

UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea in Ingegneria Informatica



Elaborato Esame -**WEB AND REAL TIME COMMUNICATION SYSTEMS**

Talk-Alot 

Professore: **Simon Pietro Romano**

Studente: **Alberto Urraro**
Matricola: **M63001156**

Anno Accademico: **2021/2022**

Indice

INTRODUZIONE	3
Struttura Applicazione	4
TECNOLOGIE UTILIZZATE	5
ReactJS.....	5
Caratteristiche:.....	5
Funzionamento:	5
ExpressJS	6
Caratteristiche:.....	6
NodeJS	6
MongoDB.....	7
Caratteristiche:.....	7
Funzionamento:	7
SendGrid.....	7
Chakra-UI.....	8
Visual Paradigm	8
PostMan.....	8
Cloudinary	8
PROGETTAZIONE	9
Requisiti Funzionali	9
Casi D'Uso:.....	9
Attori	9
Casi D'Uso – Breve Descrizione	9
REALIZZAZIONE	10
FRONT-END	10
BACK-END	12
PostMan.....	13
DATABASE.....	14
SVILUPPI FUTURI	15

INTRODUZIONE

Talk-Alot è una web-application che nasce con l'intento di poter permettere la comunicazione tra specialisti in campo medico dislocati in posti diversi nel mondo, una piccola nicchia dove poter esprimere le proprie opinioni relative a studi scientifici, diagnosi, casi di studio ecc. e perché no conoscere persone di cultura diversa e stringere nuove amicizie.

L'obiettivo di questa chat è confrontarsi scambiare informazioni, idee, aprire discussioni costruttive in ambito medico. L'utente per poter usufruire del servizio deve necessariamente registrarsi. Una volta completata la registrazione l'utente verrà indirizzato alla propria area privata, da lì sarà in grado di poter cercare altri utenti presenti sulla web-app e iniziare a conversare in una chat dedicata, è presente anche la possibilità di creare chat di gruppo, l'admin può aggiungere partecipanti rimuoverli e modificare il nome della chat di gruppo.

L'utente registrato ha anche la possibilità nel caso in cui non volesse più usufruire del servizio di eliminare il proprio account.

The image displays two side-by-side screenshots of the Talk-Alot web application interface, illustrating the login and registration processes.

Left Screenshot (Login Page):

- Header: "Talk-Alot" logo.
- Buttons: "Accedi" (Login) and "Registrati" (Register).
- Form Fields:
 - Email * (Inserisci l'email)
 - Password * (Inserisci la password)
- Buttons: "Accedi" (Login)
- Link: "Password dimenticata?" (Forgot Password)

Right Screenshot (Registration Page):

- Header: "Talk-Alot" logo.
- Buttons: "Accedi" (Login) and "Registrati" (Register).
- Form Fields:
 - Nickname * (Inserisci il tuo Nickname)
 - Email * (Inserisci la tua email)
 - Password * (Inserisci la tua password)
 - Conferma Password * (Conferma la tua password)
 - Seleziona la tua immagine (Scegli file | Nessun file selezionato)
- Buttons: "Registrati" (Register)

