SimpleSocialMedia

Squarcialupi Riccardo riccard.squarcialupi@studio.unibo.it
Matricola 0001041294 - Anno Accademico 2021-2022
Applicazione e Servizi Web

4 settembre 2022

Indice

Elenco delle figure

1 Introduzione

SimpleSocialMedia è una applicazione che permette agli utenti, una volta iscritti, la condivisione di post e scambio di messaggi in tempo reale . L'app mette a disposizione un'interfaccia con 3 schermate principali molto semplici da raggiungere:

- Schermata Home
- Schermata Profilo Personale
- Schermata Chat

2 Requisiti

2.1 Panoramica

Il software sviluppato consente ai singoli utenti di:

- Registrarsi e di Accedere al'app
- Personalizzazione info sulla pagina personale
- Condividere post con immagini e testo
- Avere delle chat private con altri utenti
- Visualizzare i post delle persone che si seguono e mettere "Like"

2.2 Caratteristiche Utenti

Gli utenti sono una qualsiasi entità che voglia utilizzare un social media con motivazione anche diverse. Un esempio potrebbe essere quello di un'azienda che vuole creare una profilo dove promuovere i propri prodotti così come allo stesso momento una persona potrebbe semplice condividere foto senza pensare ad fine di lucro. In virtù di queste considerazioni si è deciso che la scelta migliore fosse quella di astrarre a semplice entità.

2.3 Obiettivi degli utenti

L'utente vuole che l'applicazione raccolga i propri dati in input e li memorizzi in modo consistente e sicuro. Vuole inoltre accedere ai dati pubblici altrui e creare delle chat persistenti con essi.

2.4 Requisiti Funzionali

2.4.1 Registrazione

La registrazione deve garantire all'utente di potersi registrare al sistema in modo semplice ed intuitivo garantendo, al tempo stesso, che i dati inseriti dallo stesso siano validi e consistenti. In particolar modo la registrazione deve garantire che:

- nessun campo vuoto
- il campo password e conferma password devono essere uguali
- non è possibile usare lo stesso username di altre persone

2.4.2 Login

Il login deve consentire all'utente di accedere al sistema effettuando un match tra username e password. Nella funzione di login sono presenti due possibili scenari:

- L'utente inserisce il proprio username e la relativa password accedendo al sistema
- L'utente commette un errore nell'inserimento di username o password ed è impossibilitato nell'accedere al sistema.

2.4.3 Logout

Il sistema deve consentire all'utente di effettuare l'operazione di logout, con conseguente richiesta di credenziali nel caso in cui l'utente desiderasse nuovamente effettuare l'accesso.

2.4.4 Inserimento Dati

Con inserimento dati si intende:

- Primo inserimento e successiva modifica info personali di un utente
- Inserimento testo in una chat con un utente
- Inserimento testo e eventualmente immagini all'interno di un post da condividere

2.4.5 Visualizzazione Dati

Per ogni schermata (Home, Profilo Personale, Chat) il sistema deve consentire la visualizzazione di quanto segue:

- Home: panoramica post condivisi dall'utente stesso e da utenti che eventualmente segue. Per ogni post si ha la visualizzazione del numero di "like" e se l'utente corrente ha messo "like" al post.
- Profilo Personale: il sistema consente la visualizzazione dei dati personali quali:
  - Nome, cognome, username
  - Città e Nazione
  - Sede di lavoro
  - Relazione sentimentale
  - Foto profilo e foto cover profilo
- Chat: panoramica chat utenti disponibili e una volta selezionata una chat, storico della conversazione se presente.

2.5 Requisiti non funzionali

- Il DB deve garantire una consistenza dei dati inseriti
- Le password all'interno del DB vengo hashate con un "salt"
- Le conversazioni tra utenti devono essere reattive e in tempo reale.

3 Design

3.1 Design Architetturale

L'applicazione sviluppata è una Multi Page Application con interfacciamento sul Server per effettuare le chiamate al DB in modo asincrono. Attraverso Redux inoltre viene mantenuto lo stato completo di tutta l'applicazione all'interno di un altro oggetto Javascript (oggetto State globale). Sempre attraverso Redux si gestiscono le chiamate allo storage interno, mentre con Axios vengono gestite le chiamate con il Server che consentono l'aggiunta, l'aggiornamento ed il recupero dei dati. Inoltre i dati vengono inviati al server con una firma digitale ottenuta con JSON Web Token (JWT) che permette di ottenere Data Integrity. L'immagine sotto rappresenta una generalizzazione dell'architettura utilizzata.

