusiello (231382), Cesare Lever (218555)

Idea: Realizzare un'applicazione Android che offre a ciascun utente una libreria digitale personale, perfetta per tracciare tutte le letture passate e future. Inoltre, mettiamo a disposizione un vasto catalogo di titoli per stimolare la curiosità e arricchire la cultura di ogni lettore. Infine, vogliamo creare una piattaforma dove scambiare conoscenze, idee e opinioni su qualsiasi libro, rendendo la lettura un'esperienza condivisa e arricchente.

Motivazione per l'App considerata

La creazione di una libreria digitale facile da consultare risponde a diverse esigenze fondamentali degli amanti della lettura. Molti lettori desiderano un modo efficace per tenere traccia dei propri libri, sia quelli già letti che quelli ancora da leggere. Con un'app dedicata, gli utenti possono avere sempre a portata di mano un catalogo aggiornato dei loro libri, facilitando la gestione delle loro letture e delle loro collezioni personali.

L'utilità principale di questa app risiede nella capacità di offrire un mezzo per organizzare e monitorare i propri libri. Gli utenti possono aggiungere nuovi titoli, segnare quelli già letti e ricevere raccomandazioni personalizzate. Inoltre, l'app permette di consultare recensioni e valutazioni di altri lettori, aiutando gli utenti a scoprire nuove letture e a fare scelte informate sui prossimi libri da leggere.

Rispondendo ai bisogni degli appassionati di lettura, l'app stimola la curiosità e l'interesse per nuove opere letterarie. Essa non solo consente agli utenti di esplorare un vasto assortimento di titoli, ma anche di condividere le proprie opinioni e scoperte con una comunità di lettori. Questo aspetto sociale dell'app promuove la condivisione della passione per la lettura, creando uno spazio dove gli utenti possono discutere delle loro letture, scambiare consigli e trovare ispirazione reciproca.

Analisi della concorrenza

La nostra app, Shelfy, si inserisce in un mercato competitivo, dove esistono diverse alternative consolidate, ciascuna con i propri punti di forza e debolezze. Abbiamo analizzato due delle più famose: GoodReads e StoryGraph.

App	Pro	Contro
GoodReads	Ampia community di utenti, database esteso, funzionalità pseudo social, numero di recensioni abbondanti, sfide di lettura	Interfaccia datata, pubblicità invadente, raccomandazioni poco precise, funzionalità limitate
StoryGraph	Interfaccia moderna, raccomandazioni precise, analisi dei dati di lettura, nessuna pubblicità, sfide di lettura	Community più piccola, database meno esteso, funzionalità pseudo social limitate, interfaccia più complessa da capire

Shelfy si propone di creare una community ampia, favorendo l'interazione tra lettori appassionati. Grazie alle potenzialità del database di Google, Shelfy fornisce un catalogo esteso e aggiornato di libri, permettendo agli utenti di accedere a una vasta gamma di titoli, dai bestseller ai testi più di nicchia.

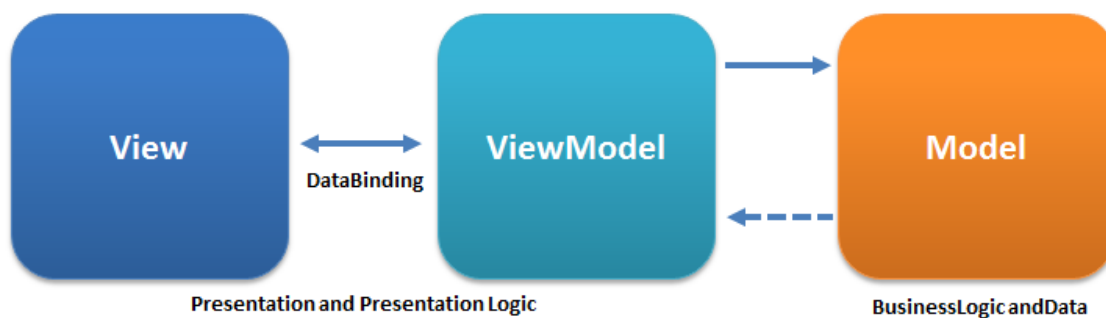
Un altro obiettivo chiave di Shelfy è lo sviluppo di un'interfaccia moderna e semplice da utilizzare, evitando complessità inutili che potrebbero scoraggiare gli utenti. Questa attenzione alla user experience garantirà che anche i nuovi utenti possano navigare e utilizzare tutte le funzionalità dell'app senza difficoltà.