Figura 1 – Login e registrazione

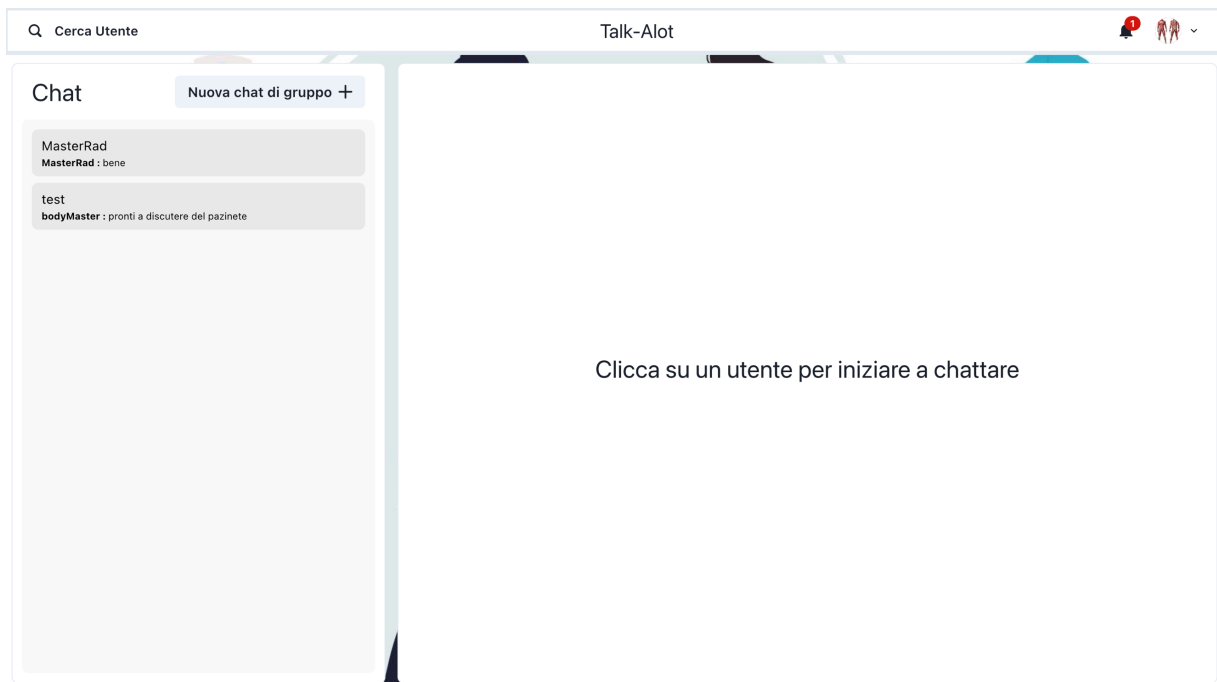


Figura 2 – Area privata

Struttura Applicazione

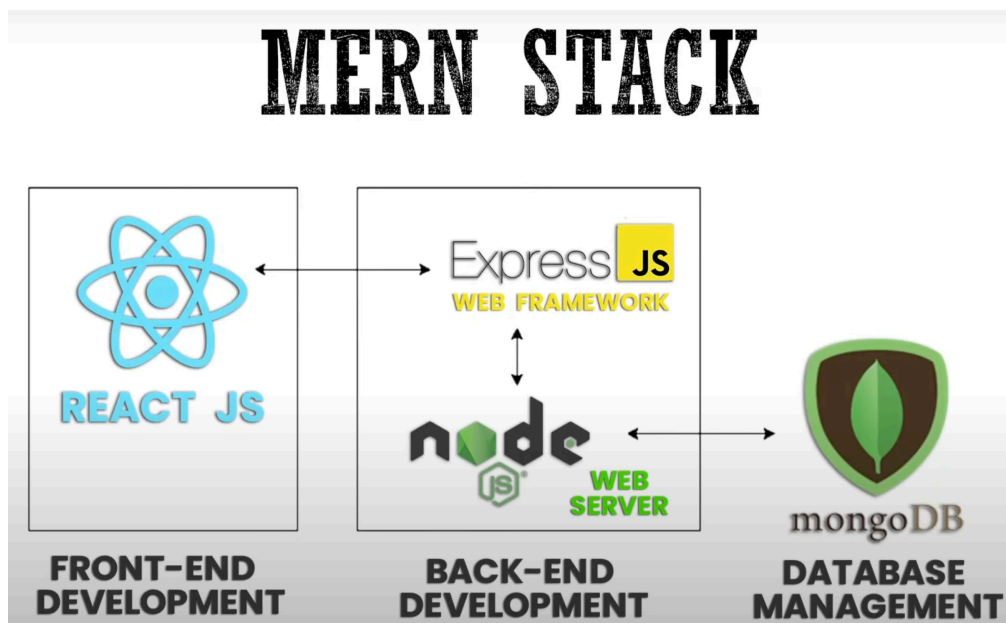
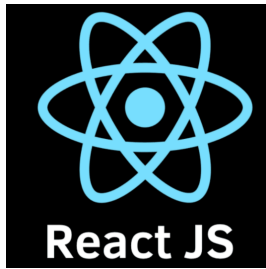


Figura 3- Struttura

Com'è possibile notare anche dalla figura MERN è uno stack software formato da diversi componenti per la creazione di Web Application.

