Analisi e documentazione del progetto

Alessandro Luppino

Politecnico di Bari
Fondamenti del Web

Contents

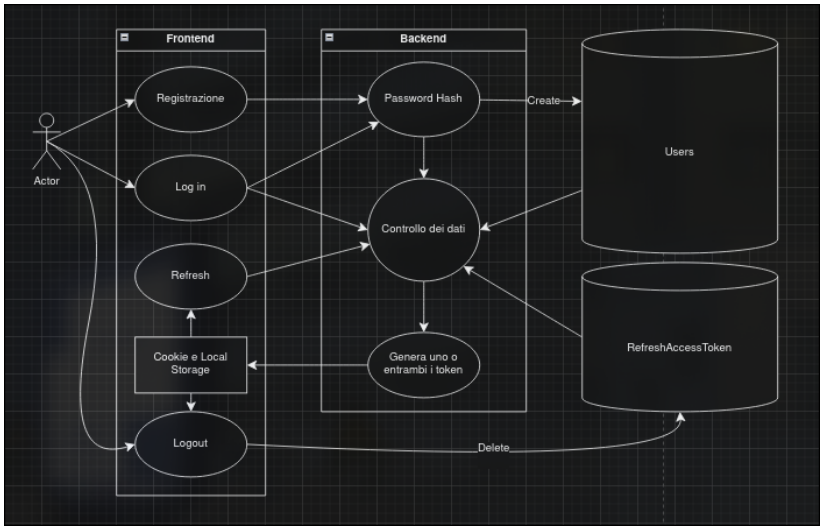
1 Introduzione

unKeep è una web application per la gestione e condivisione di note. Estende il concetto tradizionale di note-taking con funzionalità di collaborazione in tempo reale e un sistema di versioning che permette di visualizzare, ripristinare e gestire le versioni precedenti delle note. Il progetto è completamente hostato e accessibile all'indirizzo `https://unkeep.onrender.com`. Per scopi di sviluppo e testing locale, viene fornito un file `docker-compose.yml` che permette il deploy dell'intera applicazione `[Frontend :8080, Backend :8000, Database :27018]` in ambiente di sviluppo.

2 Architettura del'applicazione

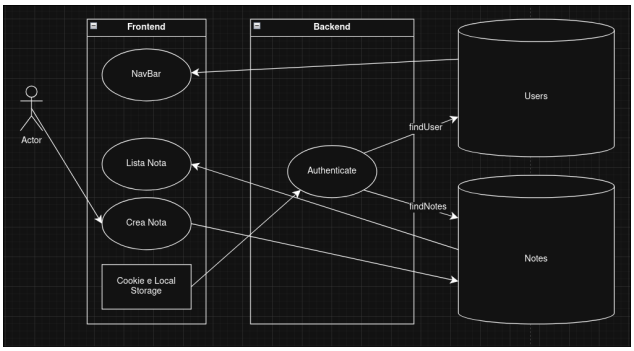
2.1 Autenticazione

Al primo accesso al sito, l'utente viene reindirizzato alla pagina di login, qui può scegliere se effettuare il login o registrarsi.
La procedura di registrazione richiede l'inserimento di un username, di una password e di un indirizzo email. I dati vengono inviati al backend, che verifica la conformità ai requisiti del modello di dati. La password viene sottoposta ad hashing e, in caso di esito positivo, viene creato un nuovo utente nel database. Al termine, l'utente viene autenticato automaticamente.
La procedura di login prevede invece l'inserimento di username (o email) e password. Le credenziali fornite vengono confrontate con quelle presenti nel database; in caso di corrispondenza l'autenticazione ha successo e l'utente viene reindirizzato alla dashboard.
Una volta autenticato, all'utente vengono forniti due token JWT (JSON Web Token): un *access token* con breve durata (15 minuti) e un *refresh token* con durata più lunga (7 giorni). L'access token autentica le richieste al backend, mentre il refresh token consente di ottenere un nuovo access token quando quello corrente scade.
Il refreshToken viene memorizzato in cookie `HttpOnly` per migliorare la sicurezza contro attacchi XSS. In caso di scadenza dell'access token, il frontend invia automaticamente una richiesta al backend per ottenere un nuovo token utilizzando il refresh token. Se anche il refresh token è scaduto, l'utente viene disconnesso tramite logout e reindirizzato alla pagina di login.
La procedura di logout prevede la cancellazione dei cookie contenenti i token e la terminazione della sessione utente e potrà essere utilizzata anche dall'utente tramite la navbar che si trova in dashboard.