Shelfy aiuta ogni lettore a scoprire sempre nuovi libri che rispecchiano i propri interessi, arricchendo la loro esperienza di lettura e mantenendo alta la loro motivazione e curiosità.

Tecnologie e "salsa speciale"

Abbiamo sviluppato l'App utilizzando il linguaggio di programmazione Kotlin e Jetpack Compose, un framework UI dichiarativo open source basato su Kotlin per Android sviluppato da Google.

Abbiamo adottato l'architettura MVVM (Model-View-ViewModel) per lo sviluppo di Shelfy, un approccio che consente una netta separazione tra le componenti principali dell'app: il front-end e il back-end, corrispondenti rispettivamente alla parte grafica e alla parte logica.



In particolare:

- **View:** si occupa di realizzare il layout che verrà visualizzato dall'utente sfruttando i dati ricevuti dal ViewModel.
- **ViewModel:** funge da intermediario tra il Model e la View. Gestisce i dati dell'interfaccia utente, fornendo i dati dal Model alla View in modo che siano presentati correttamente.
- **Model:** si occupa di gestire i dati dell'app e la business logic, restituendo i risultati al ViewModel.

Questa architettura comporta diversi vantaggi:

- **"Separation of Concerns":** con MVVM, Shelfy suddivide le sue componenti in modo che ciascuna si occupi di specifiche operazioni. Questa suddivisione di compiti rende l'architettura chiara e gestibile, evitando sovraccarichi e disordine nel codice.
- **Facilità di manutenzione:** con componenti ben definite e indipendenti, come nel caso di Shelfy, diventa più semplice aggiornare o modificare una componente specifica senza influenzare le altre.
- **Facilità nel testing:** grazie alla separazione delle componenti, il testing può essere mirato a una specifica componente.

- **Riusabilità del codice:** una volta implementato un elemento di una componente, può essere riutilizzato più volte. Questo non solo rende più efficiente lo sviluppo di Shelfy, ma consente anche di creare un codice modulare e riutilizzabile che può essere utilizzato anche in altri contesti o progetti futuri.

Per quanto riguarda il catalogo dei libri, ci siamo affidati alle API di Google, ovvero Google Books API. Queste API forniscono un vasto catalogo di libri, con una serie di informazioni molto utili, in particolare:

- **id:** id del libro
- **title:** titolo del libro
- **authors:** autore o autori del libro
- **description:** breve descrizione del libro, contenente molto spesso trama o simili
- **imageLinks:** diversi link contenenti le copertine del libro in vari formati (smallThumbnail, thumbnail, small, medium, large)

Abbiamo utilizzato due endpoint forniti da Google, i seguenti:

- <https://www.googleapis.com/books/v1/volumes?q=query+di+ricerca>
 - Questo endpoint permette di ottenere le informazioni di una lista di libri, che soddisfano la query di ricerca secondo qualche criterio deciso da Google
- <https://www.googleapis.com/books/v1/volumes/idBook>
 - Questo endpoint permette di ottenere le informazioni di uno specifico libro, in base al suo id

Per usufruire dei dati di Google Books, abbiamo utilizzato la libreria Retrofit di Kotlin. Ecco alcuni motivi per cui questa decisione è vantaggiosa:

1. **Facilità di integrazione:** Retrofit è una libreria ampiamente utilizzata per effettuare chiamate HTTP in Android. La sua sintassi è semplice e intuitiva.
2. **Gestione delle richieste HTTP:** Retrofit semplifica la gestione delle richieste HTTP, consentendo di definire in modo chiaro le API e i servizi web da utilizzare per recuperare i dati di Google Books. Questo riduce la complessità del codice e semplifica il processo di recupero e gestione dei dati.
3. **Interoperabilità con Compose:** l'interoperabilità con Retrofit di Jetpack Compose consente di integrare facilmente i dati recuperati da Google Books nell'interfaccia utente di Shelfy.

