Relazione del Progetto: LoHobbies

Iacovone Vincenzo, Ostello Angelo, Zizzari Giuseppe

1 Scenario Applicativo e Visione del Progetto

LoHobbies è una piattaforma web moderna e interattiva, concepita come un hub centrale per gli amanti del cinema e delle serie televisive. L'obiettivo è superare la semplice catalogazione di contenuti, offrendo un'esperienza utente ricca e personalizzata. L'applicazione permette di scoprire nuovi titoli in linea con i propri gusti, comporre un diario dettagliato delle proprie visioni e condividere le proprie passioni con una community.

1.1 Funzionalità Chiave

Scoperta Intelligente: Oltre a mostrare i titoli di tendenza, LoHobbies fornisce una sezione "Esplora" con filtri avanzati (per genere, anno di uscita, popolarità, valutazione) per aiutare gli utenti a trovare esattamente ciò che cercano.

Gestione delle Liste Personali:

- **Watchlist**: Funge da promemoria per i contenuti che l'utente vuole vedere.
- **Diario**: Permette di costruire uno storico delle proprie visioni. Ogni voce può essere arricchita con una valutazione (da 1 a 5 stelle), un commento personale e la data di visione, trasformando il diario in un vero e proprio archivio personale.
- **Preferiti**: Una schermata che contiene tutti i nostri titoli preferiti aggiunti cliccando sul bottone a forma di cuore.

Autenticazione Sicura e Profilo Utente: Il sistema di autenticazione è basato su token (JWT), con meccanismi di refresh automatico per garantire una sessione fluida e sicura. Gli utenti possono personalizzare il proprio profilo con un nome e una foto profilo, nel caso in cui non fosse presente una foto profilo caricata dall'utente ci sarà un placeholder con la prima lettera maiuscola del nome utente.
Pagine di Dettaglio Complete: Ogni media (film o serie TV) e persona (attore, regista) ha una pagina dedicata che aggrega tutte le informazioni rilevanti: trama, cast, produzioni correlate, trailer ufficiali (tramite modale) e raccomandazioni basate su algoritmi.
Menù impostazioni: composto da diverse voci che ci permettono di cambiare la lingua dell'applicazione (italiano e inglese), il tema dell'applicazione (white-mode e dark-mode), la possibilità di cambiare password e di aggiornare la foto profilo.

2 Architettura dell'Applicazione

L'architettura è basata sul modello a tre livelli (Three-Tier Architecture), che separa la presentazione, la logica applicativa e la gestione dei dati.

- **Frontend (Livello di Presentazione):**

  Framework: React (v19)

  Routing: react-router-dom per la navigazione client-side in una Single Page Application.

  State Management: React Context API (AuthContext, FilterContext, ThemeContext) per la gestione dello stato globale (autenticazione, filtri, tema).

  Chiamate API: axios per le comunicazioni HTTP con il backend, con un'istanza pre-configurata e interceptor per la gestione automatica del refresh dei token.

  Styling: CSS puro con un approccio modulare per componente.

- **Backend (Livello Logico):**

  Framework: Node.js con Express.js per la creazione di un'API RESTful robusta.

  Comunicazione Real-time: socket.io per la gestione dei commenti in tempo reale.

  Gestione Asincrona: Utilizzo di async/await per operazioni non bloccanti, specialmente nelle interazioni con il database e le API esterne.

  Sicurezza: Middleware authMiddleware per proteggere gli endpoint che richiedono autenticazione tramite la verifica di JSON Web Tokens (JWT).

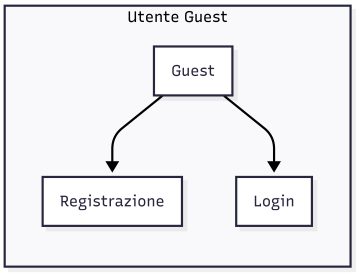
  Integrazione Esterna: Un modulo dedicato (tmdb.js) gestisce tutte le chiamate verso l'API di The Movie Database (TMDB) per l'arricchimento dei dati.