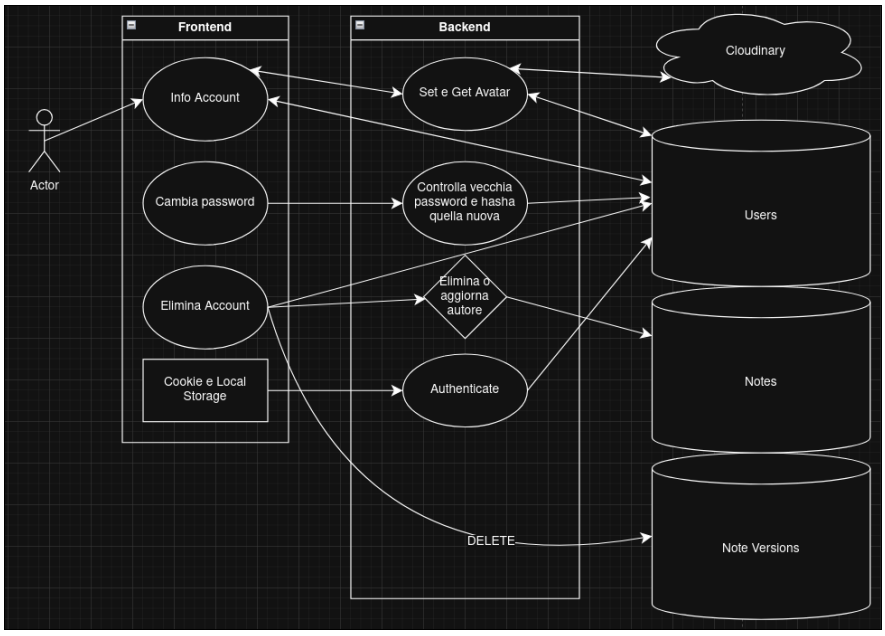
2.2 Dashboard

Dopo l'autenticazione, l'utente viene reindirizzato alla dashboard.
La dashboard presenta una *navbar* che consente di accedere al proprio profilo.
Nella sezione principale viene mostrato l'elenco delle note: sia quelle create dall'utente, sia quelle condivise con lui da altri utenti. Per ciascuna nota vengono visualizzate informazioni sintetiche come l'autore, i collaboratori con permessi di scrittura, le eventuali tag e la data dell'ultima modifica. Inoltre è possibile creare una nuova nota.



2.3 Gestione utente

In questa sezione l'utente può visualizzare e modificare le informazioni del proprio profilo, come username, email e password.
È inoltre possibile aggiornare l'immagine del profilo, caricando una nuova immagine che viene salvata su *Cloudinary*, mentre il relativo link viene memorizzato nel database.
Infine, l'utente ha la possibilità di eliminare definitivamente il proprio account. Tale operazione comporta la cancellazione di tutte le note create e la rimozione dell'utente dal database. In alternativa, l'utente può scegliere di migrare le proprie note al primo collaboratore associato a ciascuna di esse, qualora presente. In ogni caso la cronologia delle note verrà eliminata.