Oltre i dati forniti dalle API di Google, abbiamo deciso di affiancare anche dei dati relativi alla nostra app, in particolare per gestire gli utenti, le recensioni, le readlist e le note. Nel dettaglio:

- **User:** data class di Kotlin, utile a rappresentare un utente. In particolare memorizziamo email, password, username e uid, id ottenuto da Firebase Authentication (vedi parte Firebase)

```
data class User(  
  
    val username : String, // univoci per utente  
  
    val email : String, // univoci per utente  
  
    val password : String,  
  
    val uid : String  
)
```

- **Note:** data class di Kotlin, utile a rappresentare una nota di un utente riguardo un certo libro. In particolare memorizziamo bookId per risalire al libro specifico, userId per risalire all'utente che ha creato la nota e text riguardante il testo della nota.

```
data class Note(  
  
    val userId: String,  
  
    val text: String,  
  
    val bookId: String  
)
```

- **Readlist:** data class di Kotlin, utile a rappresentare una readlist ovvero una collezione di libri. In particolare memorizziamo il name della readlist, userId per risalire all'utente della readlist e infine il content ovvero la lista di libri effettiva.

```
data class Readlist (  
  
    val name : String,  
  
    val userId : String,  
  
    val content : List<Item>  
)
```

- **Review:** data class di Kotlin, utile a rappresentare una recensione di un utente riguardo un certo libro. In particolare memorizziamo bookId per risalire al libro specifico, username per

risalire all'utente che ha creato la recensione, desc ovvero il testo (facoltativo) della recensione e infine stars ovvero il voto che è stato dato al libro (0...5 stelle)

```
data class Review(  
    val bookId : String,  
    val username : String,  
    val stars : Int,  
    val desc : String  
)
```

Per quanto riguarda la parte di autenticazione e storage dei dati della nostra app, abbiamo utilizzato i servizi forniti da Firebase, in particolare: Firebase Authentication per la parte di autenticazione e Firebase Firestore per la parte di storage.

Ecco alcuni dei benefici di questa scelta:

1. **Sicurezza:** Firebase Authentication offre un sistema di autenticazione robusto e altamente sicuro, consentendo agli utenti di accedere all'app in modo protetto tramite varie opzioni di login, come email/password, account Google, account Facebook e molto altro ancora. Ciò garantisce che solo gli utenti autorizzati possano accedere ai dati sensibili dell'app.
2. **Facilità di integrazione:** Firebase Authentication è facile da integrare, grazie alla sua documentazione chiara e ai numerosi SDK disponibili per diverse piattaforme (Android, iOS, Web, etc.). Ciò consente di implementare rapidamente e senza problemi un sistema di autenticazione completo all'interno dell'app.
3. **Scalabilità:** Firebase Firestore fornisce un database cloud NoSQL completamente gestito, che offre scalabilità automatica e prestazioni elevate. Ciò significa che Shelfy può teoricamente ..gestire senza problemi grandi quantità di dati e aumentare la sua base di utenti senza doversi preoccupare di problemi di scalabilità del database.
4. **Real-time database:** Firestore supporta il modello di database in tempo reale, consentendo agli utenti di Shelfy di accedere e aggiornare i dati in tempo reale su diverse piattaforme.

Le pagine dell'App che abbiamo implementato sono:

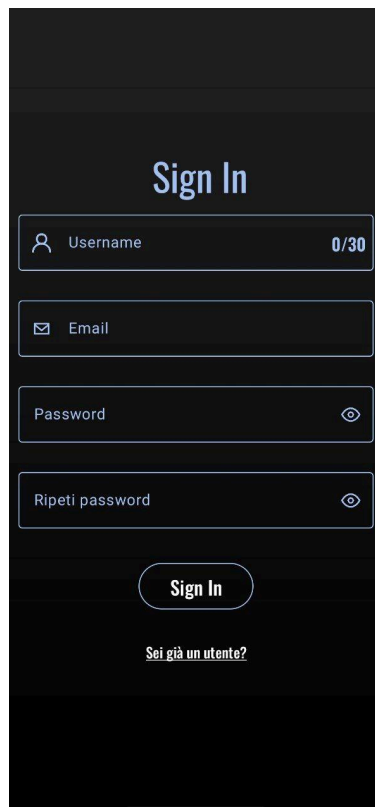
- Sign In Page
- Login Page
- Offline Page
- Home Page
- Search Page
- Book Page
- Profile Page

Sign In Page

In questa pagina è possibile effettuare la registrazione di un nuovo utente, sono presenti i vari campi richiesti sotto forma di TextField. Sono presenti dei controlli dei dati inseriti, per verificare ad esempio che la email sia del formato richiesto, la password sia lunga 8 caratteri, ecc.