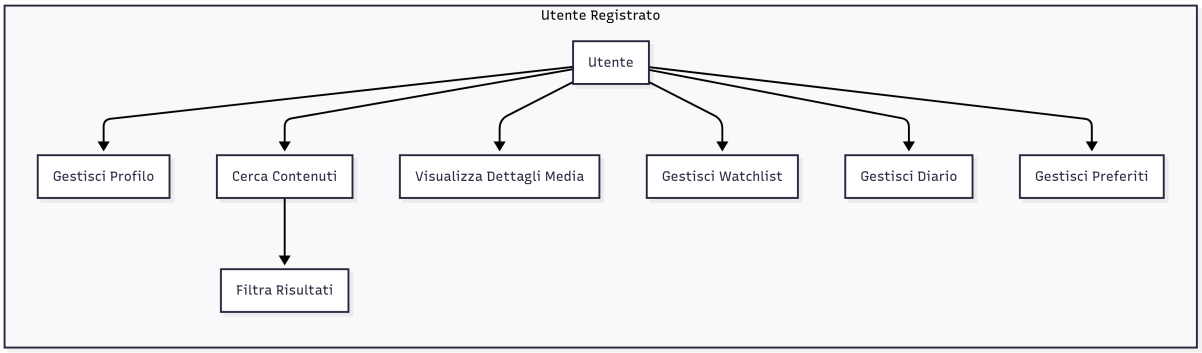
- **Database (Livello Dati):** Sistema: MongoDB, un database NoSQL a documenti che offre flessibilità per la gestione dei dati degli utenti.

  ODM: Mongoose per la modellazione degli oggetti di business (schema User) e la gestione delle interazioni con il database.