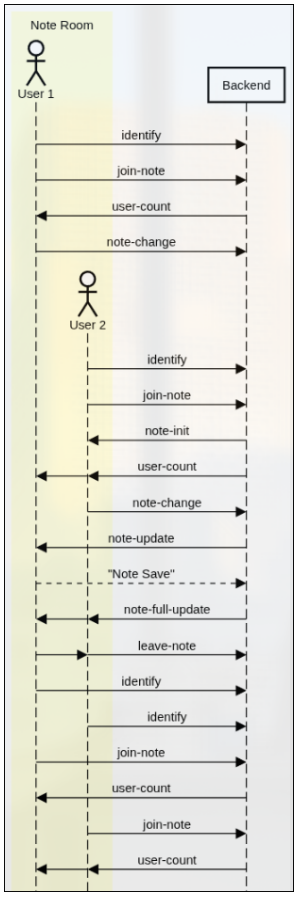


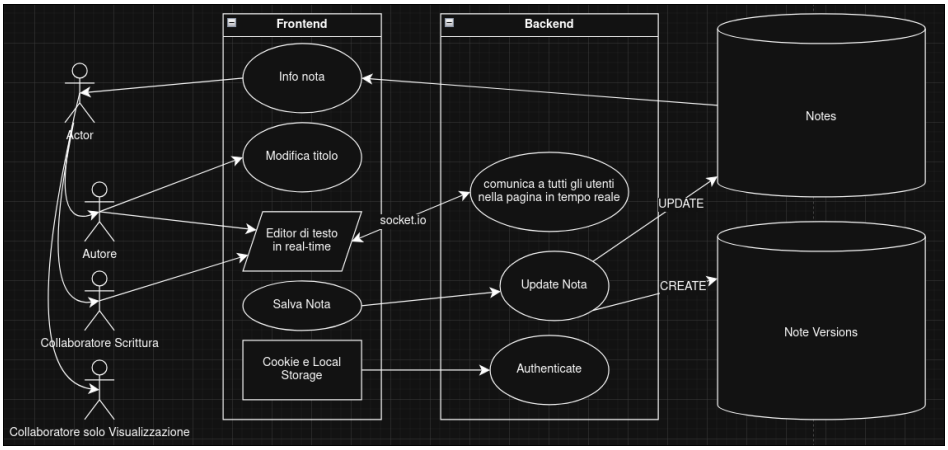
2.4 Nota

L'utente accede all'editor della nota selezionandola dalla dashboard oppure creando una nuova nota. In base ai permessi assegnati, l'utente può assumere uno dei seguenti ruoli rispetto alla nota:

- **Autore:** ha creato la nota. Può modificare il contenuto e il titolo, salvare i cambiamenti ed accedere al pannello di gestione avanzata della nota.
- **Collaboratore con permessi di scrittura:** può modificare esclusivamente il contenuto della nota e salvarne le modifiche.
- **Collaboratore con permessi di sola lettura:** può visualizzare il contenuto della nota, senza possibilità di modificarlo o salvarlo.

L'editor di testo è implementato con *Quill*, che consente la modifica collaborativa e la visualizzazione dei cursori degli altri utenti in tempo reale grazie all'estensione *Quill Cursors*.
La sincronizzazione in tempo reale è gestita tramite *Socket.IO*, che permette di propagare le modifiche a tutti i collaboratori connessi. Gli utenti dotati di permessi adeguati possono salvare le modifiche apportate, generando una nuova versione e aggiornando contestualmente quella corrente.





2.5 Gestione della nota

La gestione della nota è disponibile esclusivamente per l'autore e comprende le seguenti funzionalità:
Cronologia versioni (in tempo reale):

- Visualizzazione delle versioni salvate, con indicazione di autore e timestamp.
- Ripristino della nota a una versione precedente (*revert*).
- Eliminazione delle versioni antecedenti a quella selezionata (*rebase*).

