

READING STATS

RELAZIONE DI PROGRAMMAZIONE MOBILE

Cristian Di Cintio - S1110150
Federico Di Giovannangelo - S00000000

Contents

1	Introduzione	2
2	Progettazione	2
2.1	Requisiti	2
2.1.1	Requisiti funzionali	3
2.1.2	Requisiti non funzionali	5
2.2	Casi d'uso	6
2.3	Gestione Database - Firebase	10
2.4	API utilizzate - Google Books	11
2.5	UI - Interfaccia applicazione	11
3	Programmazione in Android	11
3.1	MVVM	11
3.2	Struttura cartelle applicazione	11
4	Programmazione in Flutter	11
5	Errori e possibili bug	11

1 Introduzione

L'applicazione sviluppata ha come obiettivo la memorizzazione delle statistiche di libri in lettura dell'utente. Tali statistiche riguardano la percentuale di completamento del libro e il tempo impiegato, memorizzato per ciascuno libro.

L'utente per usufruire delle funzionalità dell'app deve registrarsi con nome, cognome, email, username e password. Per lo username e l'email vengono effettuati dei controlli per verificare l'univocità e non permettere la registrazione di utenti con medesimo username o email.

In seguito l'utente può effettuare l'accesso con la propria email e password.

L'utente può ricercare i libri in un catalogo tramite la digitazione del titolo in una barra di ricerca o scansionando, con un pulsante apposito, il codice a barre di un libro fisico. In seguito può decidere di aggiungerlo in tre liste distinte, "da leggere", "in lettura" e "letti", per avere un resoconto delle pagine lette e del tempo impiegato nella lettura dei libri salvati. Il catalogo permette anche il filtraggio di libri tramite la selezione di categorie predefinite.

Al momento della lettura l'utente può avviare un timer per un libro in lettura, presente nell'omonima lista, e disattivarlo al termine della sessione.

L'utente può visualizzare e modificare i propri dati personali e interagire con altri utenti, visualizzando principalmente quali libri hanno salvato e quali stanno leggendo, ognuno con le loro rispettive metriche.

L'intera applicazione è stata sviluppata inizialmente in Kotlin per ambiente Android e in seguito in Flutter per un funzionamento multiplatforma, in modo tale da permettere una maggiore compatibilità e una migliore interazione tra utenti con diversi dispositivi mobile.

Lo sviluppo in Kotlin è stato applicato con Jetpack Compose, per permettere uno sviluppo moderno e in linea con le attuali applicazioni sul mercato, con una struttura architetturale basata sul Model-View-ViewModel (MVVM) garantendo una composizione comprensibile per lo sviluppo dell'app.