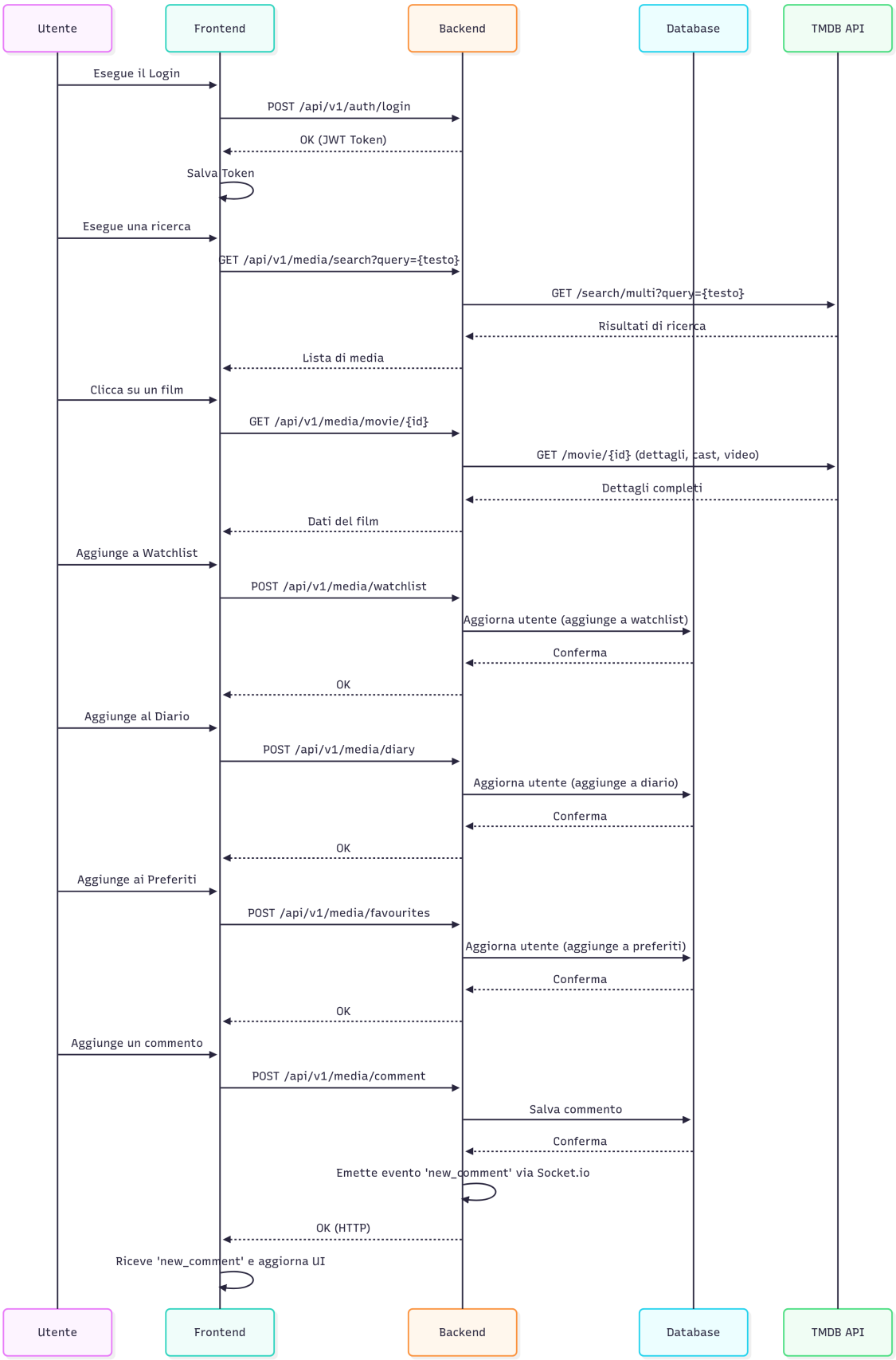
3 Diagrammi UML

3.1 Diagramma dei Casi d'Uso





3.2 Diagramma di sequenza compatto



4 Modello dei Dati

Il database MongoDB utilizza una singola collection 'users' per memorizzare tutte le informazioni.
Schema User (`LoHobbies/backend/models/User.js`)

- **name**: (String, Required) - Nome visualizzato dell'utente.
- **email**: (String, Required, Unique) - Identificativo univoco per l'autenticazione.
- **password**: (String, Required, Select: false) - Hash della password (non restituito di default nelle query).
- **refreshToken**: (String) - Token utilizzato per rinnovare la sessione senza richiedere un nuovo login.
- **profilePicture**: (String) - Path dell'immagine del profilo caricata sul server.
- **watchlist, diary, favourites**: (Array di Sub-documenti) - Ognuno di questi array contiene oggetti con una struttura definita, che include non solo l'ID del media ma anche metadati come title e posterPath per ridurre la necessità di chiamate API aggiuntive quando si visualizzano le liste.

  1. Sottodocumento watchlist e favourites: Contengono i campi base per identificare il media (mediaId, mediaType, title, posterPath, releaseDate) e la data di aggiunta (addedAt). La struttura è ottimizzata per una rapida visualizzazione delle liste.
  2. Sottodocumento diary: Oltre ai campi base, include campi aggiuntivi come rating (Number), comment (String) e watchedDate (Date) per una registrazione dettagliata della visione.
  3. timestamps: (Boolean: true) Aggiunge automaticamente i campi createdAt e updatedAt a ogni documento.

5 Documentazione delle API

Questa sezione offre una descrizione dettagliata degli endpoin preferiti

5.1 Gestione dell'Autenticazione

Registrazione di un nuovo utente: Per creare un nuovo account, è necessario fornire un nome, un'email e una password.
Accesso (Login): Per accedere, l'utente deve fornire la propria email e password. Se le credenziali sono corrette, il sistema lo autentica, restituendo un token di accesso temporaneo (accessToken) e i dati dell'utente. Inoltre, viene impostato un cookie di sicurezza (refreshToken) per gestire sessioni più lunghe.
Verifica della sessione: All'avvio dell'applicazione, viene utilizzato un endpoint protetto per verificare che la sessione dell'utente sia ancora valida e per caricare i suoi dati.
Gestione dell'immagine del profilo: Gli utenti autenticati possono caricare un'immagine del profilo inviando un file. È anche possibile rimuovere l'immagine del profilo esistente.
Gestione della password: Sono disponibili due funzioni protette, una per verificare la password attuale dell'utente e un'altra per consentirgli di reimpostarla.