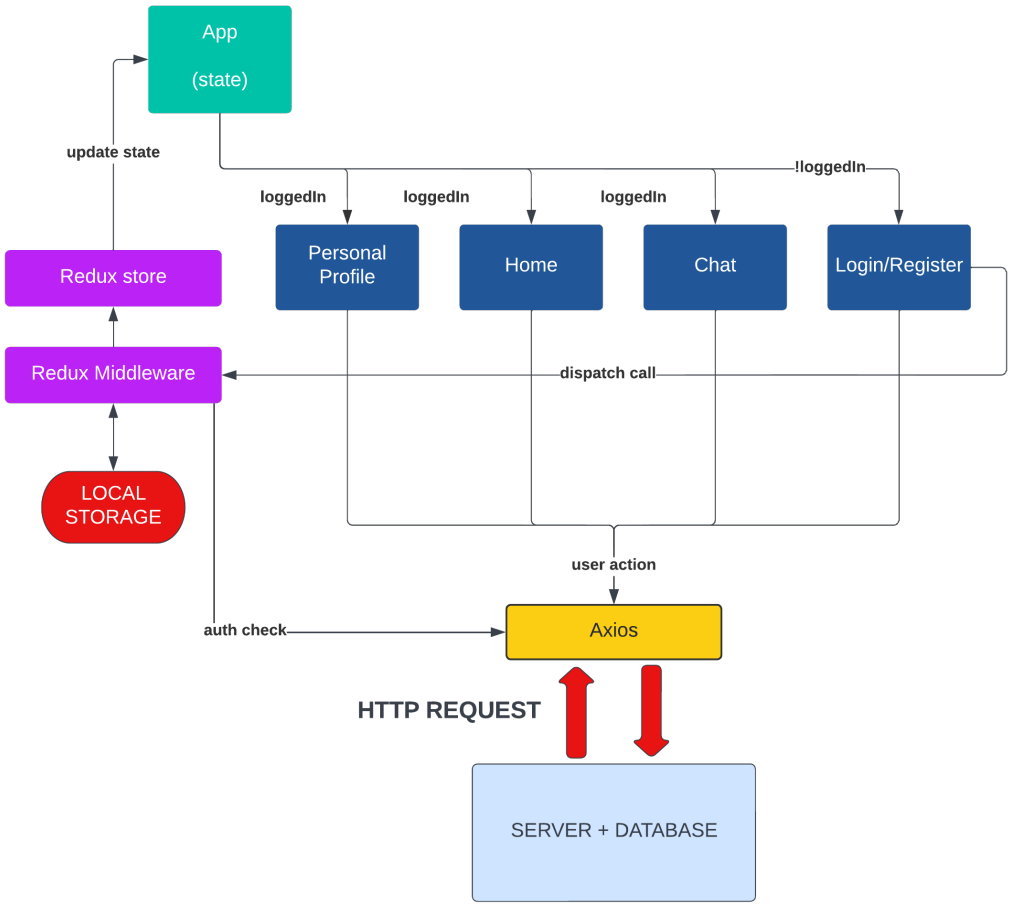

Figura 1: Schema Architetturale usato all'interno dell'applicazione

3.2 Metodologia di sviluppo

La metodologia di sviluppo adottata è Agile. Si è partiti infatti da una definizione di cosa l'applicazione dovesse fare in termini di requisiti funzionali/non funzionali da soddisfare, interfacce grafiche, usabilità da parte degli utenti per poi cambiare e aggiornare in corso d'opera le varie componenti andando a convergere sempre di più verso il risultato finale. Una parte importante di questa metodologia di sviluppo è stata quella di ragionare come utilizzatore e come progettista fornendo continuamente feedback per poter apportare mi-

glioramenti. Durante l'intero processo di sviluppo si è prestata una notevole attenzione scrivere del codice che facilitasse eventuali modifiche (preventivate e prevedibili visto l'approccio Agile adottato).