2 Progettazione

2.1 Requisiti

Si riportano di seguito i requisiti *funzionali* e *non funzionali*

2.1.1 Requisiti funzionali

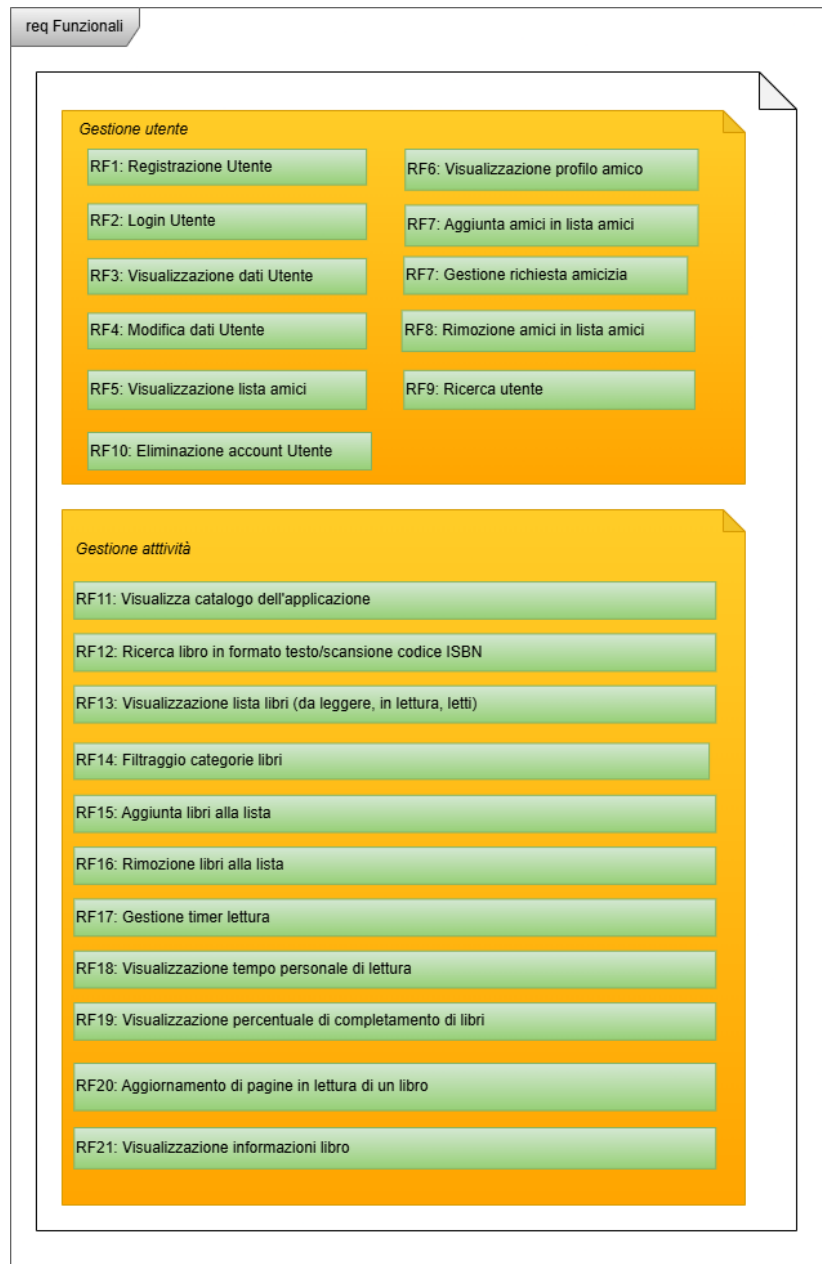


Figure 1: Requisiti funzionali

- *Gestione utente*

- *RF1: Registrazione utente* - L'applicazione permette di registrarsi con le proprie informazioni personali
- *RF2: Login utente* - L'applicazione permette di autenticare l'utente registrato con le proprie informazioni personali
- *RF3: Visualizzazione dati utente* - L'applicazione permette di visualizzare i propri dati personali
- *RF4: Modifica dati utente* - L'applicazione permette di modificare i propri dati personali
- *RF5: Visualizzazione lista amici* - L'applicazione permette di visualizzare la lista degli amici che hanno accettato la sua richiesta d'amicizia
- *RF6: Visualizzazione profilo amico* - L'applicazione permette di visualizzare il profilo di un suo amico nella lista
- *RF7: Aggiunta amici in lista amici* - L'applicazione permette di aggiungere un amico nella sua lista inviando una richiesta
- *RF8: Gestione richiesta di amicizia* - L'utente può decidere se accettare o rifiutare una richiesta di amicizia
- *RF9: Rimozione amici da lista amici* - L'applicazione permette di rimuovere un amico dalla lista amici
- *RF10: Ricerca utente* - L'applicazione permette di ricercare un utente digitando il suo username o nome completo
- *RF11: Eliminazione account utente* - L'applicazione permette all'utente di cancellare il proprio profilo