5.2 Gestione dei Media

Scoperta di contenuti: È possibile scoprire nuovi contenuti attraverso una ricerca filtrata. Ad esempio, si possono cercare film di un genere specifico e ordinarli per popolarità.
Ricerca generale: Una funzione permette di effettuare una ricerca libera tramite una parola chiave (ad esempio, "matrix"). Il sistema restituirà una lista di risultati che include film, serie TV e persone pertinenti.
Gestione delle liste personali: Gli utenti autenticati possono aggiungere film o serie TV alla propria "watchlist" (lista di contenuti da vedere), fornendo i dettagli del media. Allo stesso modo, possono rimuovere un elemento specifico dalla lista. La stessa logica si applica anche per la gestione del proprio diario (diary) e della lista dei preferiti (favourites).
Visualizzazione dei commenti: È possibile recuperare tutti i commenti associati a un film o a una serie TV specifica.

6 Componenti React

Questa sezione descrive la struttura dei componenti dell'interfaccia utente e come i dati vengono gestiti e scambiati tra di essi.

6.1 Flusso dei Dati e Gestione dello Stato

Per gestire i dati in modo efficiente ed evitare di passarli manualmente da un componente all'altro, l'applicazione utilizza un sistema centralizzato chiamato React Context API. Questo approccio organizza lo stato dell'applicazione in contesti specifici.
Gestione dell'Autenticazione (AuthContext): Questo contesto si occupa di tutte le informazioni relative all'utente. Rende disponibili i dati dell'utente (come il nome e l'email), indica se l'applicazione sta caricando dati, e fornisce le funzioni per effettuare il login e il logout. All'avvio dell'app, tenta di recuperare i dati dell'utente per mantenere attiva la sessione di accesso.
Gestione dei Filtri (FilterContext): Memorizza le preferenze di filtro selezionate dall'utente (ad esempio, genere o anno di uscita) nella finestra di dialogo dei filtri. Queste informazioni vengono poi usate nelle pagine "Esplora" e di ricerca per mostrare i risultati corretti.
Gestione del Tema Grafico (ThemeContext): Permette all'utente di passare da un tema all'altro, come la modalità chiara e quella scura. Questo contesto si occupa di applicare gli stili grafici appropriati a tutta l'applicazione.

6.2 Descrizione dei Componenti Chiave

Questi sono i mattoni fondamentali dell'interfaccia utente.
Header: È la barra di navigazione principale dell'applicazione. Include la barra di ricerca, che offre suggerimenti in tempo reale mentre l'utente digita, i collegamenti alle sezioni più importanti e un menu dedicato all'utente, da cui può accedere alle sue liste personali (contenuti da vedere, diario, preferiti) e alle impostazioni.
Carousel: Questo componente è un "carosello" versatile utilizzato per mostrare liste di film o serie TV in orizzontale. Viene impiegato in diverse parti del sito, come nella home page per i titoli di tendenza, nei risultati di ricerca e per le raccomandazioni personalizzate.
AuthModal: Si tratta della finestra a comparsa (modale) che gestisce sia la registrazione di nuovi utenti che l'accesso. Include un sistema di validazione che controlla i dati inseriti dall'utente in tempo reale, fornendo feedback immediato.
MediaPage: Questa è la pagina di dettaglio dedicata a un singolo film o serie TV. Carica e presenta tutte le informazioni pertinenti, come la trama, la lista degli attori (cast), i trailer e una sezione per i commenti degli utenti.
CommentModal: Una finestra a comparsa che permette agli utenti di scrivere e inviare nuovi commenti su un film o una serie. Grazie all'uso di una tecnologia specifica (socket.io), i commenti appaiono istantaneamente per tutti gli utenti senza bisogno di ricaricare la pagina.
FilterModal: Un'altra finestra a comparsa che offre opzioni di ricerca avanzate. È collegata alla sezione "Esplora" e consente agli utenti di affinare i risultati per genere, anno di uscita, popolarità e altri criteri.

7 Sicurezza e gestione degli errori

La stabilità e la sicurezza dell'applicazione sono state garantite attraverso un approccio completo che copre sia il backend (il server) che il frontend (l'interfaccia utente).

