UNIVERSITA' DEGLI STUDI DI SALERNO

Dipartimento di Informatica

PROGETTO

CORSO BASE DI DATI II

Book Review Manager

Anno Accademico 2024-2025

Candidato:
Debora Pucciarelli
Matricola: 0522501933

Docenti corso:
Prof.ssa Genoveffa Tortora
Prof. Luigi Di Biasi

Indice

Capitolo 1

Book Review Manager

Il progetto consiste nello sviluppo di una web app interattiva che permette la gestione e visualizzazione di libri e recensioni degli utenti, sfruttando un database NoSQL (MongoDB) per l'archiviazione dei dati. L'obiettivo è offrire una piattaforma che implementi in modo coerente le operazioni CRUD, includa funzionalità di JOIN tra collezioni e sia conforme ai principi del paradigma NoSQL (proprietà BASE, teorema CAP).

1.1 Introduzione

Nella documentazione vengono presentati i seguenti elementi chiave:

- i requisiti necessari per il progetto da sviluppare;
- l'analisi e la pulizia dei dati raccolti;
- le modalità di caricamento dei dati nel database;
- le tecnologie web scelte per creare l'interfaccia;
- la descrizione delle pagine dell'applicazione e delle funzionalità implementate;
- le istruzioni per installare e avviare l'applicazione sviluppata.

1.1.1 Tecnologie e Tool utilizzati

Durante lo sviluppo dell'applicazione **Book Review Manager**, è stato fatto uso di vari strumenti e tecnologie chiave, ognuno con un ruolo specifico nell'architettura del sistema:

- **MongoDB:** un database NoSQL orientato ai documenti, che memorizza i dati in formato `BSON` (una versione binaria di `JSON`). Questo strumento offre funzionalità come il *replica set* per garantire alta disponibilità, aggregazioni complesse tramite pipeline, e un'ottima integrazione con API moderne, rendendolo perfetto per le esigenze del progetto.
- **React:** una libreria JavaScript progettata per creare interfacce utente dinamiche e modulari, utilizzata per costruire la parte frontend dell'applicazione in modo reattivo e flessibile.
- **Flask:** un micro-framework in Python utilizzato per sviluppare le API RESTful che collegano il frontend al database. Gestisce il routing delle richieste HTTP, la serializzazione dei dati in formato JSON e le operazioni CRUD.
- **Pymongo:** il driver ufficiale Python per MongoDB, impiegato per interagire con il database, eseguendo query, comandi e operazioni avanzate come `aggregate` e `lookup` in modo semplice ed efficace.

Inoltre, è importante notare che l'uso di **MongoDB** comporta l'uso automatico modello **BASE** (Basically Available, Soft state, Eventual consistency). Anche se non sono state implementate manualmente logiche di consistenza eventuale, queste caratteristiche sono garantite dal sistema di replica set e dalle proprietà di MongoDB, che assicurano disponibilità e tolleranza ai guasti in ambienti distribuiti.

1.1.2 Requisiti e Origine del Dataset

Per la realizzazione del progetto è stato necessario individuare un dataset che contenesse informazioni dettagliate su libri, utenti e recensioni, utili per la gestione e l'analisi delle valutazioni librarie.
Il dataset utilizzato è stato preso dalla piattaforma ***Kaggle***, da questo link:
https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset
Si tratta di un dataset pubblico e ben strutturato, che comprende tre file principali:

- ***books.csv***: sono raccolte tutte le informazioni riguardanti ai libri presenti nel dataset per comprendere al meglio le caratteristiche di ciascun libro. Tra gli attributi troviamo: il *titolo*, l'*autore*, l'*anno di pubblicazione*, il *codice ISBN* che identifica ogni libro in modo univoco e la *casa editrice*.

- ***users.csv***: contiene i dati degli utenti che hanno inserito le valutazioni. Ogni utente è identificato da un *ID univoco*, la *posizione geografica* e l'*età* dell'utente.

- ***ratings.csv***: contiene le valutazioni che gli utenti hanno assegnato ai libri consentendo di esaminare le loro preferenze riguardanti ai vari libri. Ogni valutazione è associata a un *ID utente* e al codice *ISBN* del libro, e include un *punteggio* che varia da 1 (valutazione minima) a 10 (valutazione massima).

Di seguito verrà mostrata una tabella che elenca tutti gli attributi e i relativi tipi di dato per ogni file del dataset.

Tabella 1.1: Descrizione dei file e degli attributi del dataset

File	Attributo	Tipo di dato
books.csv	ISBN	stringa
	Book-Title	stringa
	Book-Author	stringa
	Year-Of-Publication	intero
	Publisher	stringa
	Url-image	stringa
users.csv	User-ID	intero
	Location	stringa
	Age	intero
ratings.csv	User-ID	intero
	ISBN	stringa
	Book-Rating	intero (1-10)

1.1.3 Preprocessing dei dati

Prima di iniziare a lavorare direttamente su questi file, sono state effettuate alcune operazioni di "pulizia" e preparazione dei dati assicurando che questi siano gestiti correttamente e mantenendo alta la loro qualità prima di essere caricati nel database. Per fare ciò è stato implementato uno script in Python chiamato `pre-processing.py` che automatizza il processo di preprocessing caricando i dataset con `pandas.read_csv`.

Pulizia del dataset Books

Durante la fase di pulizia dei dati sui libri, sono state messe in atto diverse strategie per migliorare la qualità e l'affidabilità del dataset.

- Sono state eliminate tutte le righe con valori mancanti nelle colonne *ISBN, Book-Title, Book-Author*;
- Sono stati filtrati gli *Year-Of-Publication* mantenenendo solo quelli compresi tra il 1400 e il 2025, per escludere dati anomali o errati;
- Sono stati sostituiti i valori mancanti in Publisher e negli URL delle immagini con stringhe fittizie, per evitare problemi durante la visualizzazione o l'elaborazione dei dati;
- Sono stati rimossi i libri con ISBN duplicati per garantire che ogni libro nel dataset avesse un identificativo unico.

Pulizia del dataset Users

La fase di pulizia del dataset per gli utenti ha comportato diverse operazioni per garantire la validità e la coerenza dei dati.