- *Gestione attività*

- *RF12: Visualizza catalogo dell'applicazione* - L'applicazione permette di visualizzare il catalogo dei libri distinti in categorie
- *RF13: Ricerca libro in formato testo o tramite scansione codice a barre* - L'applicazione permette la ricerca del libro digitando testualmente il titolo in una casella di ricerca apposita oppure scansionando il codice a barre di un libro fisico
- *RF14: Visualizzazione lista libri (da leggere, in lettura, letti)* - L'applicazione permette di visualizzare il contenuto delle liste principali formate dai libri salvati dall'utente
- *RF15: Filtraggio categorie libri* - L'applicazione permette all'utente di filtrare le categorie dei libri da visualizzare nel catalogo
- *RF16: Aggiunta libri alla lista* - L'applicazione permette all'utente di aggiungere i libri in una delle apposite liste fornite dall'app
- *RF17: Rimozione libri dalla lista* - L'applicazione permette la rimozione di libri dalle liste

- *RF18: Gestione timer lettura* - L'applicazione permette di tenere sotto controllo il tempo di lettura di un libro con l'utilizzo di un timer dedicato attivabile e disattivabile dall'utente
- *RF19: Visualizzazione tempo personale di lettura* - L'applicazione permette all'utente di visualizzare per ogni singolo libro il tempo di lettura impiegato con l'utilizzo del timer
- *RF20: Visualizzazione percentuale completamento di libri* - L'applicazione permette di visualizzare la percentuale di completamento di ogni singolo libro, calcolata tramite il numero di pagine lette dall'utente rispetto al numero di pagine totali di un determinato libro
- *RF21: Aggiornamento percentuale di pagine in lettura di un libro* - L'applicazione permette di aggiornare il numero di pagine lette di un libro per calcolare la percentuale di completamento
- *RF22: Visualizzazione informazioni libro* - L'applicazione permette la visualizzazione delle info di un determinato libro

2.1.2 Requisiti non funzionali

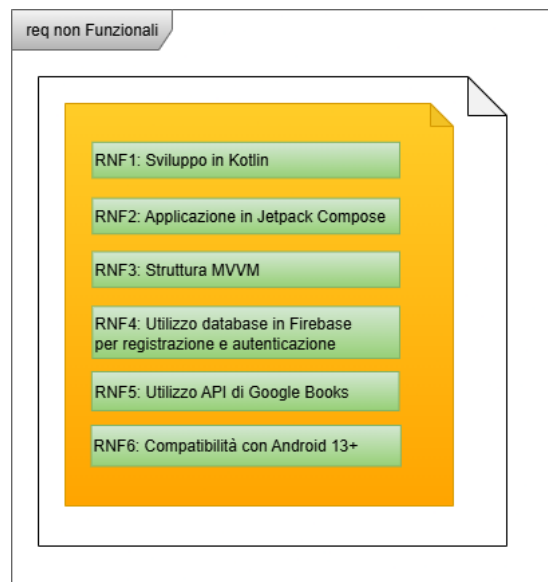


Figure 2: Requisiti non funzionali

- *RNF1: Sviluppo in Kotlin* - L'applicazione è stata sviluppata principalmente con il linguaggio di programmazione Kotlin
- *RNF2: Applicazione in Jetpack Compose* - L'applicazione è stata realizzata con Jetpack Compose per realizzare le interfacce grafiche

- *RNF3: Struttura MVVM* - L'applicazione ha una struttura architeturale basata su Model-View-ViewModel per agevolare la programmazione con sezioni dedicate
- *RNF4: Firebase e Firestore per memorizzazione e accesso/autenticazione utente* - L'applicazione ha come base per la memorizzazione di dati e la gestione della registrazione e l'accesso dell'utente un database tramite la piattaforma Firebase
- *RNF5: API di Google Books* - L'applicazione si sincronizza e fa utilizzo delle API Key di Google Books per mostrare i libri nel catalogo e nella ricerca
- *RNF6: compatibilità con Android 13+* - L'applicazione ha una compatibilità con il livello di API 33, permettendone l'utilizzo su android 13 e oltre

2.2 Casi d'uso

Si riportano gli attori e i tre casi d'uso principali che descrivono le azioni principali che si possono svolgere nell'applicazione a partire dai requisiti elencati in precedenza.

