Documentazione UniSocial

Corso di Fondamenti del Web – A.A. 2023/2024

Studenti: Lorenzo Verdura, Luca Cocito

1 Scenario applicativo

L'applicazione sviluppata rappresenta una **piattaforma sociale web** che consente agli utenti di creare un profilo, pubblicare contenuti testuali, interagire con i post tramite *like*, salvataggi e commenti, oltre che ricevere notifiche in tempo reale. L'obiettivo è simulare le principali funzionalità di un piccolo social network, con particolare attenzione alla gestione sicura dell'autenticazione, alla comunicazione client–server e all'aggiornamento in tempo reale dei dati.

Lo scenario applicativo prevede due principali attori:

- **Utente non autenticato:** può registrarsi o effettuare il login.
- **Utente autenticato:** può creare post, visualizzare i propri e quelli degli altri utenti, interagire con essi e ricevere notifiche in tempo reale.

2 Architettura dell'applicazione

L'applicazione è strutturata secondo l'architettura classica di una **Single Page Application (SPA)** basata su React per il frontend e Node.js/Express per il backend, con un database MongoDB per la persistenza dei dati. È inoltre previsto l'uso di **Socket.IO** per la gestione degli eventi in tempo reale, come le notifiche di like o commento.

2.1 Componenti principali

- **Frontend:** realizzato in React, si occupa della gestione delle interfacce, della navigazione client-side e delle chiamate alle API REST mediante Axios.
- **Backend:** realizzato in Node.js con Express, fornisce le API RESTful per l'autenticazione, la gestione dei post e delle interazioni. Utilizza *jsonwebtoken* per la generazione dei token e *cookie HttpOnly* per la sicurezza delle sessioni.
- **Database:** MongoDB, ospitato in cloud (MongoDB Atlas), utilizzato per memorizzare utenti, post, commenti e notifiche.
- **Socket Server:** integrato nel backend per gestire in tempo reale eventi come notifiche o aggiornamenti dinamici.

3 Diagramma UML dei casi d'uso

Figure 1: Diagramma UML dei casi d'uso principali dell'applicazione

Nel diagramma sono evidenziati i seguenti casi d'uso principali:

- Registrazione e login utente
- Creazione, modifica e cancellazione di un post
- Interazione con i post (like, salvataggio, commento)
- Visualizzazione profilo e post personali
- Ricezione notifiche in tempo reale

4 Modello dei dati

Il modello dei dati è stato progettato per garantire coerenza, scalabilità e semplicità di estensione.

4.1 Entità principali

- **User**
  - `_id`: identificativo univoco
  - `username`, `email`, `password (hash)`
  - `createdAt`, `updatedAt`
- **Post**
  - `_id`, `authorId`, `content`, `imageUrl`
  - `likes`: lista di utenti che hanno messo like
  - `savedBy`: lista di utenti che hanno salvato il post
  - `createdAt`
- **Comment**
  - `_id`, `postId`, `authorId`, `text`, `createdAt`
- **Notification**
  - `_id`, `recipientId`, `senderId`, `type` (like, comment, follow), `read`, `createdAt`

Le entità sono correlate in modo da permettere l'agevole gestione delle relazioni tra utenti e post, pur mantenendo la flessibilità tipica dei database non relazionali.

5 Documentazione delle API (Backend)

Le principali API REST sono organizzate come segue:

5.1 Autenticazione (/api/auth)

- `POST /register` – Crea un nuovo utente (richiede username, email e password)
- `POST /login` – Autentica un utente e genera un token JWT
- `GET /logout` – Termina la sessione rimuovendo il cookie di autenticazione

5.2 Utenti (/api/users)

- `GET /profile` – Restituisce le informazioni dell'utente autenticato
- `GET /:id` – Restituisce i dati pubblici di un utente specifico

5.3 Post (/api/posts)

- `GET /` – Restituisce tutti i post visibili all'utente
- `POST /create` – Crea un nuovo post
- `PUT /:id/like` – Mette o rimuove un like
- `PUT /:id/save` – Salva o rimuove il post dai preferiti
- `DELETE /:id` – Elimina un post

5.4 Commenti (/api/comments)

- `POST /:postId` – Aggiunge un commento a un post
- `GET /:postId` – Restituisce i commenti associati a un post

5.5 Notifiche (/api/notifications)

- `GET /` – Restituisce tutte le notifiche dell'utente
- `PUT /:id/read` – Segna una notifica come letta

Tutte le rotte autenticate sono protette da middleware di verifica del token JWT e da cookie HttpOnly per garantire un elevato livello di sicurezza.

6 Componenti React (Frontend)

Il frontend è strutturato in componenti modulari, ciascuno con un ruolo ben definito.

6.1 Componenti principali

- **App.jsx**: entry point dell'applicazione e gestione del routing.
- **AuthContext.jsx**: fornisce il contesto di autenticazione a tutti i componenti.
- **SocketContext.jsx**: gestisce la connessione Socket.IO per la ricezione di notifiche in tempo reale.
- **HomePage.jsx**: mostra il feed dei post e consente di interagire con essi.
- **ProfilePage.jsx**: visualizza i post creati dall'utente loggato.
- **NotificationPage.jsx**: mostra le notifiche ricevute, aggiornate in tempo reale.
- **PostModal.jsx**: gestisce la creazione di nuovi post.
- **NavBar.jsx** e **SideBar.jsx**: componenti grafici per la navigazione.

Ogni componente comunica con il backend tramite Axios e sfrutta gli *hook* di React per la gestione dello stato e degli effetti collaterali.