- Sono stati rimossi tutti gli utenti con valori mancanti nei campi *User-ID* e *Location*;
- Nella colonna con il campo *Age* sono stati mantenuti solo valori tra 10 e 100 per evitare valori irrealistici o errati;
- Infine sono stati rimossi, anche in questo caso, utenti con *ID* duplicati per garantire che ogni utente fosse rappresentato in modo univoco nel dataset.

Pulizia del dataset Ratings

Anche il dataset con le valutazioni ha subito un attento processo di pulizia. In particolare:

- Sono state mantenute solo le valutazioni riferite a *ISBN* e *User-ID* presenti nei dataset books e users, per garantire che le tabelle siano coerenti tra loro;
- Sono state eslusi i rating con valore zero, poiché non rappresentano vere valutazioni da parte degli utenti, mantenedo solo le valutazioni comprese tra 1 e 10;
- Per migliorare la qualità dei dati, sono state mantenuti solo quegli utenti che hanno fornito almeno tre valutazioni. Gli utenti che non soddisfacevano questo criterio sono stati rimossi anche dal dataset degli utenti, in modo da mantenere allineate le informazioni tra i due set di dati.

Output e salvataggio

Concluso il processo di pre-processing, a video vengono mostrate le dimensioni dei tre dataset e salvati i file puliti nella directory `database_clear`.

1.1.4 Caricamento dei dati nel database MongoDB

Una volta completato il pre-processing, si passa al caricamento dei dataset puliti nel database MongoDB distribuito in replica set, ovvero una configurazione ridondata su più nodi che garantisce alta disponibilità e tolleranza ai guasti e a tale scopo, è stato utilizzato uno script Python (*carica.py*).

Connessione al Replica Set MongoDB

Il client MongoDB si connette specificando tre nodi locali (`localhost:27017`, `localhost:27018`, `localhost:27019`) che fanno parte dello stesso replica set chiamato `rs0`. Questa configurazione è utile per garantire la continuità del servizio: se un nodo fallisce, gli altri possono continuare a rispondere.

```
client = MongoClient(
"mongodb://localhost:27017,localhost:27018,localhost:27019/replicaSet=rs0"
)
```

Caricamento dei file CSV puliti

Dopo la pulizia, i file *Books_ clean.csv*, *Users_ clean.csv* e *Ratings_ clean.csv* vengono caricati con la libreria pandas e trasformati in liste di dizionari Python, uno per ogni riga. Questo formato è compatibile con MongoDB e consente di inserire facilmente i dati nel database ***bookdb***.

Inserimento dei dati nel database MongoDB

Una volta che i dati sono stati trasformati in liste di dizionari, vengono inseriti nelle rispettive collezioni del database `bookdb` (books, users e ratings) grazie al metodo `insert_many` offerto da PyMongo. L'inserimento avviene seguendo queste istruzioni:

```
db.books.insert_many(books.to_dict(orient="records"))
db.ratings.insert_many(ratings.to_dict(orient="records"))
db.users.insert_many(users.to_dict(orient="records"))
```

Al termine dell'importazione vengono stampati a video il numero totale di documenti presenti in ciascuna collezione, così da verificare che l'operazione sia avvenuta correttamente.

Cancellazione Completa dei Dati

Per facilitare lo sviluppo e i test, è stata implementato uno script che permette di eliminare tutti i documenti presenti nelle collezioni principali del database (books, users e ratings) utilizzando la funzione `delete_many()` su ciascuna collezione, ripristinando quindi un database vuoto e pronto per un nuovo inserimento dati (`carica.py`).

Questo reset è utile sia per ripulire il database tra più test, per resettare lo stato dell'applicazione in fase di sviluppo ed evitare residui di dati orfani o incoerenti.

Capitolo 2

Backend applicativo

2.1 Backend Applicativo e API RESTful

Il backend del progetto si basa su *Python* e *MongoDB*, ed è progettato per offrire un'interfaccia RESTful che permetta di interagire con le tre collezioni principali del database: `books`, `users` e `ratings`. L'architettura è composta da tre script principali:

- `operazioniCrud.py`: implementa tutte le operazioni CRUD (Create, Read, Update, Delete) e la validazione dei dati per ogni collezione;
- `operazioniJoin.py`: contiene le operazioni di join tra le collezioni, basate su pipeline di aggregazione MongoDB;
- `main.py`: avvia il server e definisce tutte le rotte API che espongono le funzionalità dei moduli precedenti.

2.1.1 OperazioniCrud.py

Questo script è stato implemento per definire le funzionalità principali per la manipolazione dei dati, includendo controlli di validazione.

Connessione al Database

La connessione, come già accennato, avviene con una *replica set* MongoDB locale, tramite tre porte (27017, 27018, 27019) che permette di accedere al database `bookdb`.

Validazioni Implementate

Sono stati poi implementati alcuni controlli di valutazione quando si eseguono le operazioni Crud tra cui:

- ISBN: deve avere una lunghezza compresa tra 10 e 13 caratteri;
- Anno di pubblicazione: deve essere un valore numerico compreso tra 1400 e 2025;
- Età: numero intero tra 10 e 100;
- Recensioni: numero intero tra 1 e 10.

In generale, **Book** controlla che i campi obbligatori siano presenti e che i testi abbiano un formato valido.
User verifica che l'ID e la location siano presenti e che l'età rientri in un intervallo accettabile.
Infine, **Rating** assicura che il voto sia valido e che sia associato a un utente e a un libro esistenti.

Funzioni Crud

Infine sono state implementate la varie funzioni *Crud*:

- **Books**: `create_book`, `get_book`, `update_book`, `delete_book`
- **Users**: `create_user`, `get_user`, `update_user`, `delete_user`
- **Ratings**: `create_rating`, `get_rating`, `update_rating`, `delete_rating`

2.1.2 OperazioniJoin.py

Lo script *operazioniJoin.py* è stato sviluppato per eseguire operazioni di join tra le collezioni del database bookdb, sfruttando il costrutto *$lookup* messo a disposizione da MongoDB. Anche in questo caso, la connessione al database, avviene tramite la *replica set locale*. Le funzioni principali implementate sono:

- `get_book_with_ratings(isbn)`: restituisce, per un dato libro, le valutazioni corrispondenti effettuate dai vari utenti;
- `get_user_with_ratings(user_id)`: restituisce per uno specifico utente tutte le valutazioni lasciate dei vari libri considerando che l'utente deve avere almeno 3 recensioni.