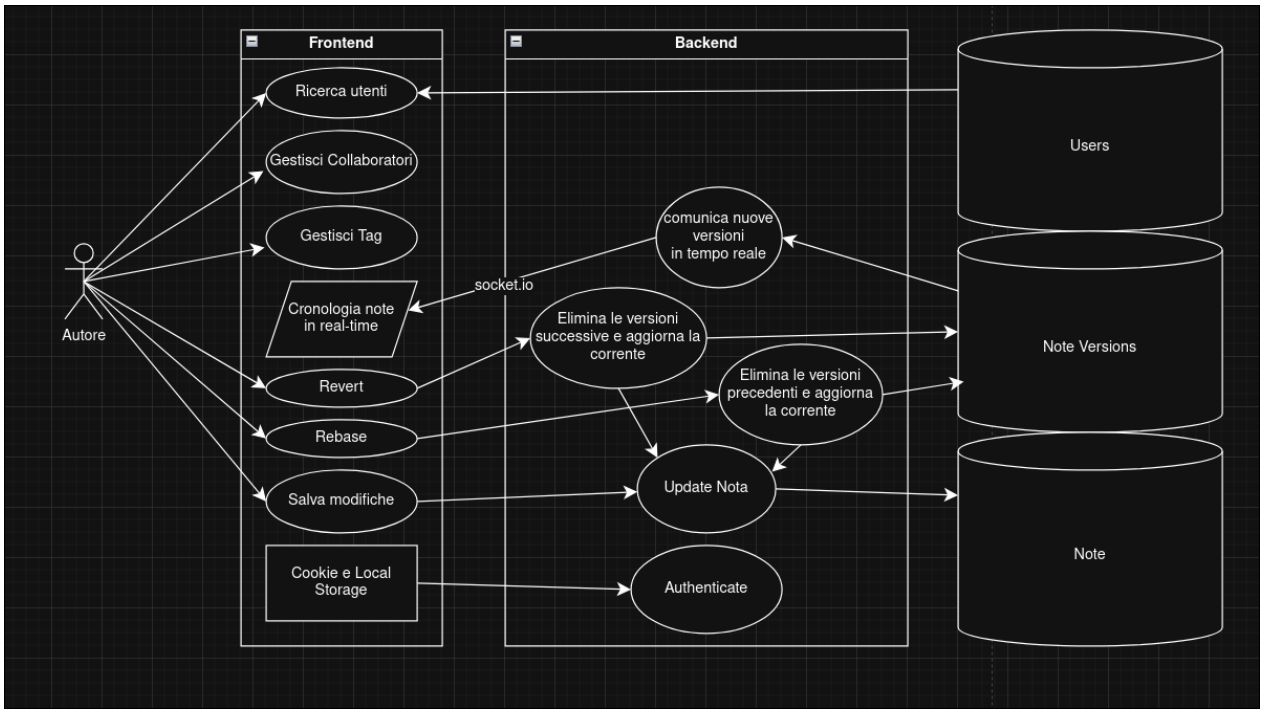
Gestione collaboratori:

- Aggiunta collaboratori tramite ricerca per *username*.
- Rimozione collaboratori.
- Modifica permessi (scrittura/sola lettura).

Gestione tag:

- Aggiunta e rimozione di tag associate alla nota.

Oltre alla possibilità di eliminare o salvare la nota.
Quando una nota viene salvata i cambiamenti vengono inviati ai collaboratori nell'editor in tempo reale.



3 Modello dei dati

3.1 User

Lo schema `User` rappresenta gli utenti registrati nell'applicazione.

- **username** (`String`, obbligatorio, unico): deve contenere da 3 a 20 caratteri alfanumerici.
- **email** (`String`, obbligatorio, unico): validata tramite espressione regolare, salvata in minuscolo.
- **password** (`String`, obbligatorio): da 6 a 128 caratteri, deve contenere almeno una lettera maiuscola, un numero e un carattere speciale.
- **avatar** (`String`, opzionale): URL dell'immagine profilo, di default stringa vuota.
- **createdAt** (`Date`): data di creazione dell'account, impostata automaticamente.

3.2 Note

Lo schema `Note` rappresenta le note create dagli utenti.

- **title** (`String`, obbligatorio): da 1 a 50 caratteri, gli spazi in eccesso vengono rimossi.
- **content** (`String`, opzionale): contenuto della nota, fino a 100000 caratteri.
- **author** (`ObjectId`, obbligatorio): riferimento all'utente autore della nota.
- **collaborators** (`Array`):
  - **user** (`ObjectId`, obbligatorio): riferimento a un utente collaboratore.
  - **permission** (`String`, obbligatorio): permessi assegnati, valori ammessi: `read`, `write`.
- **tags** (`Array di String`): tag testuali (max 30 caratteri) per organizzare le note.
- **currentVersion** (`Number`): versione corrente della nota, aggiornata automaticamente.
- **timestamps**: campi automatici `createdAt` e `updatedAt`.

3.3 NoteVersion

Lo schema `NoteVersion` gestisce la cronologia delle versioni delle note.

- **noteId** (`ObjectId`, obbligatorio): riferimento alla nota associata.
- **createdBy** (`ObjectId`, obbligatorio): riferimento all'utente che ha creato la versione.
- **delta** (`Array di Mixed`): insieme delle modifiche (diff) rispetto alla versione precedente.
- **baseVersion** (`Number`, obbligatorio): numero della versione di riferimento.
- **createdAt** (`Date`): data di creazione della versione, impostata automaticamente.

3.4 RefreshToken

Lo schema `RefreshToken` gestisce i token di refresh degli utenti.