TECNOLOGIE UTILIZZATE

ReactJS



ReactJS è una libreria open-source, front-end, JavaScript per la creazione di interfacce utente, può essere utilizzato come base nello sviluppo di applicazioni a pagina singola ma è utilizzabile anche su mobile tramite **React Native**,

Figura 4 – ReactJs

Caratteristiche:

- React si occupa solo del rendering dei dati sul **Document Object Model(DOM)**, pertanto la creazione di applicazioni React richiede generalmente l'uso di librerie aggiuntive per lo state management e il routing
- I **componenti** sono componibili e riutilizzabili. Ciò vuol dire che un componente può contenere al suo interno altri componenti e non tutti i componenti sono uguali.
- si può utilizzare JavaScript XML JSX cioè una estensione di JavaScript che serve per aggiungere elementi React al DOM senza usare createElement o appendChild

Funzionamento:

- ReactJs è costruito attorno alla creazione di funzioni, che prendono gli aggiornamenti di stato dalla pagina e li traducono in una rappresentazione virtuale della pagina risultante. Ogni volta che React viene informato di un cambiamento di stato, riesegue quelle funzioni per determinare un nuovo rendering virtuale della pagina, il tutto avviene tramite l'uso di un **DOM virtuale** un'astrazione del DOM. Si tratta di una rappresentazione in memoria del DOM, veloce e indipendente dalle specifiche implementazioni del browser, quando avviene una variazione dei dati all'interno dell'applicazione, React effettua le modifiche sul Virtual DOM e lo aggiorna per rispecchiare i cambiamenti avvenuti, calcola successivamente la differenza tra le due rappresentazioni del **Virtual DOM**, ovvero fra la rappresentazione del **Virtual DOM** prima che i dati venissero modificati e l'attuale rappresentazione del **Virtual DOM** (dopo la modifica dei dati all'interno dell'applicazione). La differenza tra le due rappresentazioni del **Virtual DOM**, è ciò che deve essere cambiato nel DOM reale. A questo punto, React effettua le modifiche nel DOM reale, aggiornando **solo ed esclusivamente** quello che deve essere cambiato.

ExpressJS



ExpressJs, o semplicemente **Express**, è un framework di applicazioni Web back-end per Node.js. È progettato per la creazione di applicazioni Web e API

Figura 5 – ExpressJS

Caratteristiche:

- Prestazioni elevate
- Migliora e semplifica l'utilizzo del modulo http, consente di gestire facilmente il routing delle richieste e l'accesso al file system

NodeJS



NodeJS nasce per estendere l'approccio di JavaScript anche al di fuori del browser e sfrutta il framework CommonJS che standardizza le interfacce di programmazione JavaScript che comprendono la modularizzazione. Esso è un ambiente che offre supporto per JavaScript a tempo di esecuzione ed è stato reso disponibile sul motore V8 di Chrome.

Figura 6 - NodeJS

NodeJS è una piattaforma event-driven, è leggero ed efficiente poiché si basa su un modello di I/O non bloccante e asincrono. Utilizza le librerie *libuv* che gestiscono il threading attraverso un pool di thread che interrogano in maniera circolare le entità interessate a interagire con il framework stabilendo quali sono quelle pronte ad essere servite.

In molti sistemi tutte le system calls sono bloccanti in quanto l'esecuzione si ferma in attesa del risultato (blocking I/O sincrono). I moderni server invece, sono multi-thread, e quindi generano un thread per ogni richiesta ricevuta e, per ogni richiesta in attesa di risposta il thread viene messo in sleep, ma continua a consumare risorse del sistema. Quindi ogni richiesta avrà un thread pendente che consuma CPU e memoria.

Per risolvere questo problema, NodeJS utilizza un approccio differente servendo tutte le richieste con un unico thread. Il codice in questo thread è sempre eseguito in maniera sincrona: ad ogni system call, essa viene delegata all'event-loop assieme a una funzione di callback. Il thread principale non va in sleep e continua a servire richieste successive. Appena la precedente system call viene completata, l'event-loop esegue la funzione di callback associata, e restituisce un valore. Pertanto, il programma principale non viene bloccato dalle operazioni di I/O.

MongoDB



mongoDB

MongoDB è un sistema di gestione di database distribuito open source. È un potente e flessibile **NoSQL** database (non relazionali) che memorizza i dati in documenti simili a **JSON**.

Figura 7 - MongoDB

Caratteristiche:

- **Query ad hoc:** MongoDB supporta ricerche per campi, intervalli e regular expression. Le query possono restituire campi specifici del documento e anche includere funzioni definite dall'utente in JavaScript.
- **Indicizzazione:** Qualunque campo in MongoDB può essere indicizzato (gli indici in MongoDB sono concettualmente simili a quelli dei tradizionali RDBMS). Sono disponibili anche indici secondari, indici unici, indici sparsi e indici full text.
- **Alta Affidabilità:** MongoDB fornisce alta disponibilità e aumento del carico gestito attraverso i replica set. Un replica-set consiste in due o più copie dei dati.

Funzionamento:

- **MongoDB** è strutturato su un modello client-server in cui un demone del server accetta connessioni dai client ed elabora le azioni del database da questi. Il server deve essere in esecuzione affinché i client possano connettersi e interagire con i database.