Queste funzioni permettono di aggregare facilmente i dati tra collezioni, rendendoli pronti per l'uso all'interno dell'applicazione web o per analisi più approfondite.

Gestione della consistenza nelle operazioni di join

Per garantire la coerenza dei dati e l'accuratezza delle join, è stata implementata una logica di cancellazione a cascata manuale all'interno dello script `operazioniCrud`, che gestisce le operazioni CRUD:

- Quando un libro viene eliminato, il metodo `delete_book` cancella automaticamente tutte le recensioni associate a quel libro (tramite lo stesso ISBN).
- Quando un utente viene eliminato, il metodo `delete_user` elimina contestualmente tutte le recensioni associate a quell'utente (tramite il suo User-ID).

Questa logica previene la presenza di dati orfani nella collezione delle recensioni e assicura che le join restituiscano dati aggiornati e coerenti, evitando riferimenti a libri o utenti non più presenti nel database.

2.1.3 Main.py

Il codice contenuto nello script *main.py* implementa un'applicazione web utilizzando il framework ***Flask***, con l'obiettivo di esporre un'API REST per interagire con un database MongoDB configurato in replica set. Nella fase iniziale, vengono importati i moduli necessari, tra cui *Flask, CORS, pymongo*, e gli script descritti precedentemente *operazioniCrud* e *operazioniJoin*. Successivamente, viene stabilita la connessione al database ***bookdb***, e viene avviata l'applicazione Flask con il supporto per le richieste da origini diverse abilitato tramite CORS.

Rotte per Books

L'API espone diverse rotte per la gestione della collezione `books`.

- `GET /books`: restituisce l'elenco completo dei libri;
- `GET /books/<isbn>`: recupera un singolo libro dato il suo ISBN;
- `POST /books`: utile per inserire un nuovo libro nel database;
- `PUT /books/<isbn>`: utile per aggiornare le informazioni di un libro esistente;
- `DELETE /books/<isbn>`: utilizzato per rimuovere un libro dal database.

Sono inoltre disponibili due rotte aggiuntive:

- `GET /api/books/isbns`: restituisce l'elenco di tutti gli ISBN presenti nella collezione;
- `GET /api/books/isbns/search`: consente di cercare ISBN utilizzando la compilazione automatica;

Rotte per Users

Anche per la collezione `users` sono presenti tutte le operazioni CRUD principali:

- `GET /users`: restituisce tutti gli utenti registrati;
- `GET /users/<user_id>`: recupera i dati di un utente specifico;
- `POST /users`: utile per aggiungere un nuovo utente;
- `PUT /users/<user_id>`: necessario per modificare un utente esistente;
- `DELETE /users/<user_id>`: utilizzato per rimuovere un utente dal database.

In aggiunta, è disponibile la rotta `GET /api/users/userids`, che fornisce l'elenco completo di tutti gli ID utente presenti nel sistema.

Rotte per Ratings

La collezione `ratings` dispone anch'essa di rotte CRUD complete:

- `GET /ratings`: consente di ottenere tutte le valutazioni registrate;
- `GET /ratings/<user_id>/<isbn>`: utile recuperare la valutazione lasciata da uno specifico utente su uno specifico libro;
- `POST /ratings`: necessario per inserire una nuova valutazione;
- `PUT /ratings/<user_id>/<isbn>`: necessario per aggiornare una valutazione esistente;
- `DELETE /ratings/<user_id>/<isbn>`: utile per cancellare una valutazione dal database.

Rotte Join Sono previste due rotte specifiche per operazioni di join tra le collezioni: `GET /books/<isbn>/ratings`, che restituisce un libro con tutte le valutazioni annidate al suo interno; `GET /users/<user_id>/ratings`, restituisce tutte le recensioni fatte da un utente. Se l'utente non esiste, mostra un errore. Se ha meno di 3 recensioni, lo segnala. Solo se ha almeno 3 recensioni vengono mostrati i dati completi.

Queste rotte permettono di visualizzare la relazione tra libri, utenti e voti in maniera aggregata e chiara.

Rotte Aggregazioni

L'applicazione offre anche alcune rotte dedicate ad analisi aggregate sui dati:

- `GET /books/top10`, che restituisce la lista dei 10 libri più recensiti nel sistema, ordinati in maniera crescente per numero di valutazioni ricevute;
- `GET /ratings/distribution`, che fornisce una panoramica della distribuzione dei voti assegnati dagli utenti, da 1 a 10.

Rotte test Replica Set

Infine, la rotta `GET /test-replica` è stata creata per scopi di test e monitoraggio. Questa rotta permette di controllare se l'applicazione è correttamente connessa al *nodo primario* del replica set MongoDB e offre anche un conteggio totale dei libri presenti nella collezione, il che è utile per confermare che i dati siano sincronizzati.

Avvio del Server

Il server viene avviato con `app.run(debug=True)` che per default gira su `localhost:3000`.

Capitolo 3

Frontend

3.1 Interfaccia Grafica Frontend

Per la realizzazione dell'interfaccia grafica del progetto è stato utilizzato **React**, una libreria JavaScript moderna e molto diffusa per la creazione di interfacce utente dinamiche e modulari. Lo scopo dell'interfaccia è quello di consentire all'utente l'interazione completa con il sistema, permettendo di effettuare operazioni di visualizzazione, inserimento, modifica e cancellazione dei dati relativi a libri, utenti e valutazioni.

3.2 Inizializzazione e interfaccia del progetto con React

L'ambiente di sviluppo è stato configurato utilizzando lo strumento `create-react-app`, che automatizza la creazione della struttura iniziale del progetto. I seguenti comandi sono stati eseguiti da terminale:

```
npx create-react-app db2
cd db2
npm start
```

- `npx create-react-app db2`: questo comando crea una nuova applicazione React chiamata `db2` definendo i file e le dipendenze necessarie;
- `cd db2`: accede alla directory dell'applicazione;
- `npm start`: avvia il server locale di sviluppo, rendendo l'interfaccia accessibile tramite browser all'indirizzo `http://localhost:3000`.

3.2.1 Home_Page: App.js

Lo script `App.js` definisce la ***home page*** dell'interfaccia web **Book Review Manager**, la quale consente la navigazione tra diverse sezioni (libri, utenti, recensioni, join e operazioni di aggregazione) sfruttando il `React Router` per la gestione delle rotte.