Una volta premuto il pulsante “Sign In”, viene eseguita la funzione “signInUser” del ViewModel, e se i dati inseriti sono corretti effettua il sign in mediante Firebase Authentication, crea un nuovo utente nel nostro database ed esegue il login, indirizzando l'utente alla home page.

Nel caso in cui l'utente sia già registrato, è possibile arrivare alla pagina di Login, tramite il pulsante “Sei già registrato?”



Sign In

Username 0/30

Email

Password

Ripeti password

Sign In

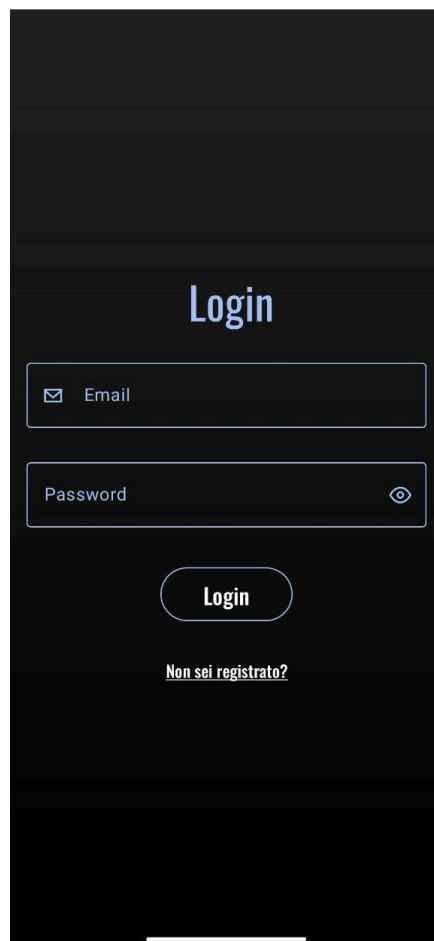
[Sei già un utente?](#)

Login Page

In questa pagina è possibile effettuare il login di un utente, inserendo email e password. Una volta premuto il pulsante “Login”, viene eseguita la funzione “login” del ViewModel che controlla se i dati inseriti corrispondono a un utente nel nostro database. In tal caso, viene eseguita l’autenticazione. Sono presenti dei controlli nel caso in cui i dati inseriti non siano corretti, mostrando un testo di errore.

E’ presente anche una funzione crittografica che cripta la password tramite SHA256, salvando nel database la password criptata.

Nel caso in cui l’utente non sia già registrato, è possibile arrivare alla pagina di SignIn, tramite il pulsante “Non sei registrato?”

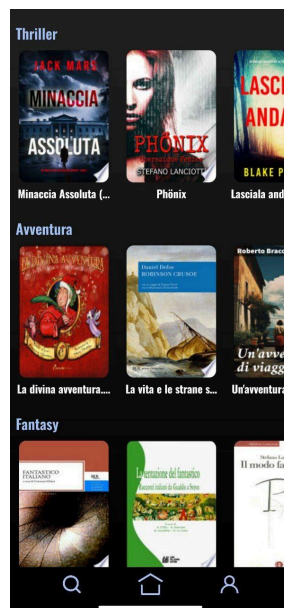


Offline Page

E’ una pagina che viene mostrata solamente quando l’utente apre l’app senza una connessione ad Internet. Nel caso si ritorna ad essere connessi, basta cliccare su Riprova per tornare alla pagina online

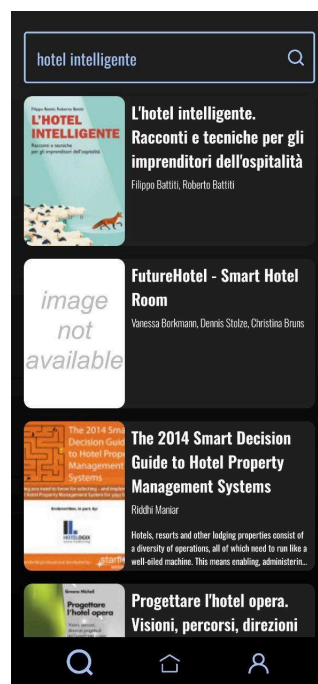
Home Page

E' la pagina principale della nostra applicazione. E' accessibile solamente se l'utente è stato correttamente autenticato, e comprende i suggerimenti consigliati dall'app in base ai generi. E' possibile visualizzare un determinato libro, trasferendosi alla pagina "Book Page" cliccando su una copertina. (Nota: attualmente le raccomandazioni sono statiche)



