

Corso di Applicazioni web: progettazione e sviluppo

Book tracker – sviluppo di una Single Page Application
utilizzando la libreria React



Indice

1.0 Introduzione	3
1.1 Funzionalità principali	3
1.2 Stack tecnologico	3
2.0 Architettura del software	5
2.1 Model	5
2.2 View	5
2.3 View model	6
2.4 Service	7
3.0 Conclusioni	8
3.1 Eventuali sviluppi futuri	8

1.0 Introduzione

Obiettivo principale del progetto è lo sviluppo di una Single Page application mediante l'utilizzo della libreria Java Script React che permetta all'utente di ricercare libri tramite titolo, autore o ISBN e, in seguito ad un eventuale login aggiungere o togliere libri dalla propria libreria personale.

1.1 Funzionalità principali

L'applicazione offre quattro funzionalità principali:

- Ricerca: attraverso l'integrazione con servizi esterni, l'utente può cercare libri filtrando per titolo, autore o codice ISBN;
- Visualizzazione dettagli libro: ogni libro trovato dispone di una scheda dedicata che mostra la copertina, gli autori, l'ISBN e una descrizione estesa del contenuto;
- Gestione della propria libreria: gli utenti registrati possono aggiungere libri alla propria collezione mediante un apposito pulsante e rimuoverli in qualsiasi momento;
- Sistema di Autenticazione: l'accesso alle funzioni di salvataggio è protetto da un sistema di registrazione ed accesso che garantisce la persistenza dei dati personali per ciascun utente.

1.2 Stack tecnologico

Per l'interfaccia è stato utilizzato React, una libreria che permette di costruire la UI attraverso componenti riutilizzabili e reattivi, facilitando la gestione dello stato dell'applicazione. Per la navigazione tra le diverse sezioni, è stato implementato React Router Dom, così da avere una Single Page Application capace di cambiare vista istantaneamente senza ricaricare il browser.

Per quanto riguarda l'autenticazione è stato usato Firebase Authentication per la gestione dei profili utente, mentre Firestore datastore funge da database NoSQL per memorizzare le collezioni personali. Il reperimento delle informazioni bibliografiche avviene tramite l'integrazione con la Google Books API, che permette ricerche mirate e restituisce metadati dettagliati per ogni volume.

Infine, per l'aspetto estetico, è stato utilizzato il framework Bootstrap tramite la libreria Reactstrap, che fornisce componenti pronti all'uso e coerenti. Le

Lorenzo Colombo 885895

personalizzazioni grafiche avanzate, i colori e le animazioni sono state invece definite in un file CSS esterno.

2.0 Architettura del software

Per l'architettura della soluzione è stato scelto un pattern Model View View-Model.

2.1 Model

I Model rappresentano la struttura dei dati fondamentali dell'applicazione:

- **BookModel:** È la classe che definisce l'entità libro all'interno dell'app. Il metodo statico `fromGoogleApi(item)` si occupa di normalizzare i dati grezzi provenienti dalla Google Books API, gestendo eventuali campi mancanti come le copertine o gli autori e assegnando valori di fallback;
- **UserModel:** Rappresenta l'utente autenticato. Include il metodo `fromFirebase(firebaseUser)` che mappa l'oggetto utente di Firebase nel modello interno, associandovi un'istanza della libreria personale;
- **LibraryModel.js:** Gestisce la logica della collezione. Include il metodo `hasBook(bookId)`, utilizzato dalle View per determinare se il pulsante di un libro debba mostrare l'opzione "Aggiungi" o "Rimuovi".

2.2 View

Le View rappresentano lo strato di presentazione. Il loro compito è esclusivamente quello di renderizzare i dati forniti dai ViewModel e catturare le interazioni dell'utente.

Di seguito vengono descritte nel dettaglio:

- **App:** rappresenta il componente radice della parte visuale e funge da contenitore per l'intera applicazione. Al suo interno viene definita la struttura portante della UI, inclusa la Navbar che rimane persistente durante la navigazione e cambia dinamicamente i propri elementi in base allo stato dell'utente fornito dall'AuthViewModel. Utilizza il componente Routes di React Router Dom per mappare gli indirizzi URL ai rispettivi componenti, garantendo che l'utente visualizzi sempre il contenuto corretto in base al percorso selezionato;
- **Home:** integra un form di ricerca con filtri sulla tipologia di ricerca e coordina la visualizzazione dei risultati. Implementa la logica di scroll infinito tramite un IntersectionObserver collegato all'ultimo elemento della lista;