Figura 3.1: Home_page WebApp Book Review Manager

Gestione delle Rotte

Il componente `<Routes>` si occupa della navigazione all'interno dell'applicazione. Collega ogni URL a una specifica pagina, caricando il corrispondente componente React. In questo modo, quando l'utente cambia pagina, l'app mostra il contenuto corretto.

- **Sezione Libri** (`/books`): questa sezione include varie sottopagine riguardanti la ricerca, l'inserimento, l'aggiornamento e la cancellazione dei libri.

Figura 3.2: menu_libri

- **Sezione Utenti** (`/users`): gestisce le principali operazioni CRUD associate agli utenti.

Figura 3.3: menu_utenti

- **Sezione Recensioni** (`/ratings`): offre le funzionalità base per creare, leggere, modificare ed eliminare recensioni.

Figura 3.4: menu_recensioni

- **Sezione Join** (`/join`): permette la selezione di uno delle due operazioni join implementate quali o la visualizzazione di un libro con le sue recensioni o un utente con tutte le sue valutazioni.

Figura 3.5: menu_join

- **Sezione Aggregazioni** (`/operazioni`): consente la scelta di due operazioni avanzate implementate come filtraggio dei primi 10 libri con il maggior numero di recensioni e la distribuzione delle recensioni dei vari libri (da 1 a 10).

Figura 3.6: menu_aggregazione

Modularità e Manutenibilità

Tutti i componenti per ciascuna pagina sono modulari e separati in directory specifiche: `pagesBook`, `pagesUser`, `pagesRating`, `pagesJoin`, `pagesOperazioni`. garantendo riutilizzabilità e manutenibilità del codice, facilitando lo sviluppo e il debugging dell'interfaccia.

3.2.2 PagesBook: gestione libri

Il sistema permette di gestire i libri tramite le classiche operazioni CRUD: inserimento, eliminazione, ricerca e aggiornamento. Ogni funzionalità è accessibile attraverso un'interfaccia React semplice e intuitiva, dove ogni pagina è dotata di una barra di navigazione `Navigation_Bar` che consente di spostarsi rapidamente tra le diverse sezioni dell'applicazione.

L'***inserimento*** di un nuovo libro richiede che tutti i campi obbligatori siano compilati correttamente, come l'ISBN, il titolo, l'autore, l'anno di pubblicazione, l'editore e gli URL delle immagini. Se uno di questi dati manca o non è valido, ad esempio un ISBN non conforme o un anno fuori dall'intervallo consentito, il sistema blocca l'operazione e restituisce un messaggio di errore chiaro.
La ***cancellazione*** di un libro si effettua specificando il relativo ISBN. Se il libro esiste nel database, viene rimosso con successo; in caso contrario, l'utente viene avvisato che il libro non è stato trovato.
La funzionalità di ***ricerca*** permette di ottenere tutte le informazioni su un libro a partire dal solo ISBN.
L'***aggiornamento***, invece, consente di modificare tutti i campi ad eccezione dell'ISBN, che rimane fisso come identificativo. Anche in questo caso, i dati vengono controllati e validati lato backend prima di essere salvati.

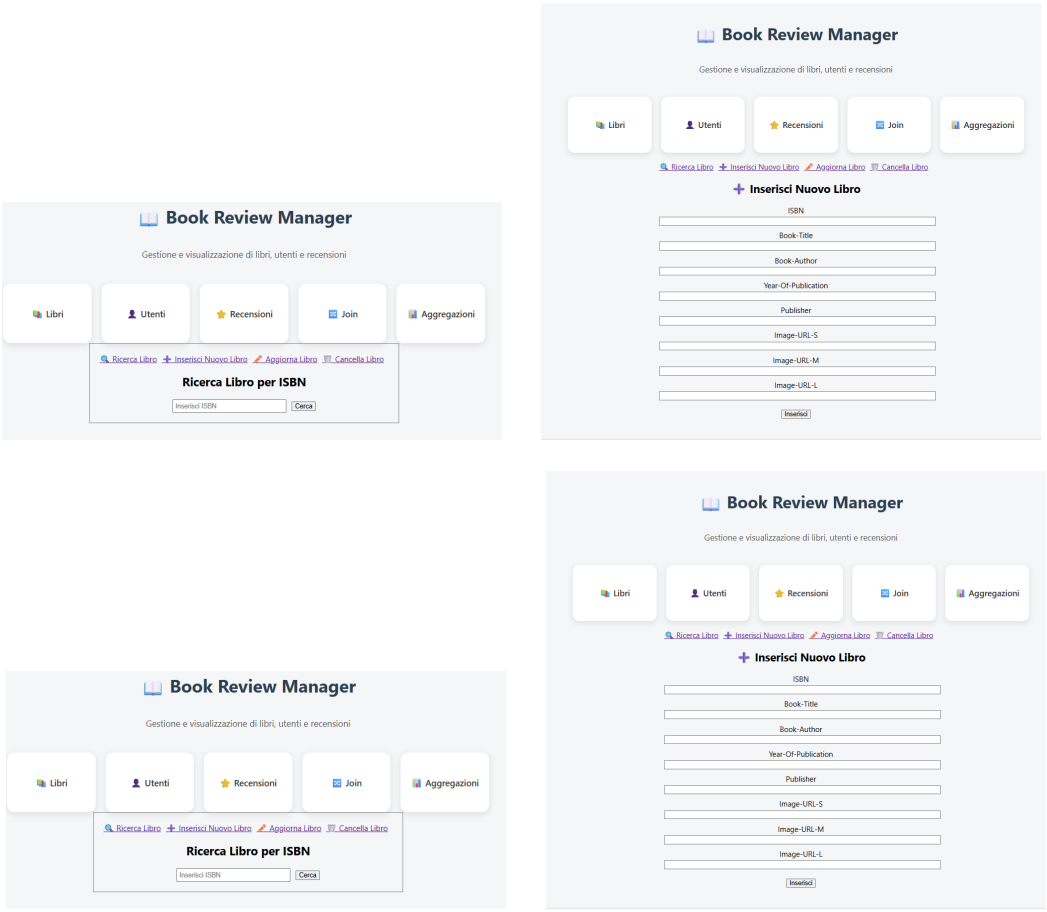


Figura 3.7: Interfaccia web: operazioni crud sui libri

3.2.3 PagesUser: gestione utenti

Come nel caso precedente, la gestione degli utenti avviene tramiete le operazioni Crud.