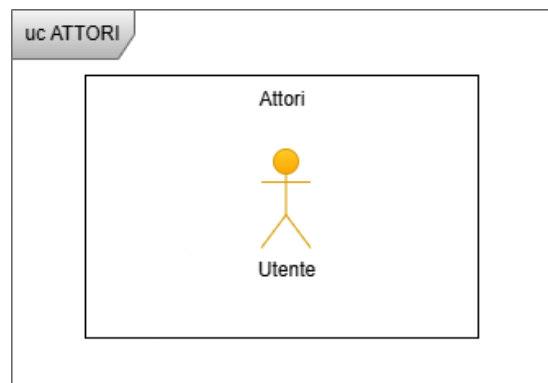


Figure 3: Attori principali

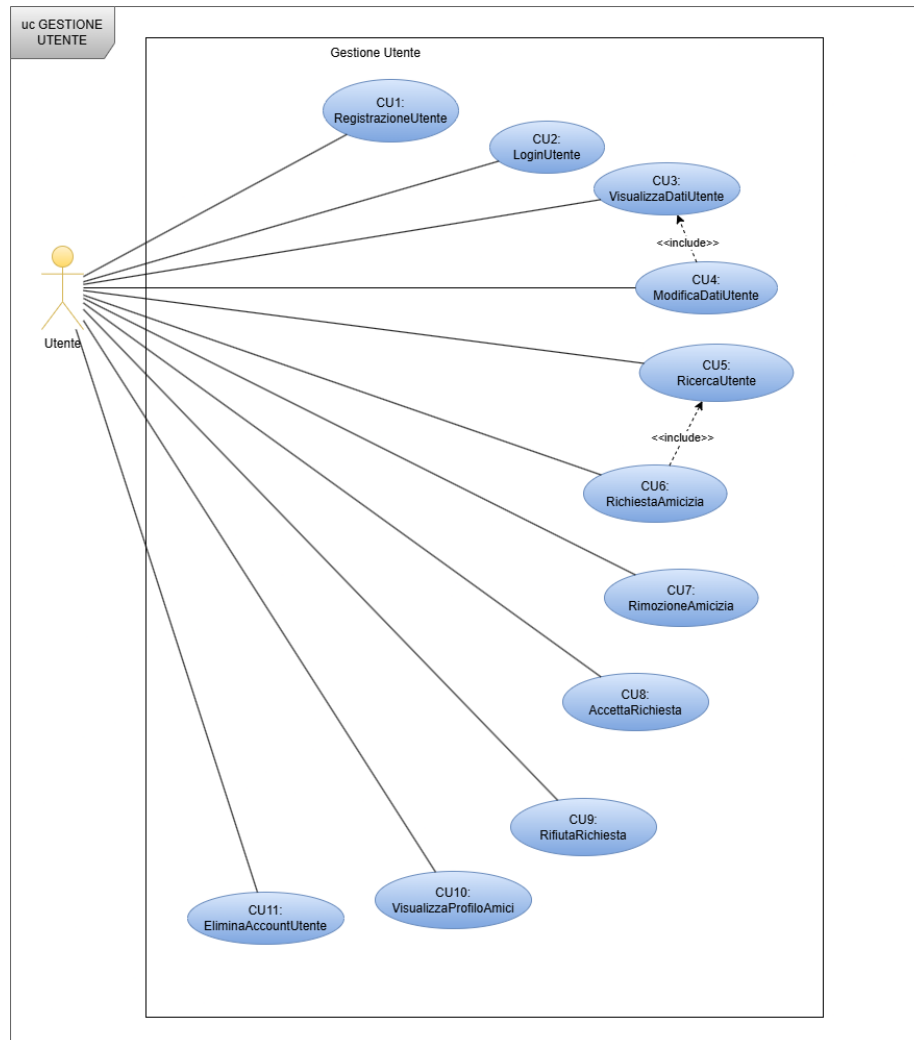


Figure 4: Gestione utente

- *CU1: RegistrazioneUtente* - L'utente, per usufruire dei servizi offerti dall'applicazione, ha bisogno di registrarsi con le proprie informazioni personali
- *CU2: LoginUtente* - L'utente può accedere ai servizi forniti inserendo le proprie credenziali (email e password) precedentemente utilizzate per registrarsi
- *CU3: VisualizzaDatiUtente* - L'utente può visualizzare le proprie informazioni personali in una sezione dedicata
- *CU4: ModificaDatiUtente* - L'utente può modificare i propri dati personali