SendGrid



SendGrid fornisce un servizio di consegna della posta elettronica basato su cloud per la consegna della posta elettronica. Il servizio gestisce vari tipi di e-mail e fornisce il monitoraggio dei collegamenti. Consente inoltre di tenere traccia di aperture, annullamenti di iscrizioni, rimbalzi e rapporti di spam di posta elettronica.

Figura 8 - SendGrid

Chakra-UI



Chakra UI è una libreria di componenti React con accessibilità integrata. Fornisce una serie di componenti React accessibili, riutilizzabili e componibili necessari per creare applicazioni front-end.

Figura 9 - Chakra

Visual Paradigm



Visual Paradigm

Visual Paradigm è uno strumento che supporta UML per la realizzazione di diagrammi di progetto e la progettazione di applicazioni e segmenti di codice. Oltre al supporto per la modellazione, fornisce funzionalità di generazione di report e ingegneria del codice, inclusa la generazione del codice stesso

Figura 10 – Visual Paradigm

PostMan



Postman è un'applicazione utilizzata per il test delle API. È un client HTTP che testa le richieste HTTP, utilizzando un'interfaccia utente grafica, attraverso la quale otteniamo diversi tipi di risposte che devono essere successivamente validate

POSTMAN

Figura 11 – PostMan

Cloudinary



Cloudinary è una soluzione di gestione delle immagini end-to-end per siti Web e app mobili. L'app supporta il caricamento di immagini, l'archiviazione su cloud. Cloudinary offre API e funzionalità di amministrazione flessibili per l'integrazione con applicazioni Web e mobili nuove ed esistenti

Figura 12 - Cloudinary

PROGETTAZIONE

Requisiti Funzionali

Casi D'Uso:

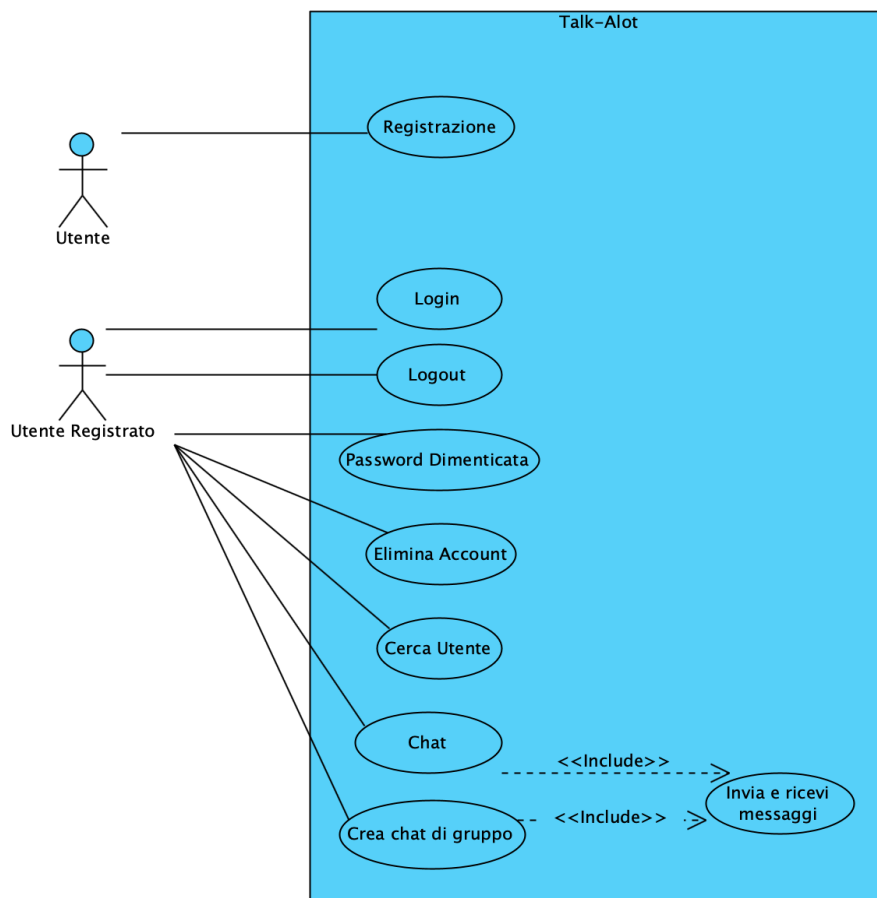


Figura 13 – Casi D'uso

Attori

- **Utente**
- **Utente-Registrato**

Casi D'Uso – Breve Descrizione

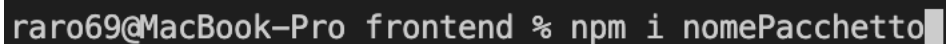
- **Registrazione:** permette all'utente di accedere alla web-app
- **Login:** permette all'utente registrato di accedere alla propria area privata
- **Logout:** una volta effettuato il login, permette di disconnettersi successivamente.
- **Password Dimenticata:** consente di modificare la password dell'utente registrato qualora non la ricordi
- **Elimina Account:** consente all'utente registrato di eliminare il proprio account dalla web-app
- **Cerca Utente:** consente all'utente registrato di poter cercare altri utenti presenti sulla web-app per iniziare ad interagire
- **Chat:** l'utente registrato una volta ricercato un utente può iniziare una conversazione in una chat dedicata
- **Crea chat di gruppo:** l'utente registrato ha la possibilità di creare una chat di gruppo

REALIZZAZIONE

FRONT-END

Prima di andare a vedere com'è strutturato il front-end è opportuno elencare tutti i moduli che sono stati installati, e successivamente utilizzati, tramite il **Node Package Manager** ovvero un gestore di pacchetti per JavaScript.


Per l'installazione di qualunque tipo di pacchetto bisogna lanciare il seguente comando:



```
raro69@MacBook-Pro frontend % npm i nomePacchetto
```

Figura 14 – npm command

I pacchetti installati nell'ambito del front-end sono i seguenti:



```
"@chakra-ui/icons": "^1.1.1",  
"@chakra-ui/react": "^1.7.4",  
"@emotion/react": "^11.7.1",  
"@emotion/styled": "^11.6.0",  
"@testing-library/jest-dom": "^5.16.1",  
"@testing-library/react": "^12.1.2",  
"@testing-library/user-event": "^13.5.0",  
"axios": "^0.25.0",  
"react": "^17.0.2",  
"react-chips": "^0.8.0",  
"react-dom": "^17.0.2",  
"react-lottie": "^1.2.3",  
"react-notification-badge": "^1.5.1",  
"react-router-dom": "^5.3.0",  
"react-scripts": "5.0.0",  
"react-scrollable-feed": "^1.3.1",  
"socket.io-client": "^4.4.1",  
"web-vitals": "^2.1.3"
```

Figura 15 – frontend pacchetti

Maggiore attenzione va ai seguenti:

- **Axios:** permette di effettuare richieste http e di intercettare sia la richiesta che la risposta dando l'opportunità di modificarle
- **Chakra-ui/react:** permette di usufruire dell'interfaccia utente di Chakra che contiene un set di componenti React
- **Chakra-ui/icons:** permette di utilizzare componenti icona React di base per l'interfaccia utente di Chakra.
- **React-chips:** Un input React controllato per array di dati. I chip sono elementi compatti che rappresentano un input, un attributo o un'azione.
- **React-Notification-badge:** Componente React di reazione del badge di notifica
- **React-scrollable-feed:** Un componente React che permette di scorrere manualmente una chat box quando arrivano nuovi messaggi
- **Socket-io.client:** permette la comunicazione bidirezionale in tempo reale lato client verso il server

Una volta elencati tutti i pacchetti installati è possibile fare un'analisi dei file e delle cartelle che costituiscono la struttura dell'applicazione lato front-end:

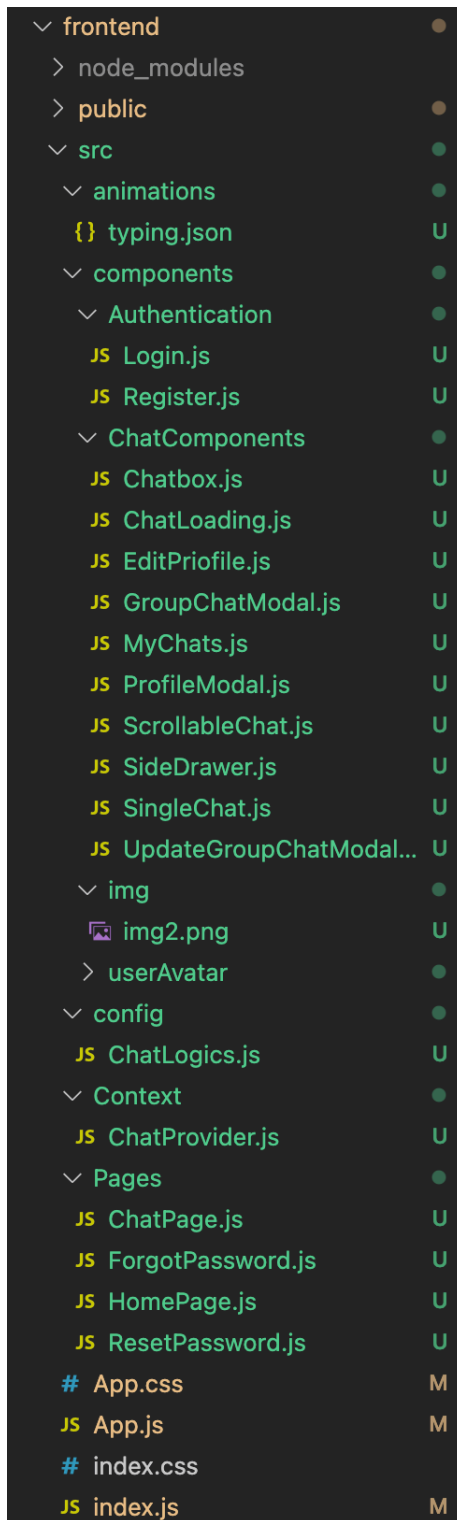


Figura 16 – struttura frontend

BACK-END

Anche nel caso del back-end è opportuno elencare tutti i moduli installati ed utilizzati

```
"@sendgrid/mail": "^7.6.0",  
"bcryptjs": "^2.4.3",  
"dotenv": "^14.2.0",  
"express": "^4.17.2",  
"express-async-handler": "^1.2.0",  
"jsonwebtoken": "^8.5.1",  
"mongoose": "^6.1.7",  
"nodemon": "^2.0.15",  
"socket.io": "^4.4.1"
```

Figura 17 – backend pacchetti

Maggiore attenzione va ai seguenti:

- **Sendgrid/email:** Questa libreria viene utilizzata per i servizi Twilio SendGrid per effettuare richieste all'API Web Twilio SendGrid v3 per la consegna della posta elettronica.
- **Bcryptjs:** Una libreria che permette di eseguire l'hashing delle password
- **Jsonwebtoken:** Una libreria che utilizza i jsonwebtoken che consente di conservare le informazioni sulla sessione dopo l'accesso.
- **Dotenv:** è un modulo che carica le variabili di ambiente da un.env
- **express-async-handler:** Middleware per la gestione delle eccezioni all'interno di percorsi asincroni e il loro passaggio ai gestori di errori.
- **Mongoose:** Mongoose è uno strumento di modellazione a oggetti MongoDB progettato per funzionare in un ambiente asincrono
- **Nodemon:** è uno strumento che aiuta a sviluppare applicazioni basate su node.js riavviando automaticamente l'applicazione del nodo quando vengono rilevate modifiche ai file nella directory, è un wrapper sostitutivo per node
- **Socket.io:** permette la comunicazione bidirezionale in tempo reale lato server verso il client

Una volta elencati tutti i pacchetti installati, è possibile fare un'analisi dei file e delle cartelle che costituiscono la struttura dell'applicazione lato back-end:

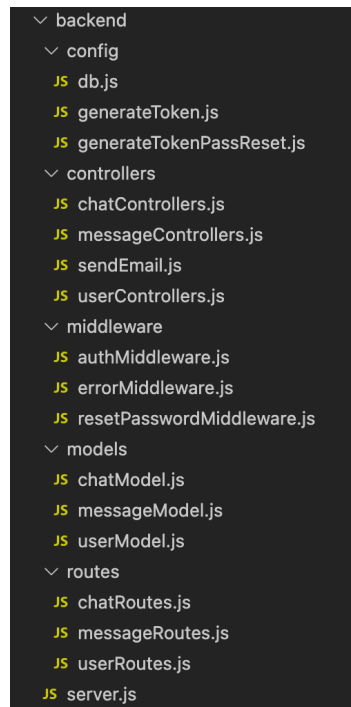


Figura 18 – struttura backend

PostMan

Al fine di testare il back-end in maniera indipendente dal front-end è stato utilizzato PostMan che permette di effettuare richieste http verso un qualsiasi indirizzo