7.1 Misure di sicurezza Implementate

Per proteggere i dati degli utenti e garantire l'integrità del sistema, sono state adottate le seguenti misure:
Autenticazione Sicura con Token (JWT): Per gestire l'accesso degli utenti, viene utilizzato un sistema basato su token digitali. L'accesso alle aree protette avviene tramite un "Access Token" a breve scadenza, che riduce i rischi in caso di furto. Per mantenere la sessione attiva più a lungo, viene usato un "Refresh Token", memorizzato in un cookie speciale (httpOnly) che lo protegge da attacchi informatici comuni (come XSS) rendendolo inaccessibile a script malevoli. Crittografia delle Password: Le password degli utenti non sono mai salvate in chiaro. Vengono invece trasformate in un codice crittografato (hash)

tramite un algoritmo robusto e standard del settore (bcrypt) prima di essere memorizzate nel database. Questo garantisce che, anche in caso di accesso non autorizzato al database, le password originali rimangano segrete.
Protezione delle Funzionalità Riservate: Ogni funzionalità del backend che richiede l'autenticazione (come la modifica del profilo o l'aggiunta di un film alla watchlist) è protetta da un "controllore" (middleware). Questo componente verifica automaticamente la validità del token digitale inviato con ogni richiesta, bloccando qualsiasi tentativo di accesso non autorizzato.
Restrizione degli Accessi Esterni (CORS): Il server è configurato per accettare richieste solo dall'indirizzo web dell'applicazione stessa (il frontend). Questa misura, nota come CORS, impedisce a siti web esterni e potenzialmente dannosi di interagire con le API del sistema. **Validazione dei Dati in Ingresso**: Tutti i dati inviati dagli utenti (come nei moduli di registrazione o di ricerca) vengono controllati e validati per prevenire l'inserimento di codice malevolo e altri attacchi comuni.

7.2 Gestione degli Errori e dell'Affidabilità

Per garantire un'esperienza utente fluida e un sistema stabile, è stato implementato un solido sistema di gestione degli errori.
Lato Server (Backend): Il server è dotato di un sistema di sicurezza che cattura qualsiasi errore imprevisto. In caso di problemi, l'errore viene registrato per future analisi e il server viene riavviato in modo controllato per evitare che l'applicazione diventi instabile. Inoltre, il server comunica in modo standardizzato con il client usando codici di stato HTTP (come "404 Pagina non trovata" o "401 Non autorizzato"), per far capire la natura di ogni problema.
Lato Client (Frontend): L'interfaccia utente utilizza un meccanismo automatico per intercettare e gestire gli errori provenienti dal server. Ad esempio, se la sessione di un utente scade (errore 401), il sistema tenta autonomamente di rinnovarla senza interrompere l'utente. Per altri tipi di errore, vengono mostrate notifiche chiare e non invasive, garantendo un'esperienza di navigazione fluida e senza frustrazioni.

8 Flusso di Dati dettagliato nel Frontend

L'architettura del frontend è progettata per essere chiara e prevedibile, seguendo un flusso di dati unidirezionale, uno dei principi fondamentali di React.

8.1 Avvio dell'Applicazione

Quando un utente apre l'applicazione per la prima volta, si verificano i seguenti passaggi:
Preparazione: La componente principale dell'app carica e attiva i sistemi di gestione centralizzati, come quello per l'autenticazione, i filtri e il tema grafico.
Verifica della Sessione: Il sistema di autenticazione verifica immediatamente se l'utente ha una sessione di accesso ancora valida. Per farlo, invia una richiesta al server utilizzando un cookie di sicurezza (refresh token) memorizzato in precedenza. Caricamento dei Dati: Se la sessione è valida, il server risponde inviando i dati dell'utente (nome, email, liste personali, ecc.). Queste informazioni vengono salvate nello stato globale dell'applicazione, rendendole disponibili a tutte le componenti dell'interfaccia. In questo modo, l'utente rimane connesso senza dover effettuare nuovamente il login.