- *CU5: RicercaUtente* - L'utente può ricercare un utente digitando il suo nome utente (username) o nome completo (nome e/o cognome)
- *CU6: RichiestaAmicizia* - L'utente può aggiungere amici nella propria lista inviando richieste ad altri utenti
- *CU7: RimozioneAmicizia* - L'utente può rimuovere gli amici dalla propria lista
- *CU8: AccettaRichiesta* - L'utente può accettare la richiesta di amicizia da un altro utente che l'ha inviata
- *CU9: RifiutaRichiesta* - L'utente può rifiutare la richiesta di amicizia da un altro utente che l'ha inviata
- *CU10: VisualizzaProfiloAmici* - L'utente può visualizzare il profilo dei propri amici con le loro statistiche di lettura
- *CU11: EliminaAccountUtente* - L'utente può eliminare il proprio account utente

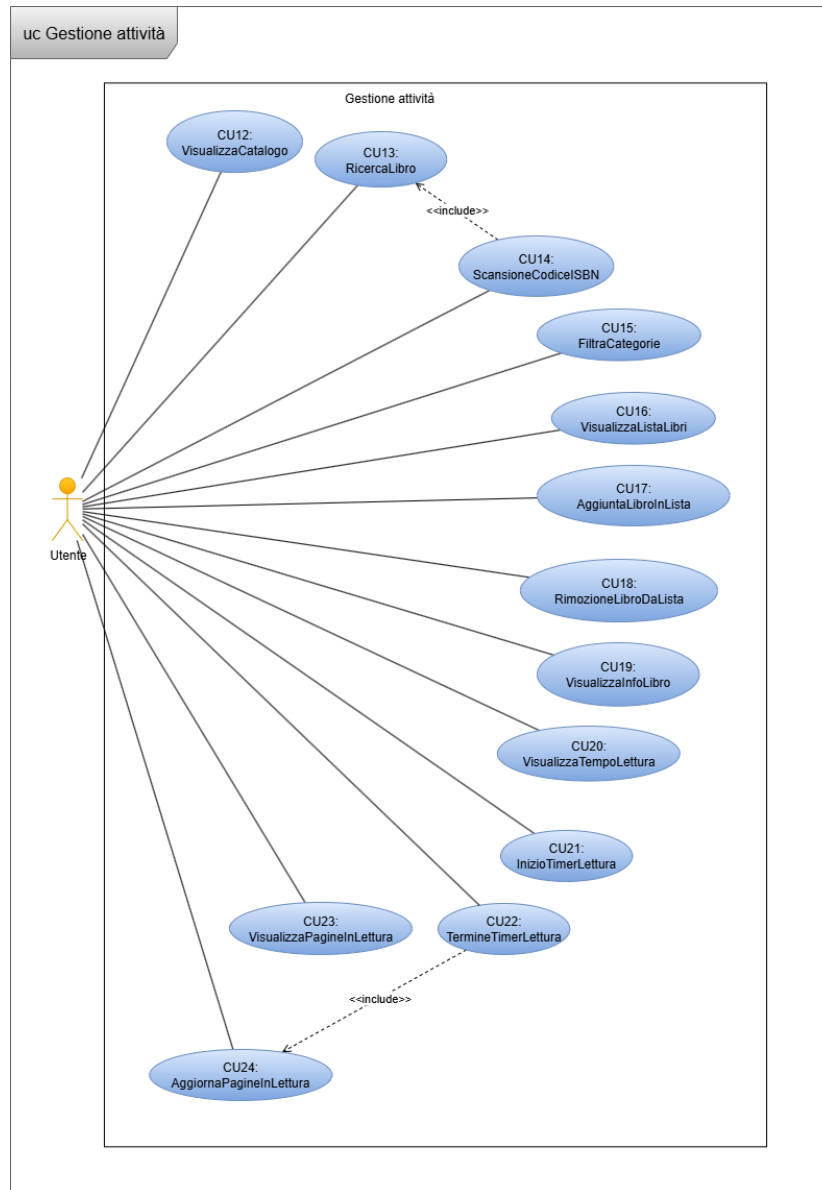


Figure 5: Gestione attività

- *CU12: VisualizzaCatalogo* - L'utente può visualizzare il catalogo di libri separato in categorie
- *CU13: RicercaLibro* - L'utente può ricercare il libro digitando il titolo nella barra di ricerca presente nel catalogo