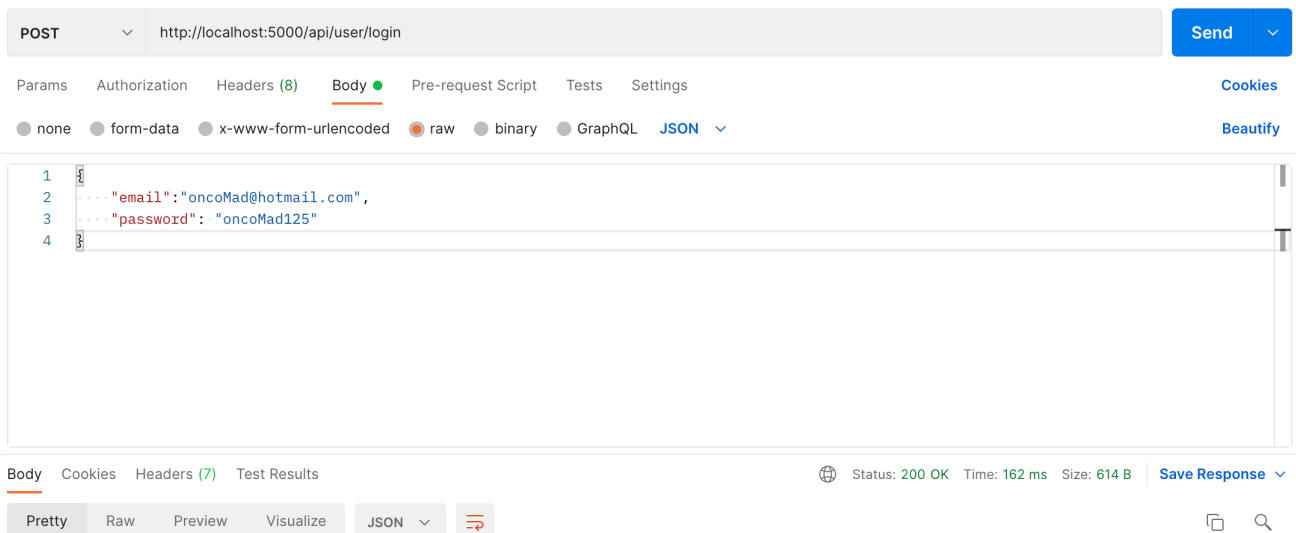


Figura 19 – PostMan

Com'è possibile notare è stata fatta una richiesta di tipo POST all'indirizzo utilizzato per il login con all'interno le credenziali, il codice di risposta è 200 OK.

NB: Si demanda al codice per una vista migliore delle classi

DATABASE

Come già anticipato in precedenza, è stato utilizzato un database NoSQL che permette la memorizzazione dei dati in un formato JSON-like: **MongoDB**. Esso contiene le informazioni relative agli utenti, alle chat e ai messaggi inviati dagli utenti in una determinata chat.

