Le API di base sono state strutturate in modo da rispondere alle seguenti esigenze:

* **Gestione dei libri** (in bookController.js, bookModel.js e bookRoutes.js):
  + CRUD su libri (creazione, lettura, aggiornamento e cancellazione).
  + Le operazioni di lettura possono includere la ricerca per ISBN, autore e titolo, come richiesto dal progetto.
* **Gestione delle recensioni** (in reviewController.js, reviewModel.js e reviewRoutes.js):
  + Ogni libro può avere recensioni da parte degli utenti.
  + Operazioni per aggiungere, modificare e cancellare le recensioni sono previste.
* **Gestione degli utenti** (in userController.js, userModel.js e userRoutes.js):
  + Gli utenti possono registrarsi, accedere e fare operazioni legate alle recensioni dei libri.
  + L'accesso è gestito tramite JWT per garantire la sicurezza dell'autenticazione.

**5. Autenticazione tramite JWT**

* **JWT Middleware** (in authMiddleware.js):
  + È stato creato un middleware per proteggere le rotte che richiedono l'autenticazione.
  + Utilizza il token JWT per verificare che l'utente sia autenticato prima di accedere a determinate API.

**6. Connessione a MongoDB**

* **Configurazione del database** (in config/db.js):
  + La connessione a MongoDB è stata configurata tramite Mongoose per consentire la gestione dei dati (libri, recensioni e utenti) in modo persistente.
  + Il file .env viene utilizzato per memorizzare le credenziali del database in modo sicuro.

**7. Impostazione degli Endpoint API**

Sono stati definiti diversi endpoint che permettono di eseguire le operazioni CRUD richieste:

* **GET /books**: Ottieni tutti i libri disponibili nel negozio.
* **GET /books/:isbn**: Ottieni un libro specifico in base all'ISBN.
* **GET /books/author/:author**: Ottieni tutti i libri di un autore.
* **GET /books/title/:title**: Ottieni tutti i libri in base al titolo.
* **POST /register**: Registrazione di un nuovo utente.
* **POST /login**: Autenticazione dell'utente tramite login.

**8. Modalità di Sviluppo e Produzione**

Il comando npm run dev è stato configurato con **nodemon** per permettere l'avvio automatico del server durante lo sviluppo. Questo significa che ogni volta che modifichi un file (per esempio, un controller o una rotta), nodemon riavvierà automaticamente il server.

* npm start è utilizzato per eseguire il server in modalità di produzione, senza il riavvio automatico.

**9. Testing con Postman**

Una volta che tutto è stato configurato, puoi utilizzare **Postman** per testare le API. Le operazioni di test includono la registrazione dell'utente, l'accesso, l'ottenimento dei libri e delle recensioni e la gestione delle recensioni stesse.

**10. Obiettivo del Progetto**

Il progetto ha lo scopo di creare un sistema backend per un'applicazione di recensione di libri, dove gli utenti possono:

* Visualizzare libri.
* Filtrare i libri in base a vari criteri (ISBN, autore, titolo).
* Aggiungere e modificare recensioni sui libri.
* Accedere come utenti registrati con autenticazione tramite JWT.

**Tecniche Utilizzate**

* **Node.js** e **Express.js**: Per costruire l'applicazione server-side.
* **MongoDB** con **Mongoose**: Per la gestione dei dati.
* **JWT**: Per l'autenticazione e la protezione delle rotte.
* **Async/Await** e **Promises** con **Axios**: Per gestire le operazioni asincrone nelle API.

**Conclusione**

Tecnica e concettualmente, sono stati creati tutti i componenti necessari per costruire un'applicazione RESTful in Node.js per un negozio di libri, con un sistema di recensioni, autenticazione tramite JWT, e funzionalità di ricerca avanzate.