3.3 Coinvolgimento Utenti

Durante l'intero arco di sviluppo del progetto quello che è stato fatto è adottare un approccio User Centered Design. In particolare si è deciso di utilizzare due utenti virtualizzati diversi tra di loro per poter soddisfare le richieste lungo uno spettro che fosse il più ampio possibile. Le virtualizzazioni adottate sono le seguenti:

- Utente non esperto nell'utilizzo di servizi online
- Utente esperto nell'utilizzo di servizi online

Queste due tipologie di utenti hanno fortemente influenzato il design delle interfacce grafiche.

3.4 Casi d'uso

Per lo sviluppo dell'applicazione e conseguentemente delle interfacce grafiche sono stati individuati diversi macro casi d'uso in grado di fornire un'idea estremamente precisa di cosa il sistema dovesse fare. Di seguito sono riportati i casi d'uso analizzati.

- Registrazione
- Login
- Inserimento/Modifica dati nella Info Utente
- Creazione Post/Rimozione Post
- Follow e UnFollow Utenti
- Chat Utente

3.5 Design Interfacce Grafiche

Per le interfacce grafiche sono state indivduate due Personas corrispondenti alle seguenti categorie:

- Utente non esperto nell'utilizzo di servizi online
- Utente esperto nell'utilizzo di servizi online

Le personas individuate, conseguentemente a ciò, sono le seguenti:

- Mario è un pensionato che vive in un paesino di una provincia italiana. Da qualche tempo sta imparando ad utilizzare il computer grazie all'ausilio dei nipoti. Vorrebbe iniziare ad utilizzare i social per conoscere persone della sua età.
- Roberto è a capo di uno studio marketing di una grande città. Un'azienda ha puntato su di lui per creare e mantenere un profilo social per sponsorizzare i propri prodotti.

Da queste personas si evince che la facilità di customizzazione e la praticità dell'app devono essere il fulcro dello sviluppo in ambo i casi. La soluzione quindi è stata di suddividere le funzionalità in 3 pagine separate (Home, Pagina Personale e Chat) cercando di ottenere un interfaccia più semplice possibile lasciando spazio alle immagini che predominano lo spazio centrale. Un ulteriore sforzo è stato fatto per quanto riguarda la parte di registrazione in quanto il form da compilare richiede due volte la password e notifica l'utente se esse non corrispondono oppure viene utilizzato un username già esistente.