Per ***ricercare*** un utente, bisogna inserire `User-ID`, visualizzando opportunamente tutte le informazioni ad esso associate, come anno e locazione. Se l'user-id inserito non è presente restituirà un errore.
L'***inserimento***, invece, richiede i campi obbligatori `User-ID`, `Location` e `Age`. Il backend effettua controlli di validazione per garantire la correttezza dei dati (ad esempio: età compresa tra 10 e 100 anni, User-ID numerico, Location non vuota). In caso di valori non validi o mancanti, viene restituito un messaggio di errore chiaro all'utente.
L'***aggiornamento*** è basato anch'esso sull'identificativo `User-ID`: se l'utente è presente nel database, è possibile aggiornare i campi `Location` e/o `Age`, previa validazione. Se l'utente non esiste, l'operazione viene annullata e viene notificato un errore.
Infine, la ***cancellazione*** è possibile specificando l'`User-ID`. Se corrisponde a un utente esistente, viene rimosso dal database; altrimenti viene restituito un errore.

Tutte le pagine dedicate alla gestione degli utenti sono dotate di una barra di navigazione, che consentendo a chi la utilizza di muoversi facilmente tra le sezioni principali dell'applicazione.

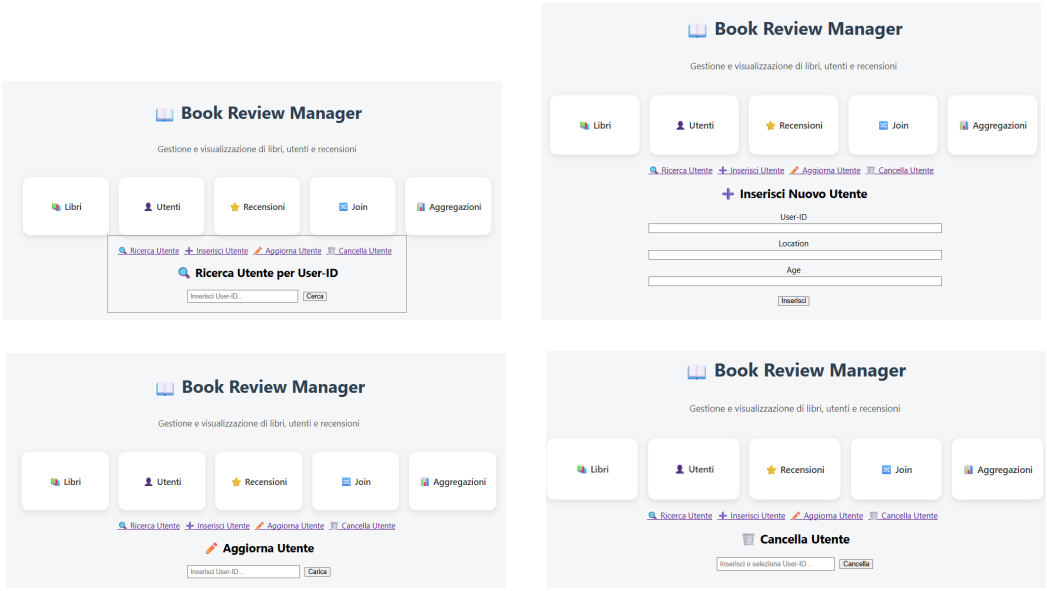


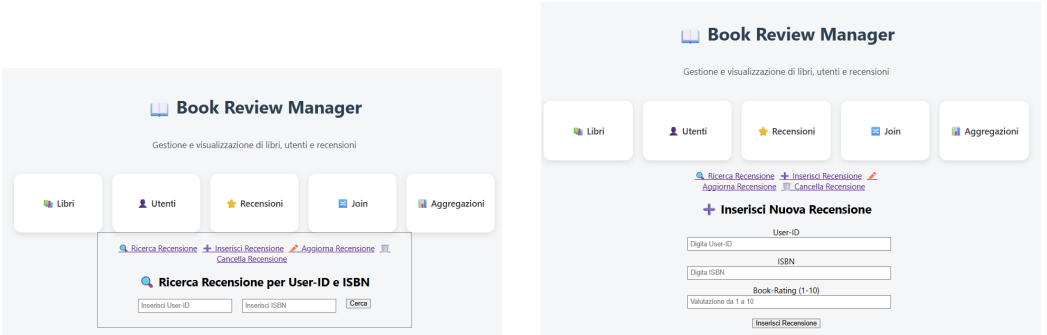
Figura 3.8: Interfaccia web: operazioni crud sugli utenti

3.2.4 PagesRating: gestione recensioni

La gestione delle recensioni permette di associare una valutazione a una coppia composta da utente e libro, tramite operazioni CRUD.

La ***ricerca*** viene effettuata specificando sia lo `User-ID` che l'`ISBN`. Se non è presente una valutazione da parte di quell'utente per il libro indicato, il sistema segnala l'assenza; altrimenti, mostra la recensione corrispondente.
L'***inserimento*** è composto dai seguenti campi `User-ID`, `ISBN` e `Book-Rating` (valore intero da 1 a 10). Prima di aggiungere una nuova recensione, il backend verifica che non esista già una valutazione per quella coppia utente-libro. In caso contrario, viene restituito un messaggio di errore.
L'***aggiornamento*** funziona allo stesso modo: se esiste una recensione per l'utente e il libro selezionato, è possibile modificare la valutazione inserendo un nuovo valore valido; se invece la recensione non è presente, il sistema notifica l'utente dell'errore.
Anche la ***cancellazione*** richiede `User-ID` e `ISBN` per identificare la recensione. Se questa esiste, viene rimossa correttamente; in caso contrario, viene segnalata la sua assenza.

Come per le altre sezioni dell'applicazione, ogni pagina relativa alla gestione delle recensioni include una barra di navigazione per facilitare la consultazione delle funzionalità.