- **userId** (`ObjectId`, obbligatorio): riferimento all'utente proprietario del token.
- **token** (`String`, obbligatorio): valore del token JWT.
- **createdAt** (`Date`): data di creazione, impostata automaticamente. Il campo include un TTL di 14 giorni, dopo i quali il documento viene eliminato automaticamente dal database.

4 Documentazione API

Ogni endpoint restituisce risposte in formato JSON con messaggi di successo o errore. Non verranno documentati, ma si farà riferimento ai codici di stato HTTP standard per indicare il risultato delle operazioni.

4.1 Autenticazione e Autorizzazione (Auth)

4.1.1 `POST /auth/register`

- **Descrizione**: Registra un nuovo utente nel sistema.
- **Metodo**: `POST`.
- **Autenticazione**: Nessuna (endpoint pubblico).
- **Body (raw JSON)**: Richiede un oggetto JSON contenente l'`username`, l'`email` e la `password` del nuovo utente.

4.1.2 `POST /auth/login`

- **Descrizione**: Autentica un utente e restituisce un `accessToken` come json e un `refreshToken` come cookie `HttpOnly`.
- **Autenticazione**: Nessuna (endpoint pubblico).
- **Body (raw JSON)**: Richiede un oggetto JSON contenente l'email o l'username in `emailOrUsername` e la `password` dell'utente.

4.1.3 `POST /auth/refresh`

- **Descrizione**: Aggiorna l'`accessToken` utilizzando il `refreshToken` fornito nel cookie `HttpOnly`.
- **Metodo**: `POST`.
- **Autenticazione**: Nessuna (endpoint pubblico).

4.1.4 `POST /auth/logout`

- **Descrizione**: Disconnette l'utente corrente eliminando il `refreshToken` dal database e cancellando il cookie.
- **Metodo**: `POST`.
- **Autenticazione**: Richiede `Bearer Token`.

4.1.5 `POST /auth/change-password`

- **Descrizione**: Permette all'utente autenticato di cambiare la propria password.
- **Metodo**: `POST`.
- **Autenticazione**: Richiede `Bearer Token`.
- **Body (raw JSON)**: Richiede un oggetto JSON contenente la `currentPassword` e la `newPassword`.

4.2 Gestione Note (Notes)

4.2.1 `POST /notes`

- **Descrizione**: Crea una nuova nota.
- **Metodo**: `POST`.
- **Autenticazione**: Richiede `Bearer Token`.
- **Body (raw JSON)**: Richiede un oggetto JSON con il `title` della nota e opzionalmente: il `content`, un array di `collaborators` (ciascuno con un `user` ID e un `permission`), e un array di `tags`.

4.2.2 `GET /notes`

- **Descrizione**: Recupera tutte le note di cui l'utente autenticato è autore o collaboratore. Le note senza cronologia associata vengono considerate temporanee e non vengono restituite; durante la chiamata esse vengono rimosse dal database per garantire coerenza dei dati.
- **Metodo**: `GET`.
- **Autenticazione**: Richiede `Bearer Token`.

4.2.3 `GET /notes/{noteId}`

- **Descrizione**: Recupera una specifica nota tramite il suo ID se l'utente autenticato è l'autore o un collaboratore.
- **Metodo**: `GET`.
- **Autenticazione**: Richiede `Bearer Token`.
- **Parametri Path**: il {`noteId`} della nota da recuperare [`String`].

4.2.4 `PUT /notes/{noteId}`

- **Descrizione**: Aggiorna una nota esistente tramite il suo ID se l'utente autenticato è l'autore può modificare tutti i campi, mentre i collaboratori con permessi di scrittura possono modificare solo il `content`. Se la nota non ha cronologia viene creata la prima versione, altrimenti viene creata una nuova versione con le modifiche.
- **Metodo**: `PUT`.
- **Autenticazione**: Richiede `Bearer Token`.
- **Parametri Path**: il {`noteId`} della nota da aggiornare [`String`].
- **Body (raw JSON)**: Richiede un oggetto JSON contenente almeno uno di questi campi `title`, `content`, array di `collaborators` (ciascuno con un `user` ID e un `permission`) o un array di `tags` diversi da quelli attuali.

4.2.5 `DELETE /notes/{noteId}`

- **Descrizione**: Elimina una nota e tutte le sue versioni tramite il suo ID.
- **Metodo**: `DELETE`.
- **Autenticazione**: Richiede `Bearer Token`.
- **Parametri Path**: il {`noteId`} della nota da eliminare [`String`].