8.2 Esempio di Interazione: Aggiunta di un Film al Diario

Per capire come i dati si muovono durante l'uso, vediamo cosa succede quando un utente aggiunge un film al proprio diario personale: **Azione dell'Utente**: L'utente si trova sulla pagina di dettaglio di un film e clicca sul pulsante "Aggiungi al diario".
Inserimento Dati: Si apre una finestra di dialogo (modale) che chiede all'utente di inserire una valutazione e un commento personale per quel film.
Invio al Server: Una volta che l'utente clicca su "Salva", l'applicazione invia una richiesta al server contenente l'ID del film e i dati inseriti (valutazione e commento). A questa richiesta viene automaticamente allegato il token di autenticazione per confermare l'identità dell'utente.
Elaborazione del Server: Il server riceve la richiesta, la elabora e salva le nuove informazioni nel database, associandole al profilo dell'utente. Successivamente, invia una risposta di successo all'applicazione.
Aggiornamento dell'Interfaccia: Una volta che l'applicazione riceve la conferma dal server, aggiorna l'interfaccia per riflettere il cambiamento. Questo avviene in modo intelligente, ad esempio ricaricando

solo la lista del diario o, per una maggiore reattività, aggiungendo direttamente il nuovo elemento alla visualizzazione corrente senza bisogno di ulteriori richieste al server.
Feedback Visivo: Infine, viene mostrato un messaggio di successo all'utente (ad esempio, "Film aggiunto al diario!"). Questo approccio garantisce che l'interfaccia utente sia sempre sincronizzata con i dati sul server, offrendo un'esperienza fluida e reattiva.

8.3 Comunicazione in Tempo Reale con Socket.io (per i commenti)

Per funzionalità come la sezione dei commenti, l'interazione è ancora più dinamica.
Quando un utente invia un nuovo commento, la procedura è simile a quella descritta sopra cioè la richiesta viene inviata al server, che salva il commento nel database.
Tuttavia, il server non si limita a confermare l'operazione. Utilizzando una tecnologia speciale (Socket.io), invia una notifica istantanea a tutti gli utenti che stanno visualizzando quella pagina in quel momento.
L'applicazione di ogni utente è costantemente in ascolto di questi messaggi. Appena riceve la notifica del nuovo commento, lo aggiunge immediatamente alla lista dei commenti visibili, il tutto senza che l'utente debba ricaricare la pagina.

9 Deployment

L'applicazione è stata resa operativa e accessibile al pubblico attraverso un'infrastruttura cloud ospitata su Google Cloud Platform. È stato scelto un approccio moderno che mantiene separati il frontend (l'interfaccia utente) e il backend (la logica del server), una strategia che garantisce maggiore flessibilità per la manutenzione e la crescita futura dell'applicazione.

9.1 Configurazione dell'Ambiente Server

Per ospitare l'applicazione, è stato configurato un ambiente server specifico composto da:

- Server Virtuale: È stato creato un server virtuale su Google Compute Engine, che funziona con un sistema operativo Linux (Ubuntu), noto per la sua stabilità e sicurezza.
- Software Essenziale: Sul server sono stati installati i programmi necessari per far funzionare l'applicazione: Node.js (l'ambiente che esegue il codice del backend), npm (per gestire le librerie software).
- Database Esterno: Il database non risiede direttamente sul server, ma è affidato a MongoDB Atlas, un servizio specializzato e gestito. Questa scelta offre vantaggi significativi, come l'alta affidabilità, la garanzia di disponibilità continua e i backup automatici, sollevando il team dalla gestione diretta del database.

9.2 Processo di Installazione del Backend

L'installazione del backend sul server segue una procedura standard e sicura:

- Recupero del Codice: Il codice sorgente del backend viene scaricato sul server dal suo repository online (Git).
- Installazione delle Librerie: Tramite un apposito comando (npm install), vengono installate tutte le librerie e i pacchetti software di cui il backend ha bisogno per funzionare.
- Configurazione Sicura: Viene creato un file di configurazione separato (.env) per memorizzare tutte le informazioni sensibili, come la chiave di connessione al database MongoDB Atlas e le chiavi segrete per la gestione della sicurezza (JWT). Questo assicura che i dati critici non siano mai esposti direttamente nel codice sorgente.