3.5.1 Componenti del progetto

Figura 2: Schermata Login

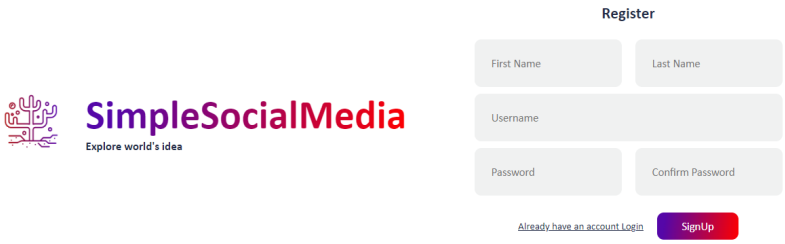

Figura 3: Schermata Registrazione

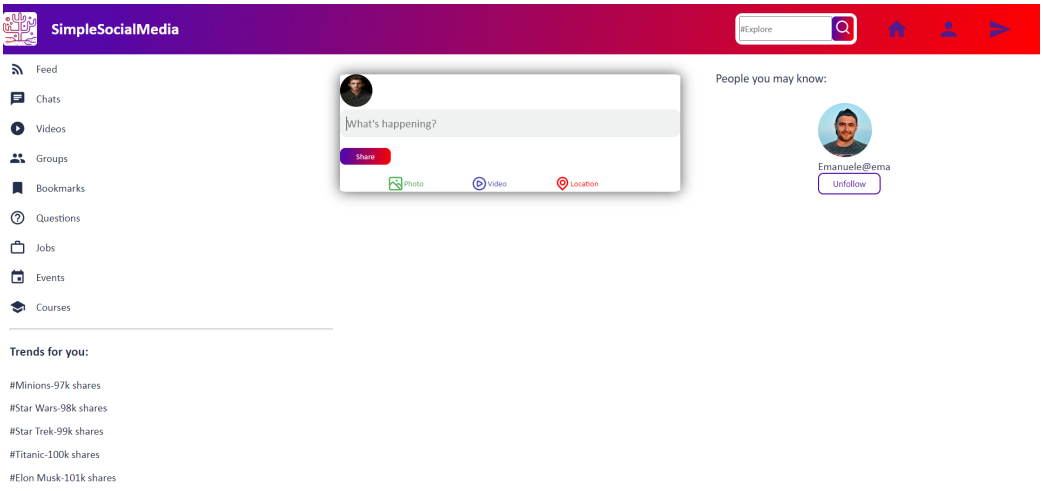

Figura 4: Schermata Home

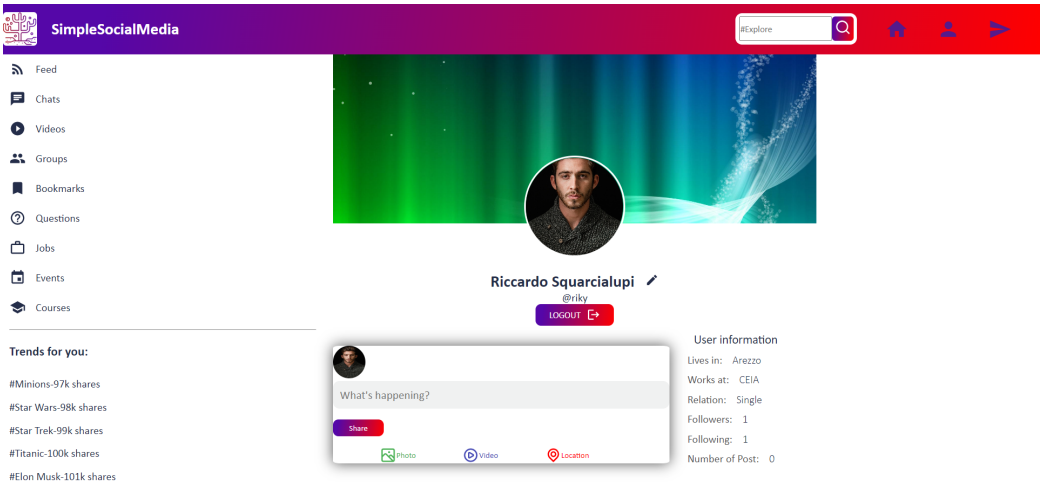

Figura 5: Schermata Profilo Utente

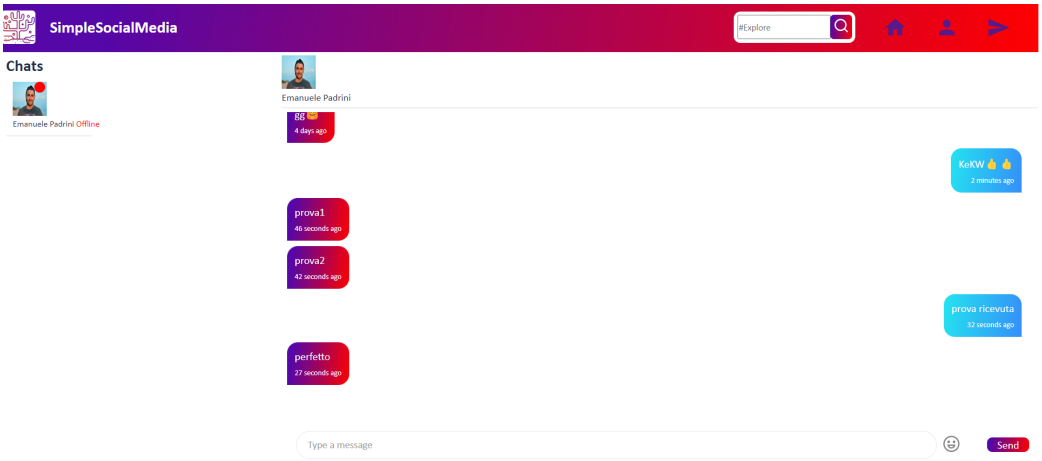

Figura 6: Schermata Chat

4 Tecnologie

Si è utilizzato come solution stack MERN. Le tecnologie utilizzate per sviluppare il sistema sono:

- Node ed Express: sviluppo server
- React: sviluppo client
- MongoDB: gestione persistenza dati
- Socket.io: scambio dati in real time (Chat)
- Javascript (ES6): lato client e server
- Material-UI, Mantine e CSS: per parte grafica client
- Axios: per connettersi alle API del backend ed effettuare richieste HTTP.
- Redux: per gestire lo "stato" dell'applicazione
- JWT per avere Data Integrity nelle comunicazioni tra client e server
- Mongoose: per creare le strutture dati che saranno salvate su MongoDB