4.3 Cronologia Note (Versions)

Accessibili solo dall'autore della nota.

4.3.1 `GET /notes/{noteId}/versions?type={string}`

- **Descrizione**: Recupera la cronologia delle versioni per una specifica nota tramite il suo ID.
- **Metodo**: `GET`.
- **Autenticazione**: Richiede `Bearer Token`.
- **Parametri Path**: il {`noteId`} della nota [`String`].
- **Parametri Query**: `type={"delta"}` opzionale, default: "delta". Attualmente è l'unico valore supportato.

4.3.2 `GET /notes/{noteId}/versions/{version}?type={string}`

- **Descrizione**: Recupera una specifica versione di una nota tramite il suo ID e il numero di versione.
- **Metodo**: `GET`.
- **Autenticazione**: Richiede `Bearer Token`.
- **Parametri Path**: il {`noteId`} della nota [`String`] e il numero di {`version`} da recuperare [`Int`].
- **Parametri Query**: `type={"delta", "full"}` opzionale, default: "delta". "delta" restituisce solo le modifiche rispetto alla versione precedente, mentre "full" restituisce il contenuto completo della nota a quella versione.

4.3.3 `POST /notes/{noteId}/versions/{version}/revert`

- **Descrizione**: Ripristina una nota alla versione precedente specificata ed elimina tutte le versioni successive.
- **Metodo**: `POST`.
- **Autenticazione**: Richiede `Bearer Token`.
- **Parametri Path**: il {`noteId`} della nota [`String`] e il numero di {`version`} a cui ripristinare [`Int`].

4.3.4 `POST /notes/{noteId}/versions/{version}/rebase`

- **Descrizione**: Sposta la versione "0" della nota alla versione specificata, eliminando tutte le versioni precedenti. Non modifica la versione corrente della nota.
- **Metodo**: `POST`.
- **Autenticazione**: Richiede `Bearer Token`.
- **Parametri Path**: il {`noteId`} della nota [`String`] e il numero di {`version`} a cui effettuare il rebase [`Int`].

4.4 Gestione Utenti (Users)

4.4.1 `GET /users?q={string}`

- **Descrizione**: Ricerca utenti per username utilizzando una stringa di query.
- **Metodo**: `GET`.
- **Autenticazione**: Richiede `Bearer Token`.
- **Parametri Query**: `q={query}`, obbligatorio. Rappresenta la stringa di ricerca deve essere almeno di 2 caratteri.

4.4.2 `GET /users/me`

- **Descrizione**: Recupera il profilo dell'utente autenticato. Se viene fornito un file
- **Metodo**: `GET`.
- **Autenticazione**: Richiede `Bearer Token`.

4.4.3 `PUT /users/me`

- **Descrizione**: Aggiorna le informazioni del profilo dell'utente autenticato (username, email e avatar). Supporta la sostituzione o la rimozione dell'avatar, gestendo l'upload su *Cloudinary* e l'eventuale eliminazione dell'immagine precedente. Restituisce un errore se non viene effettuata alcuna modifica.
- **Metodo**: `PUT`.
- **Autenticazione**: Richiede `Bearer Token`.
- **Body (JSON o multipart/form-data)**: Richiede un `username` una `email` e opzionalmente un `avatar` che può essere un file o una stringa.

4.4.4 `DELETE /users?migrateNotes={boolean}`

- **Descrizione**: Elimina l'account dell'utente autenticato. Se il parametro `migrateNotes` è impostato, le note create dall'utente vengono migrate al primo collaboratore con permessi di scrittura, altrimenti tutte le note e le relative versioni vengono eliminate. L'utente viene inoltre rimosso come collaboratore dalle note di altri utenti, e il refresh token viene invalidato.
- **Metodo**: `DELETE`.

- **Autenticazione**: Richiede `Bearer Token`.
- **Parametri Query**: `migrateNotes`, opzionale. Se presente, attiva la migrazione delle note al primo collaboratore disponibile.

5 Componenti React

5.0.0.1 App Componente principale dell'applicazione. Inizializza la connessione WebSocket con il backend (`socket.io-client`). Applica il tema globale tramite `CustomThemeProvider`. Gestisce il routing con `BrowserRouter`. Fornisce contesto di autenticazione (`AuthProvider`). Carica le rotte definite in `AppRoutes`.