Search Page

In questa pagina è possibile effettuare la ricerca di un libro, attraverso una certa query che verrà utilizzata nella query di ricerca di Google Books API. Successivamente alla richiesta, viene mostrata una lista di risultati verticale, e per ciascun libro è possibile selezionarlo trasferendosi alla pagina "Book Page"

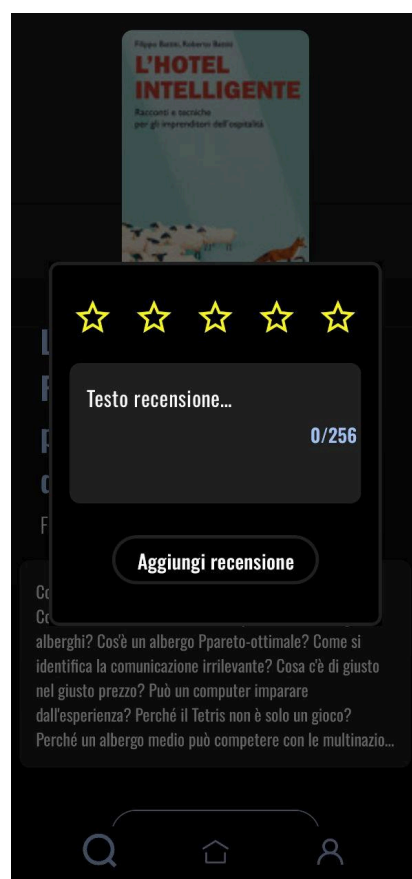
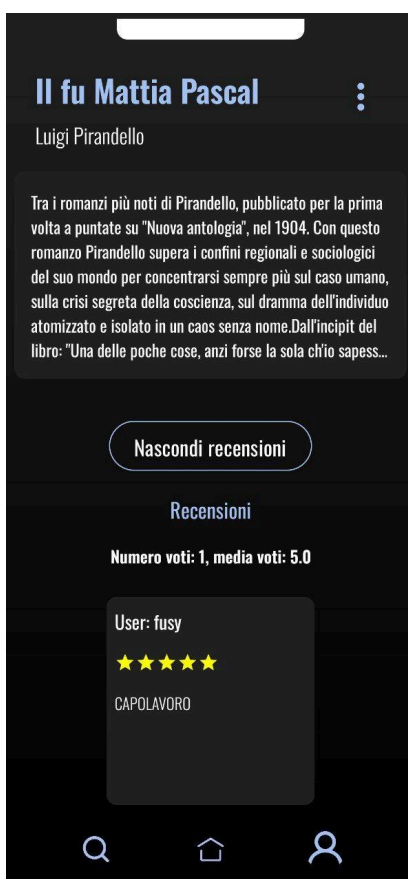


Book Page

Questa pagina mostra tutte le informazioni relative a un libro, in particolare titolo, autori e trama (in versione compatta o estesa). E' possibile compiere delle operazioni premendo il pulsante con i 3 punti verticali, in particolare:

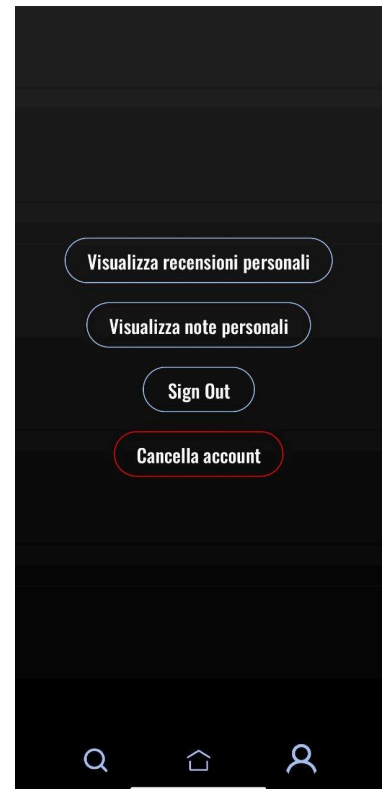
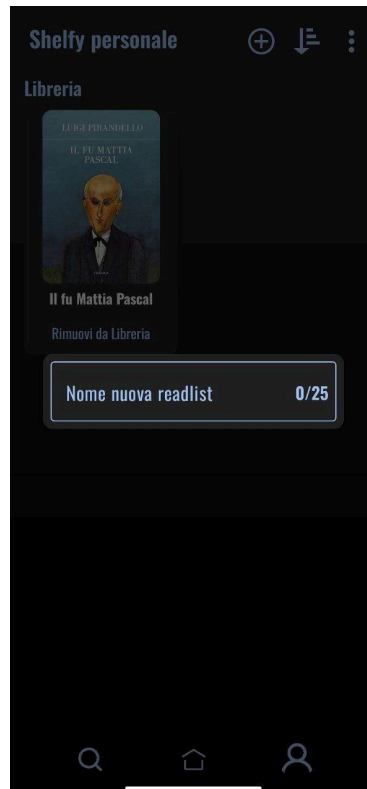
- Condividere il libro
- Aggiungere o visualizzare una nota (se già presente una nota, viene sovrascritta)
- Aggiungerlo alla libreria
- Aggiungerlo a una readlist
- Aggiungere una recensione (se già presente una recensione, viene sovrascritta)