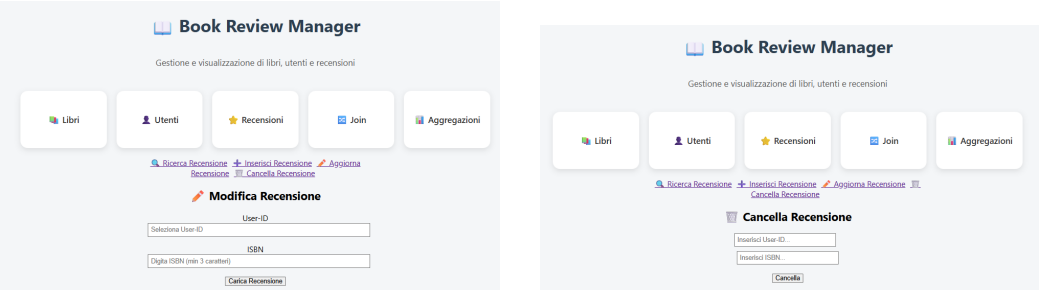


Figura 3.9: Interfaccia web: operazioni crud sulle recensioni

3.2.5 Funzionalità di Completamento Automatico

Per migliorare l'esperienza dell'utente e ridurre gli errori durante l'inserimento dei dati, è stato implementato un sistema di completamento automatico nei moduli per l'inserimento, l'aggiornamento e la ricerca delle tre entità principali: libri, utenti e recensioni.
In particolare, i campi ISBN (per libri e recensioni) e User-ID (per utenti e recensioni) offrono suggerimenti dinamici che si aggiornano in tempo reale mentre si digita. Il sistema interroga il backend tramite apposite API REST e mostra un elenco di valori validi che corrispondono alla parte di testo inserita. Così l'utente può selezionare rapidamente un valore corretto già presente nel database, migliorando l'usabilità dell'interfaccia e prevenendo errori dovuti a ID inesistenti.
Questa funzione è stata integrata in tutte le pagine del frontend React dove è utile, rendendo l'applicazione più semplice e veloce da usare.

3.2.6 PagesJoin: visualizzazione Join

Una volta accesso alla sezione dedicata alle operazioni `Join`, l'utente può scegliere tra due opzioni, visualizzate come due distinti pulsanti:

- **Libro con tutte le sue recensioni**: selezionando questa opzione, è possibile inserire un `ISBN` e ottenere l'elenco degli utenti che hanno recensito quel libro, con le relative valutazioni. In particolare, cliccando su ciascun utente è possibile visualizzare i dettagli a lui associati, come lo *User-ID* e l'*età*.

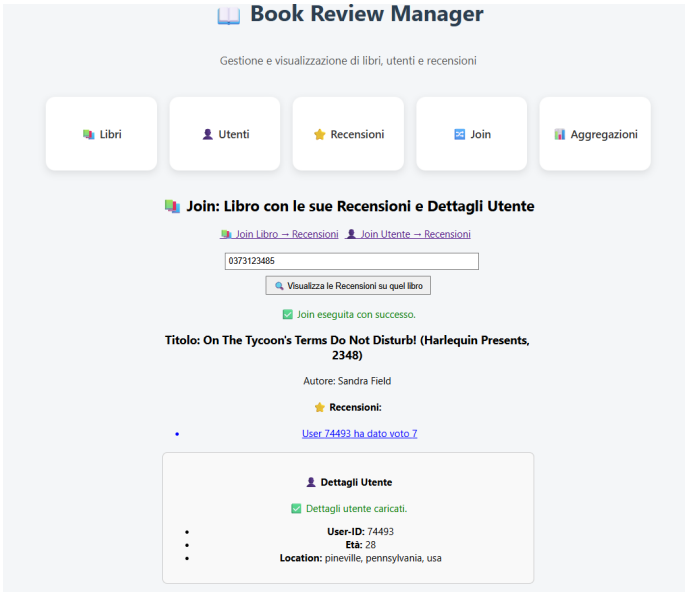


Figura 3.10: Esempio di join: libro con tutte le recensioni

Nel caso in cui l'ISBN inserto non si trova nella base di dati, allora viene restituito un errore.

- **Utente con tutte le sue recensioni**: selezionando questa opzione, è possibile inserire uno `User-ID` e visualizzare tutte le recensioni effettuate da quell'utente, con i relativi libri valutati. Vengono mostrati solo gli utenti che hanno effettuato almeno 3 recensioni, in modo da essere coerente con il pre-processing effettuato e quindi considerare solo gli utenti "attivi". In caso contrario, se l'utente esiste ma ha meno di 3 recensioni, viene mostrato un messaggio informativo; mentre se l'utente non esiste nel database, viene segnalato con un messaggio di errore. Anche in questo caso, cliccando su uno dei libri recensiti è possibile ottenere informazioni aggiuntive sul volume, come il *titolo*, l'*autore* e l'*anno di pubblicazione*.

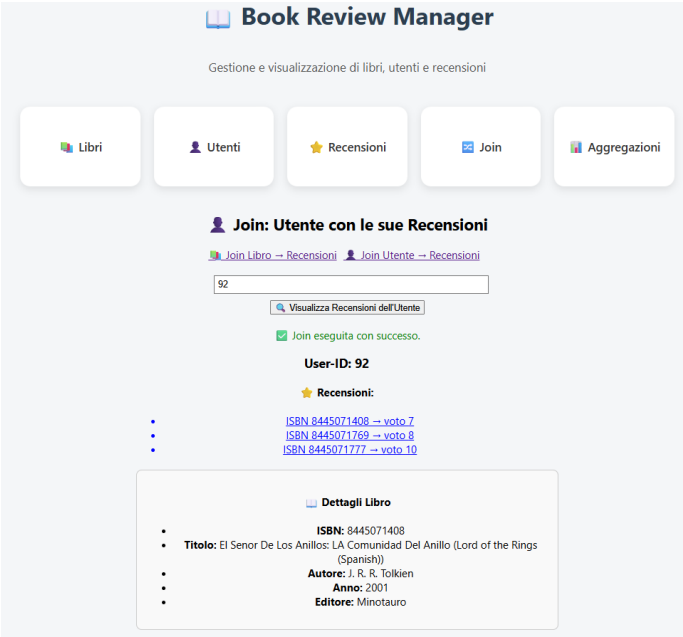


Figura 3.11: Esempio Join: utente con tutte le sue recensioni

Anche in questo caso, se viene inserito un USer-Id non presente nella base di dati, viene segnatalo.