- *CU14: ScansioneCodiceISBN* - L'utente può effettuare la scansione del codice ISBN di un libro fisico tramite la fotocamera del telefono
- *CU15: FiltroCategorie* - L'utente può filtrare le categorie del catalogo selezionandole da un lista con checkbox
- *CU16: VisualizzaListaLibri* - L'utente può visualizzare una delle tre liste principali (da leggere, in lettura, letti) per memorizzare i libri salvati
- *CU17: AggiuntaLibroInLista* - L'utente può aggiungere un libro selezionato in una delle tre liste principali (da leggere, in lettura, letti)
- *CU18: RimozioneLibroDaLista* - L'utente può rimuovere un libro presente in una delle tre liste principali (da leggere, in lettura, letti)
- *CU19: VisualizzaInfoLibro* - L'utente può visualizzare le informazioni principali che rappresentano il libro selezionandolo dal catalogo o dalle liste
- *CU20: VisualizzaTempoLettura* - L'utente può visualizzare il tempo di lettura di un proprio libro
- *CU21: InizioTimerLettura* - L'utente può iniziare la sessione di lettura di un libro tramite l'inizio di un timer dedicato
- *CU22: TermineTimerLettura* - L'utente può terminare la sessione di lettura di un libro permettendo di fermare il timer
- *CU23: VisualizzaPagineInLettura* - L'utente può visualizzare le pagine in lettura di un libro salvato nell'omonima lista
- *CU24: AggiornaPagineInLettura* - L'utente può aggiornare le pagine

2.3 Gestione Database - Firebase

In presenza di un sistema di autenticazione e registrazione a un servizio online, con la possibilità per l'utente di interagire con altri utenti, e permettere il salvataggio personale di libri nel proprio profilo, l'infrastruttura migliore per manipolare questi dati risulta essere il database gestionale Firestore di Google Firebase.

Firestore è un DB NoSQL strutturato a documenti, il quale permette il salvataggio di dati in documenti raggruppati in collezioni e può contenere sotto-collezioni e campi annidati. Tali documenti e collezioni vengono generati con la scrittura di dati tramite codice.

Il suo utilizzo si concentra su due componenti della sezione *Creazione*:

- *Authentication*: permette di gestire il metodo di accesso degli utenti tramite diversi servizi. Nel caso di questo progetto abbiamo implementato esclusivamente l'autenticazione semplice tramite email e password.

- *Firestore Database*: permette la memorizzazione tramite documenti e collezioni descritti in precedenza dei dati dell'utente, dei libri salvati e delle richieste di amicizia inviate tra utenti. Sono state implementate delle regole specifiche per permettere un corretto e coerente accesso ai dati e la loro relativa modifica.

2.4 API utilizzate - Google Books

Le API di Google Books sono state utilizzate per ottenere le informazioni principali dei libri, in modo da permettere all'utente la visualizzazione di tali informazioni e il conseguente salvataggio. L'API viene fornita tramite un link per le ricerche e una chiave generata da Google Cloud per permetterne l'abilitazione e la limitazione del tipo di dispositivo che la utilizzerà.

Link: <https://www.googleapis.com/books/v1/volumes?q=search-terms>.

"q" è il parametro da inserire per effettuare la ricerca tramite titolo o autore

2.5 UI - Interfaccia applicazione

3 Programmazione in Android

3.1 MVVM

3.2 Struttura cartelle applicazione

4 Programmazione in Flutter

5 Errori e possibili bug