9.3 Tecnologie e Librerie Chiave del Backend

Il backend è costruito utilizzando un insieme di librerie software (pacchetti Node.js) consolidate e affidabili, ognuna con un compito specifico:

- **Express**: È lo scheletro dell'applicazione, il framework principale utilizzato per costruire l'API e gestire le richieste web.
- **Mongoose**: Serve a semplificare la comunicazione tra l'applicazione e il database MongoDB.
- **Bcryptjs**: Si occupa di crittografare in modo sicuro le password degli utenti.
- **Jsonwebtoken (JWT)**: Gestisce la creazione e la verifica dei token di sicurezza usati per l'autenticazione.
- **Cookie-parser**: Permette al server di leggere e scrivere i cookie nel browser dell'utente, fondamentale per il sistema di sessione sicura.
- **CORS**: Una libreria di sicurezza che garantisce che solo l'interfaccia utente autorizzata possa comunicare con il backend.
- **Dotenv**: Carica le informazioni sensibili dal file di configurazione .env in modo sicuro.
- **Multer**: Gestisce il caricamento di file, come le immagini del profilo degli utenti.
- **Socket.io**: Abilita la comunicazione in tempo reale, essenziale per funzionalità come la visualizzazione istantanea dei nuovi commenti.

10 Monitoraggio e Analytics

Per comprendere il comportamento degli utenti e migliorare l'esperienza sulla piattaforma, LoHobbies integra Google Analytics 4 (GA4), una soluzione di analisi di nuova generazione che permette di misurare il traffico e l'engagement su siti web e applicazioni.

10.1 Configurazione e integrazione

Inizializzazione: Google Analytics è stato integrato nel frontend React tramite il pacchetto react-ga4. La configurazione viene inizializzata nel file principale dell'applicazione (App.js), utilizzando un Measurement ID univoco fornito da Google.
Tracciamento delle Pagine Viste: react-ga4 traccia automaticamente le visualizzazioni di pagina ogni volta che la cronologia di navigazione del browser cambia, grazie all'integrazione con react-router-dom. Questo permette di raccogliere dati su quali pagine vengono visitate più frequentemente.
Core Web Vitals: Oltre al tracciamento delle pagine viste, l'integrazione è stata configurata per inviare a Google Analytics i dati relativi ai Core Web Vitals.
Questi indicatori, misurati tramite la libreria web-vitals, forniscono insight cruciali sulle performance reali dell'applicazione dal punto di vista dell'utente:
LCP (Largest Contentful Paint): Misura la velocità di caricamento, indicando quando l'elemento più grande della pagina diventa visibile.
FID (First Input Delay): Misura l'interattività, calcolando il tempo che intercorre tra la prima interazione dell'utente e la risposta del browser.
CLS (Cumulative Layout Shift): Misura la stabilità visiva, quantificando gli spostamenti inaspettati degli elementi della pagina durante il caricamento.
TTFB (Time to First Byte): Misura il tempo che intercorre tra la richiesta di una risorsa e l'arrivo del primo byte della risposta.
FCP (First Contentful Paint): Misura il momento in cui il primo contenuto (testo, immagine, ecc.) viene visualizzato sullo schermo.
Questi dati permettono di identificare e risolvere problemi di performance che potrebbero compromettere l'esperienza utente.

10.2 Limiti dell'integrazione attuale e Prospettive future

Al momento, la nostra analisi si concentra principalmente sul tracciamento automatico delle pagine visitate. Sebbene questi dati siano molto utili, non ci forniscono dettagli su azioni specifiche compiute dagli utenti, come il numero di accessi, le registrazioni o quante volte un film viene aggiunto a una lista personale. Per evolvere e prendere decisioni basate su dati ancora più precisi, il prossimo passo sarà implementare il tracciamento di eventi personalizzati. Questo ci permetterà di rispondere a domande

specifiche come: "Quante volte un utente aggiunge un film ai preferiti?" o "Quali sono le funzionalità più cliccate nel menu utente?". La raccolta di questi dati granulari ci aiuterà a capire quali sono le funzionalità più amate, a guidare le priorità di sviluppo future e, in definitiva, a ottimizzare l'esperienza per tutti i nostri utenti.