Queste funzionalità sfruttano le operazioni di `lookup` di MongoDB per unire dinamicamente i dati delle collezioni `books`, `users` e `ratings`, rendendo immediata la consultazione delle relazioni tra le entità. Come tutte le altre sezioni dell'applicazione, anche quella dedicata ai `JOIN` include una **barra di navigazione** per facilitare il ritorno ad altre pagine.

3.2.7 PagesOperazioni: selezione operazioni di aggregazioni

Una volta recato nella sezione dedicata alle **aggregazioni**, è possibile fare una scelta di una delle due opzioni implementate tramite pulsanti:

- **Top 10 libri più recensiti**: cliccando su questa opzione, viene mostrata una classifica dei dieci libri che hanno ricevuto il maggior numero di recensioni.
  Per ciascun libro vengono visualizzati l'*ISBN*, il *titolo*, l'*autore* e il *numero totale di recensioni ricevute*.

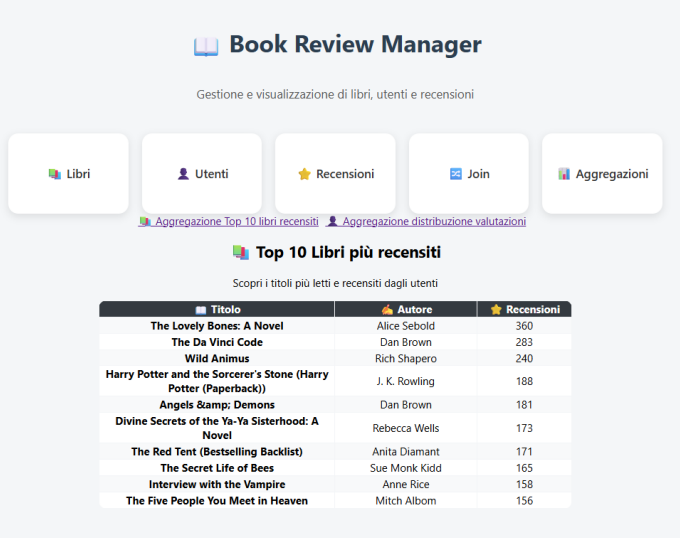


Figura 3.12: Risultato aggregazione: Top 10 libri più recensiti

- **Distribuzione delle valutazioni**: questa funzione permette di visualizzare una distribuzione aggregata delle valutazioni assegnate ai libri dagli utenti, con conteggi per ciascun punteggio da 1 a 10. L'obiettivo è fornire una panoramica sull'andamento generale delle recensioni nel sistema.

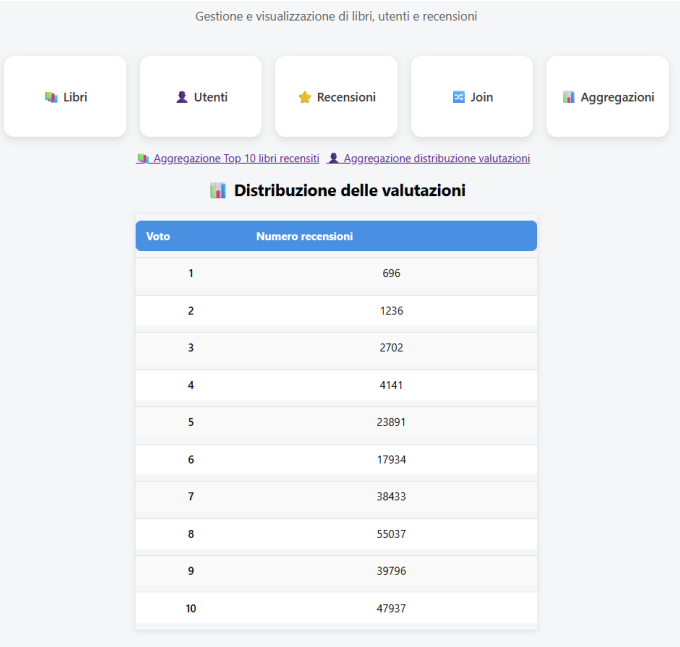


Figura 3.13: Risultato aggregazione: Distribuzione delle valutazioni

Anche in questa sezione, è presente una **barra di navigazione** che consente all'utente di accedere rapidamente alle altre funzionalità dell'applicazione.

Capitolo 4

Note aggiuntive Replica Set MongoDB locale

4.1 Requisiti configurazione Replica Set MongoDB

Per configurare correttamente un Replica Set locale in MongoDB e importare i dati nel database, è necessario disporre di alcuni strumenti essenziali da installare sul computer quali ***MongoDB***, in aggiunta anche la shell ***mongosh*** per interagire con database e avviare la replica; ***mongoimport***, incluso nei MongoDB Database Tools, serve per caricare i dati a partire dai file CSV. Infine, si devono avere a disposizione i tre file contenenti i dati: *Books_ clean.csv, Users_ clean.csv e Ratings_ clean.csv.*

4.2 Avvio del Replica Set

Per attivare l'ambiente di replica in locale, si è utilizzato uno script batch (.bat) che automatizza tutto il processo di configurazione e avvio dei tre nodi MongoDB. Lo script compie in modo sequenziale le seguenti operazioni:

- Definisce i percorsi di lavoro e crea tre directory distinte per i tre nodi MongoDB ognuna destinata a contenere i dati;
- Avvia tre istanze del server mongod, ognuna in ascolto su una porta diversa (27017, 27018, 27019) e appartenente allo stesso Replica Set denominato rs0. In particolare, il nodo sulla porta `27017` funge da nodo primario, mentre gli altri due sono nodi secondari;

Dopo aver avviato i tre processi, lo script mostra un messaggio che guida l'utente a collegarsi con mongosh sulla porta 27017 ed eseguire manualmente il comando `rs.initiate(...)` per inizializzare il Replica Set, specificando i dettagli dei nodi. Questa procedura permette di creare un ambiente MongoDB distribuito su più nodi, utile per testare localmente la replica dei dati, la tolleranza ai guasti e le funzionalità di alta disponibilità del sistema.

4.2.1 Connessione tramite `mongosh` e interrogazione

Per connettersi al nodo principale della replica set, è possibile sia tramite la `PowerShell` che il `Prompt dei comandi` eseguendo il seguente comando:

```
mongosh -port 27017
```

Una volta connessi è possibile visualizzare i database disponibili con il comando ***show dbs***; accedere al database, che nel nostro caso sarà `use booksdb`; elencare le collezioni presenti con `show collections`; ed eseguire interrogazioni e comandi MongoDB tramite operazioni CRUD o aggregazioni.