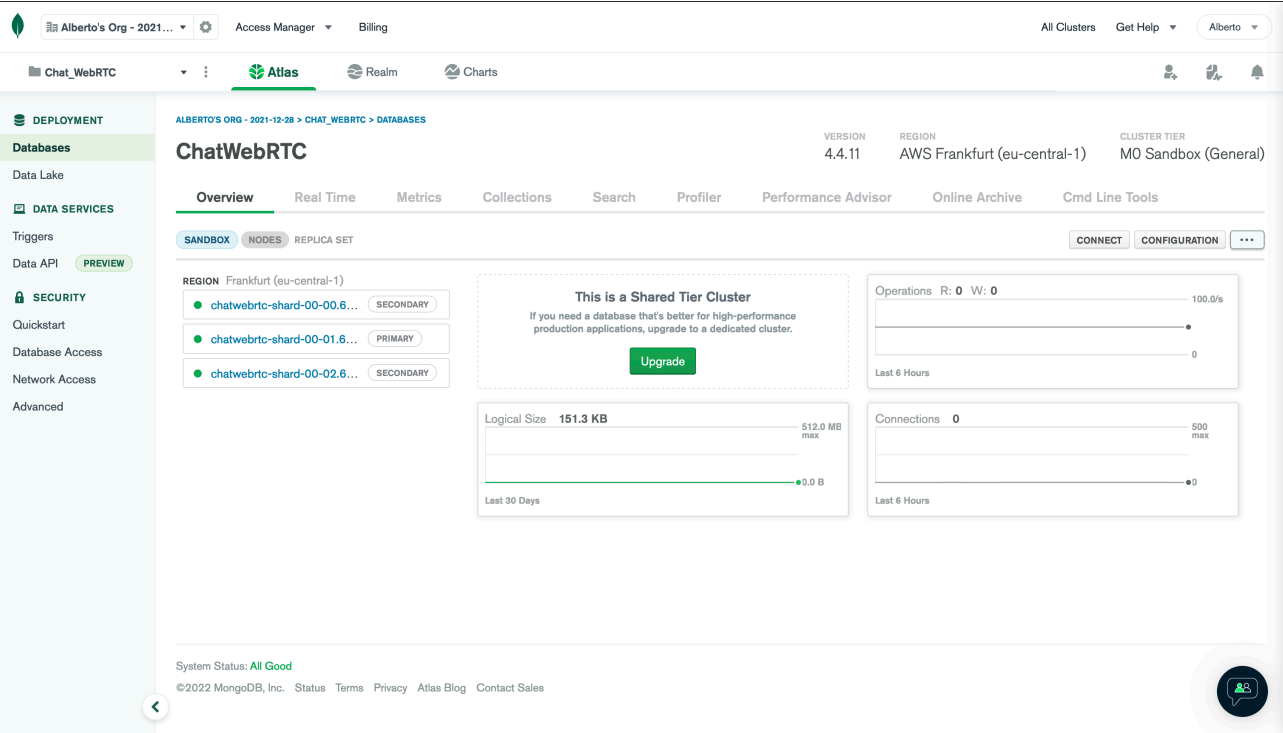


Figura 20 – MongoDB

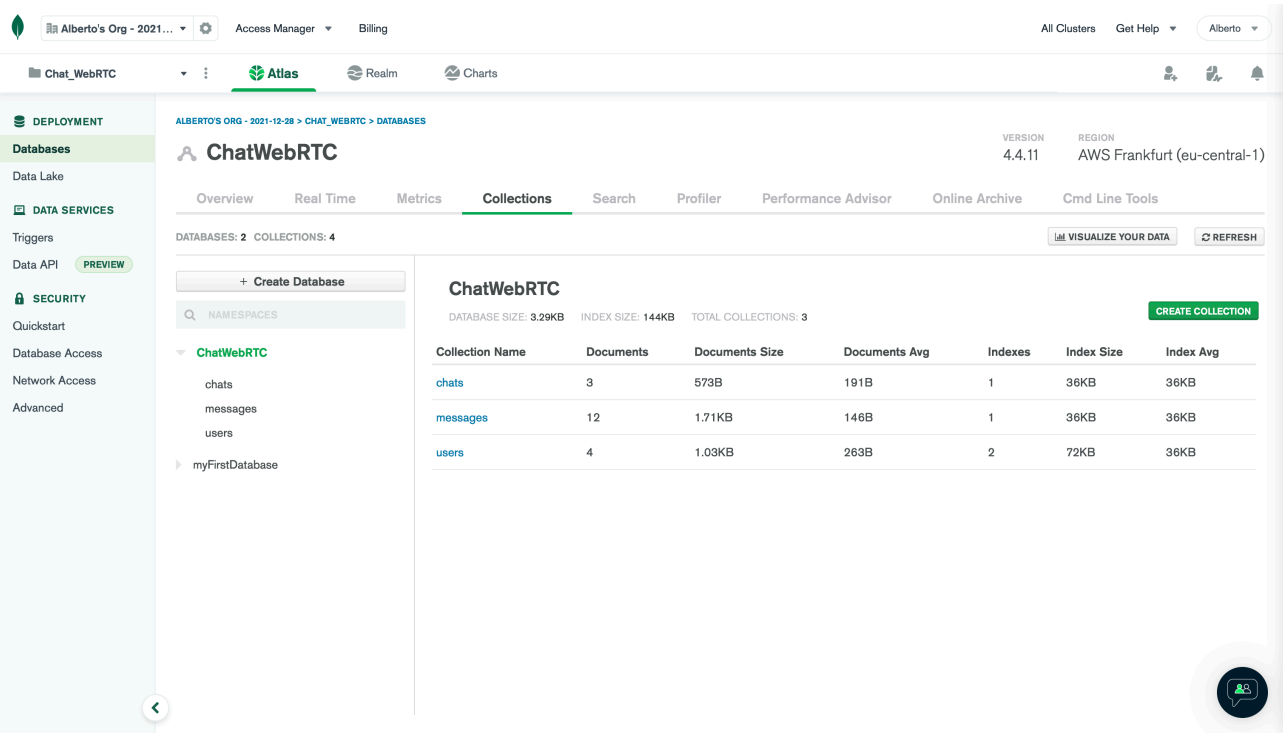


Figura 21 – Collections MongoDB

SVILUPPI FUTURI

Di seguito è riportato un elenco di alcune feature da implementare nel prossimo futuro:

- Possibilità di inoltrare file multimediali (immagini e video)
- Possibilità di effettuare chiamate e video-chiamate
- Realizzazione di un eventuale bot di supporto
- Accesso con Google, Facebook, Github