5.0.0.2 Main Componente che racchiude la struttura principale dell'applicazione.

5.0.0.3 AppRoutes Gestore centrale delle rotte. Definisce tutte le rotte principali con `react-router`. Utilizza `ProtectedRoute` per le pagine accessibili solo ad utenti autenticati (`Dashboard`, `MyProfile`, `ChangePassword`, `NoteEdit`, `NoteSettings`). Gestisce rotta "catch-all" (`*`) mostrando `Navbar` ed `ErrorBanner 404`.

5.1 Components

5.1.0.1 AddNoteFab Pulsante flottante per la creazione di una nuova nota. È sempre visibile in basso a destra della dashboard.

5.1.0.2 AvatarField Campo che permette di caricare, visualizzare e rimuovere un avatar utente. Mostra l'avatar corrente tramite `CustomAvatar`, con la possibilità di selezionare una nuova immagine da file locale o rimuoverla tramite un pulsante di chiusura. Gestisce interazioni utente legate all'immagine del profilo.

5.1.0.3 CustomAvatar `Avatar` di MUI personalizzato per la visualizzazione di un avatar che utilizzi il logo dell'applicazione come contenuto se non viene fornita un'immagine. Il colore di sfondo è generato dinamicamente dalla funzione `stringToColor`, che trasforma una stringa (tipicamente lo username) in un colore HSL stabile e unico. La funzione è esportata separatemente e viene utilizzata anche in altri componenti garantendo coerenza cromatica tra gli utenti.

5.1.0.4 DashboardBanner Banner informativo per la dashboard. Mostra opzionalmente un titolo e un messaggio, centrati all'interno di una card. Supporta contenuti aggiuntivi tramite `children`, per includere pulsanti o altri elementi UI a supporto del messaggio principale. Viene utilizzato per invitare a creare la prima nota quando l'elenco è vuoto.

5.1.0.5 DeleteDialog Dialogo di conferma per la cancellazione di un'entità. Mostra titolo, descrizione, eventuali contenuti extra e un campo di input per confermare l'operazione (ad esempio digitando il nome della risorsa). Il pulsante di eliminazione è disabilitato finché il valore inserito non corrisponde a quello richiesto.

5.1.0.6 ErrorBanner Banner che visualizza un messaggio di errore per la dashboard. Mostra informazioni provenienti dall'oggetto `error`, come codice di stato HTTP, codice generico o messaggio.

5.1.0.7 ErrorLog Lista animata capace di gestire più errori. Utilizza `Collapse` e `Transition Group` per mostrare e rimuovere dinamicamente gli errori.

5.1.0.8 FormField Campo di input con etichetta. Supporta sia input testuali standard che campi di tipo password (attraverso l'uso del componente `PasswordField`).

5.1.0.9 FormWrapper Contenitore per form centrati nella pagina. Visualizza un titolo, un logo opzionale e un pulsante di chiusura in alto a destra. Gli elementi figli (`children`) rappresentano i campi o le azioni del form.

5.1.0.10 Logo `Box` svg che mostra il logo dell'applicazione, si può personalizzare dimensione, colore e scegliere tra tre varianti: logotipo, logo e solo testo.

5.1.0.11 Navbar Barra di navigazione superiore della dashboard. Cliccare il logo consente di ricaricare le note. Se l'utente non è autenticato: mostra pulsanti `Login` e `Sign Up`. Se autenticato: mostra nome utente, avatar personalizzato e un `Popover` a comparsa per navigare verso `Profile` oppure fare `Logout`.

5.1.0.12 Note Card che rappresenta una singola nota. Mostra autore e collaboratori con permesso di scrittura con avatar cliccabili per visualizzare meglio l'avatar; titolo, contenuto, tag e data ultima modifica della nota. Se l'utente è autore mostra un pulsante per accedere alle impostazioni della nota. Cliccando sul contenuto si naviga all'edit della nota.

5.1.0.13 NoteList Contenitore che gestisce l'elenco delle note. In fase di caricamento mostra placeholder con `Skeleton`.

5.1.0.14 NoteTitleBar Componente che mostra il titolo della nota e permette di modificarla in base ai permessi dell'utente.