4.2.2 Esempi di interrogazione

Una volta connessi al database `booksdb`, è possibile interrogare le collezioni presenti. Di seguito si riportano alcuni esempi pratici:

- Estrazione dei primi 10 libri più recensiti ordinati per numero di valutazioni, con i dettagli del titolo e autore.

```
    db.ratings.aggregate([
  {
    $group: {
      _id: "$ISBN",
      count: { $sum: 1 }
    }
  },
  {
    $sort: { count: -1 }
  },
  {
    $limit: 10
  },
  {
    $lookup: {
      from: "books",
      localField: "_id",
      foreignField: "ISBN",
      as: "book_info"
    }
  },
  {
    $unwind: "$book_info"
  },
  {
    $project: {
      ISBN: "$_id",
      title: "$book_info.Book-Title",
      author: "$book_info.Book-Author",
      count: 1
    }
  }
])
```

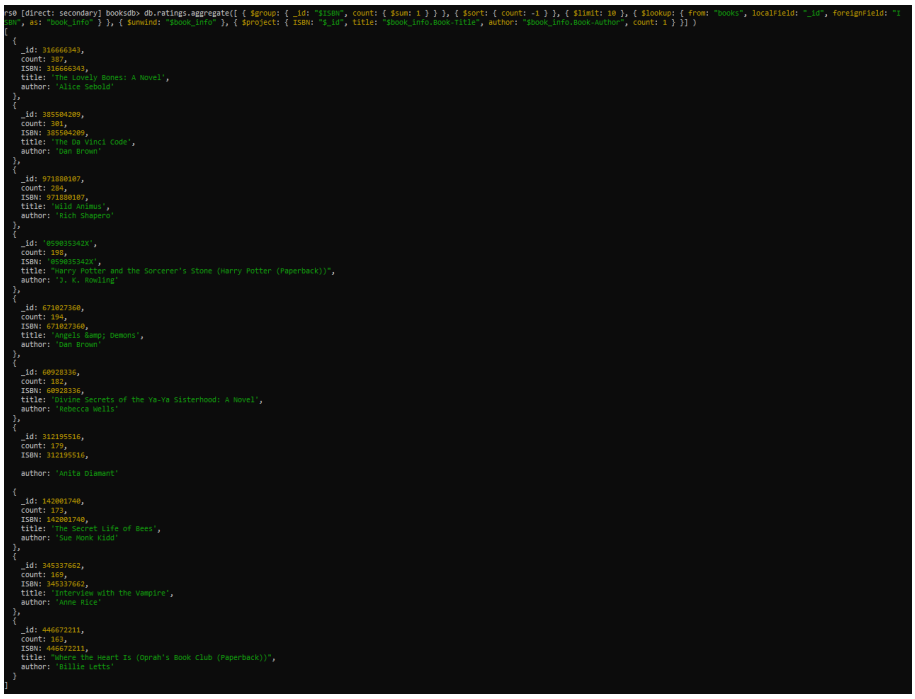


Figura 4.1: Risultato query

- Filtrare solo le valutazioni con punteggio massimo:

```
db.ratings.find({ "Book-Rating": 10 })
```

4.2.3 Forzare arresto del nodo

Inizialmente si identifica quale nodo del replica set ha attualmente il ruolo di `PRIMARY`, si accede al prompt di `mongosh` e si esegue il seguente comando:

```
rs.status().members.filter(m => m.stateStr === "PRIMARY")
```

Una volta individuato il nodo, con un nuovo terminale si procede alla terminazione del processo corrispondente alla porta primaria (27017) con i comandi:

```
netstat -ano | findstr :27017
taskkill /PID <PID_PROCESSO> /F
```

Il primo comando restituisce il PID del processo che occupa la porta 27017; il secondo lo termina forzatamente.

Una volta forzato l'arresto del nodo primario, è necessario identificare quale dei nodi rimanenti ha assunto il ruolo di `PRIMARY` e per farlo, si può rieseguire il comando di connessione a `mongosh` su un'altra shell o prompt dei comandi, ad esempio: `mongosh -port 27018` e successivamente eseguire: `rs.status()` che fornisce una panoramica aggiornata dello stato del Replica Set, evidenziando chiaramente quale nodo è attualmente il nuovo `PRIMARY`.
Se tutto è configurato correttamente, il sistema continua a funzionare senza interruzioni, dimostrando la capacità del Replica Set di garantire continuità operativa anche in presenza di guasti.

Capitolo 5

Installazione di Book Review Manager

5.1 Istruzioni per l'avvio del progetto su altri computer

Per eseguire il progetto **Book Review Manager** su un altro computer, ci sono alcuni semplici passaggi da seguire per configurare correttamente l'ambiente di sviluppo e garantire che l'applicazione funzioni senza problemi.

- **Prerequisiti**: assicurarsi di avere installato Python (versione >=3.7), Node.js con npm, e MongoDB con replica set configurato sui tre nodi;
- **Clonazione del repository**: clonare il repository Git contenente il codice sorgente dell'applicazione.
- **Installazione delle dipendenze backend**: posizionarsi nella cartella backend e installare le librerie Python necessarie, ad esempio tramite `pip install -r requirements.txt`.
- **Configurazione del database**: prima di avviare l'applicazione, è importante assicurarsi che il replica set di MongoDB sia attivo, altrimenti non ci sarà connessione con il database. Per farlo basta eseguire il file *star-replica.bat* che si trova nella cartella del progetto.
- **Avvio del server Flask**: aprire il prompt dei comandi, spostarsi nella cartella del progetto dove si trova lo script principale e avviare il server eseguendo il comando `python main.py`.
- **Avvio del frontend**: aprire un nuovo terminale, spostarsi nella cartella `frontend`, installare le dipendenze Node.js con `npm install` (se non già fatto) e avviare l'interfaccia grafica con il comando `npm start`. Questo aprirà automaticamente il browser all'indirizzo locale `http://localhost:3000`..

Seguendo queste istruzioni, sarà possibile eseguire e testare il progetto in un ambiente locale o anche in ambienti di sviluppo remoti, garantendo così portabilità e replicabilità.