- `BookDetailPage`: vista di dettaglio che presenta tutte le informazioni di un libro, inclusa la descrizione testuale formattata, in quanto, in alcuni casi viene fornita dalle Google books in formato HTML. Se l'utente è autenticato, la pagina mostra dinamicamente un pulsante per aggiungere o rimuovere il libro dalla propria collezione, basandosi sullo stato fornito dal `LibraryViewModel`;
- `LibraryPage`: visualizza la collezione privata dell'utente. Offre un controllo sullo stato dell'autenticazione: se l'utente non è loggato, mostra un messaggio di invito all'accesso, in caso contrario, visualizza i libri salvati permettendo di scegliere il layout tra griglia ed elenco;
- `BookGridView`, `BookListView`, `ViewModeToggle`: questi sono componenti stateless di supporto. Ricevono i dati tramite "props" e si occupano di gestire la resa grafica dei libri, le immagini di fallback e il passaggio tra le diverse modalità di layout;
- `LoginPage` e `SignUp`: gestiscono l'interfaccia per l'inserimento delle credenziali. Comunicando con l' `AuthViewModel` per processare le richieste di accesso o creazione account.

2.3 View model

I `ViewModel` agiscono come intermediari tra `Model`, `Service` e le `View`, gestendo lo stato dell'interfaccia attraverso i custom hook di React.

Di seguito vengono descritti nel dettaglio:

- `AuthViewModel`: questo modulo centralizza la gestione dell'utente. Utilizza l'hook `useEffect` per impostare un osservatore che rileva automaticamente se un utente è loggato, aggiornando lo stato globale tramite il metodo `setUser`. Espone inoltre le funzioni `login`, `logout` e `signUp`, che permettono all'app di comunicare direttamente con il sistema di autenticazione di Firebase;
- `BookViewModel`: gestisce lo stato dei risultati, i caricamenti e la paginazione. Il metodo `search` pulisce i risultati precedenti e interroga il servizio per l'acquisizione dei libri, mentre `loadMore` si occupa di incrementare l'indice dei risultati per implementare lo scroll sino ad esaurimento dei risultati;
- `LibraryViewModel`: gestisce l'interazione tra l'utente e la sua collezione personale. Utilizza un listener in tempo reale su Firestore per fare in

modo che la libreria mostrata sia sempre sincronizzata con il database. Include metodi per aggiungere o rimuovere volumi, e gestisce un sistema di feedback per avvisare l'utente del successo delle operazioni di aggiunta e rimozione, tramite notifiche a scomparsa;

- `BookDetailViewModel`: si occupa del recupero delle informazioni di un singolo libro. Riceve un identificativo univoco e attiva un effetto che interroga il `bookService`, gestendo gli stati di caricamento per permettere alla vista di mostrare uno spinner durante l'attesa dei dati.

2.4 Service

I Service sono moduli dedicati esclusivamente alla comunicazione con l'esterno, isolando le chiamate di rete dal resto della logica.

Di seguito vengono descritti nel dettaglio:

- `bookService`: gestisce le richieste HTTP verso l'API Google Books. Il metodo `searchBooks` costruisce dinamicamente la query e gestisce la paginazione tramite i parametri `startIndex` e `maxResults`. Il metodo `getBookById` permette invece di recuperare i dettagli di un singolo volume per la visualizzazione di dettaglio;
- `libraryService`: si occupa delle operazioni di lettura e scrittura su Firestore per quanto riguarda la libreria utente. Include `addBook` per salvare un nuovo documento nella collezione dell'utente, `removeBook` per eliminarlo e `getUserLibrary` per recuperare l'intera collezione;
- `firebase`: contiene la configurazione necessaria per inizializzare la connessione ai servizi Google Firebase Authentication e Firestore Datastore.

3.0 Conclusioni

Lo sviluppo di BookTracker ha permesso di realizzare un'applicazione solida e performante, dimostrando l'efficacia del pattern MVVM nell'organizzare il codice in modo scalabile. L'integrazione tra React e i servizi Firebase ha garantito un'esperienza utente reattiva, dove la sincronizzazione dei dati avviene quasi istantaneamente e senza la necessità di ricaricare la pagina. L'utilizzo di API esterne come quella di Google Books ha inoltre fornito una base dati immensa, rendendo lo strumento utile e ben fornito permettendo all'utente di effettuare ricerche in maniera affidabile.

3.1 Eventuali sviluppi futuri

Nonostante l'applicazione sia completa nelle sue funzioni principali, sono stati identificati diversi ambiti di miglioramento per rendere l'esperienza ancora più ricca.

Di seguito alcune delle idee sviluppabili:

- Personalizzazione della Libreria: implementazione di categorie personalizzate per permettere all'utente di organizzare meglio i propri volumi;
- Tracker di lettura: dare all'utente la possibilità per ciascun volume di segnare le pagine lette mostrando percentuale di lettura e pagine mancanti;
- Recensioni e interazioni social: aggiunta di una sezione per le recensioni personali e la possibilità di condividere la propria lista di libri e letture con altri utenti della piattaforma;
- Statistiche di Lettura: dashboard che mostri graficamente il numero di libri letti nell'anno, preferenze e performance di lettura per un certo lasso di tempo.