Sono presenti anche le recensioni, nascoste o visualizzate, relative al libro da parte degli utenti, con il numero totale di recensioni e la media voti.



Profile Page

Questa pagina mostra tutto ciò che riguarda il profilo personale dell'utente. Mostra tutte le readlist da lui create e la libreria. E' possibile aggiungere o cancellare readlist (non è possibile avere readlist con nomi uguali), e aggiungere o rimuovere libri dalle readlist. Le readlist si possono ordinare in ordine alfabetico crescente o decrescente. Inoltre, si può effettuare il logout e ritornare alla pagina di login.



Nota: tutte le funzionalità con operazioni fatte online (chiamate a Google Books API, operazioni CRUD al database Firebase, ecc...) sono vietate se l'utente non è connesso alla rete, in questo caso verrà mostrata una finestra di dialogo in cui viene notificato il seguente divieto. Inoltre, non abbiamo implementato la pagina di “Visualizza recensioni personali” e “Visualizza note personali”.

Primi risultati ottenuti nei test

Abbiamo condotto dei crash test sulla nostra applicazione per verificare la robustezza delle pagine che richiedono input dell'utente. Abbiamo testato l'applicazione con vari tipi di input, inclusi caratteri speciali, stringhe vuote, stringhe corte ed eccessivamente lunghe, per assicurarci che non ci fossero crash; per ovviare a questi problemi abbiamo limitato le lunghezze delle stringhe che l'utente può scrivere. I problemi riscontrati sono stati risolti, con particolare attenzione alle pagine di Login, Sign In, Search Page, Profile Page e Book Page.

La reattività del sistema è buona: le funzioni online di Firebase e Google Books vengono richiamate senza causare rallentamenti significativi. Gli eventuali rallentamenti sono per lo più dovuti al rendering grafico delle informazioni, che dipende dalla velocità della connessione alla rete dell'utente.

Parametri di qualità (percepiti dai tester della nostra app):

- **Correttezza:** il software funziona come previsto e tutte le operazioni mostrate all'utente sono effettuate correttamente
- **Affidabilità:** purtroppo non siamo riusciti a creare un traffico elevato, tuttavia tutte le operazioni effettuate dagli utenti si sono risolte in tempi costanti e senza errori.
- **Robustezza:** abbiamo cercato di controllare tutti gli input mandati dall'utente in maniera accurata, in modo tale da evitare situazioni sgradevoli, come rendering sbagliato, ecc...
- **Efficienza:** il software non effettua operazioni super complesse, perciò risulta essere abbastanza efficiente e senza usare eccessivamente le risorse dello smartphone
- **Usabilità:** il software, a detta dei tester, è risultato facile da usare, anche dopo qualche minuto di utilizzo
- **Scalabilità:** non abbiamo potuto certificare la scalabilità del software a causa dei pochi utenti attualmente presenti sulla piattaforma
- **Sicurezza:** il software protegge in maniera sicura i dati degli utenti attraverso l'architettura Firebase

Purtroppo non siamo riusciti ad implementare la parte "community" dell'app per via della mancanza di tempo a disposizione, però l'intenzione di crearla continua ad esserci da parte nostra.

Nota: riguardo la cancellazione dell'account, non è stata perfezionata al 100%, perciò delle volte ha dei bug che non siamo riusciti a risolvere con Firebase Authentication.