5.1.0.15 PasswordField Campo di input per password con possibilità di toggle della visibilità.

5.1.0.16 ProtectedRoute Componente che protegge le rotte che richiedono autenticazione reindirizzando alla pagina di login se l'utente non è autenticato.

5.1.0.17 UnsavedDialog Dialogo che avvisa quando l'ultimo utente presente nella nota tenta di navigare via da una nota con modifiche non salvate. Offre la possibilità di rimanere sulla pagina o procedere con la navigazione salvando oppure scartando le modifiche.

5.1.0.18 UserModal Modale che permette di visualizzare in grande l'avatar di un utente.

5.1.0.19 VersionModal Modale che permette di visualizzare una nota in una versione specifica.

5.1.1 NoteSettingsTabs

In questa cartella sono presenti i componenti che verranno renderizzati da NoteSettings

5.1.1.1 CollaboratorsTab Scheda che permette di gestire i collaboratori di una nota. Permette di cercare utenti tramite username, aggiungerli come collaboratori con permessi iniziali di sola lettura, e visualizzare la lista dei collaboratori già presenti. L'autore della nota può modificare i permessi (lettura/scrittura) o rimuovere un collaboratore.

5.1.1.2 NoteHistoryTab Scheda che mostra la cronologia delle versioni di una nota utilizzando una `Timeline` di Material UI. Ogni elemento della cronologia riporta l'autore della modifica, la data e l'ora di creazione, e consente di selezionare una versione per il revert o la visualizzazione. Il colore della versione è determinato dall'utente che ha effettuato la modifica.

5.1.1.3 TagsTab Scheda che consente di aggiungere, visualizzare e rimuovere tag associati a una nota. I tag possono essere inseriti tramite un campo di input e non possono superare i 30 caratteri. Ogni tag viene rappresentato come una `Chip` interattiva che può essere eliminata con un'azione diretta dell'utente.

5.2 Context

5.2.0.1 AuthContext & AuthProvider Contesto globale per la gestione dell'autenticazione e delle operazioni con l'API backend. Fornisce metodi per:

- Registrazione, login, logout e refresh del token di accesso.
- Gestione profilo utente (lettura, aggiornamento, cambio password, eliminazione con opzione di migrazione note).
- Operazioni sulle note (creazione, modifica, eliminazione, recupero) e sulle versioni delle note (cronologia, revert, rebase).
- Ricerca di altri utenti.

Integra un `axios interceptor` per il refresh automatico dei token scaduti. `useAuth()` è l'hook che permette l'accesso ai metodi e allo stato del contesto.

5.3 Hooks

5.3.0.1 useControlledNavigation Hook personalizzato che permette di intercettare e controllare la navigazione condizionata.

- Accetta un flag `shouldBlock` per bloccare temporaneamente la navigazione.
- Fornisce una funzione di navigazione che, se bloccata, salva la destinazione in sospeso.
- Espone metodi `confirm()` e `cancel()` per rispettivamente confermare o annullare la navigazione salvata.

5.4 Pages

5.4.0.1 ChangePassword Pagina che permette all'utente autenticato di cambiare la propria password.

5.4.0.2 Dashboard Pagina principale dell'applicazione, accessibile solo ad utenti autenticati.

5.4.0.3 Login Pagina di login per utenti esistenti.

5.4.0.4 MyProfile Pagina di gestione del profilo utente, accessibile solo ad utenti autenticati.

5.4.0.5 NoteEdit

Pagina di modifica in tempo reale di una nota, accessibile solo ad utenti autenticati che sono autori o collaboratori della nota.

5.4.0.6 NoteSettings

Pagina di gestione delle impostazioni di una nota, accessibile solo all'autore della nota.

5.4.0.7 SignUp

Pagina di registrazione per nuovi utenti.

5.5 Theme

5.5.0.1 CustomThemeProvider

Tema MUI personalizzato all'applicazione. Le principali personalizzazioni includono:

- Palette colori (testo, primario, errori, sfondi).
- Tipografia (font `Lexend` e `Tiny5` con pesi e dimensioni personalizzate).
- Stili globali per componenti MUI (`Tooltip`, `Popover`, `Button`, `IconButton`, `FormLabel`, `DialogTitle`).

Tutti i figli (`children`) sono resi all'interno del `ThemeProvider`.