5 Codice

Di seguito vengono riportati frammenti di codice di esempio del progetto.

```
// handle Image Change
const onImageChange = (event) => {
  if (event.target.files && event.target.files[0]) {
    let img = event.target.files[0];
    setImage(img);
  }
};

const imageRef = useRef();

const handleUpload = async (e) => {
  e.preventDefault();

  //post data
  const newPost = {
    userId: user._id,
    username: user.username,
    imageUser: user.profilePicture,
    desc: desc.current.value,
  };

  // if there is an image with post
  if (image) {
    const data = new FormData();
    const fileName = Date.now() + image.name;
    data.append("name", fileName);
    data.append("file", image);
    newPost.image = fileName;
    console.log(newPost);
    try {
      dispatch(uploadImage(data));
    } catch (err) {
      console.log(err);
    }
  }
  dispatch(uploadPost(newPost));
  resetShare();
};

// Reset Post Share
const resetShare = () => {
  setImage(null);
  desc.current.value = "";
};
```

Figura 7: Funzioni di supporto all'inserimento di un'immagine in un Post.

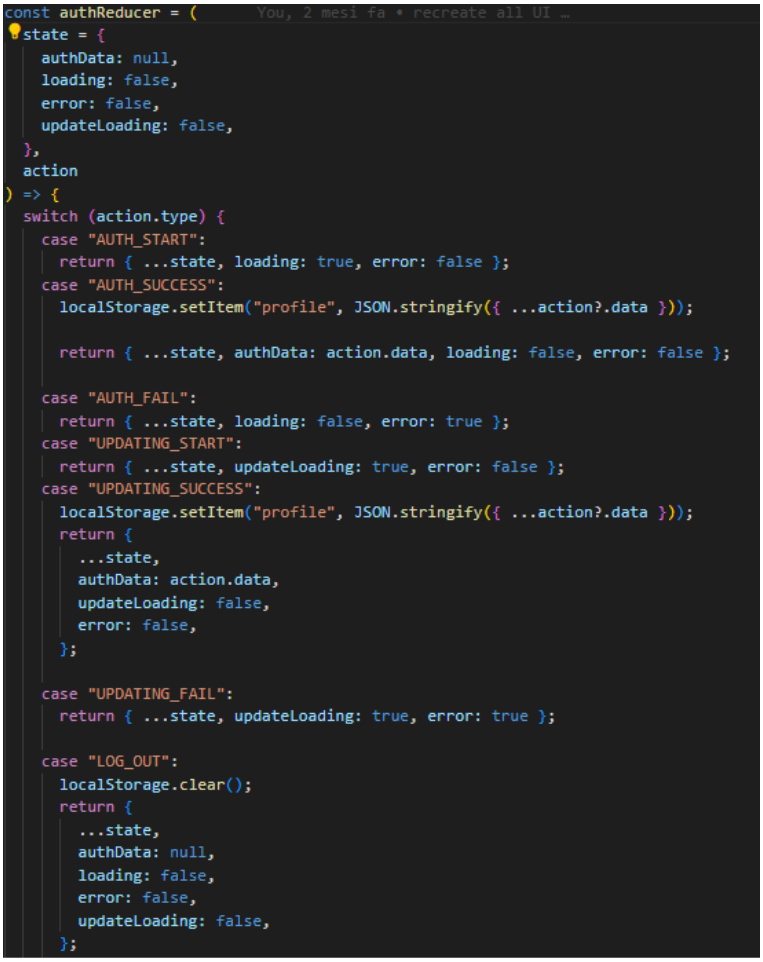

Figura 8: Reducer per l'Autenticazione - 1.

Figura 9: Reducer per l'Autenticazione - 2.

6 Test

Il sistema è stato testato su Google Chrome, Safari, Edge, Chromium, Mozilla e Safari in modo da garantire la portabilità dello stesso. Ulteriori test sono

stati fatti per testare l'interfaccia grafica e la Data integrity di JWT. La funzionalita delle chiamate Axios sia per quanto riguarda le operazioni GET che per quanto riguarda le operazioni PUT/POST sono state testate con Thunder Client, un plugin simil-PostMan integrabile all'interno di Visual Studio Code.

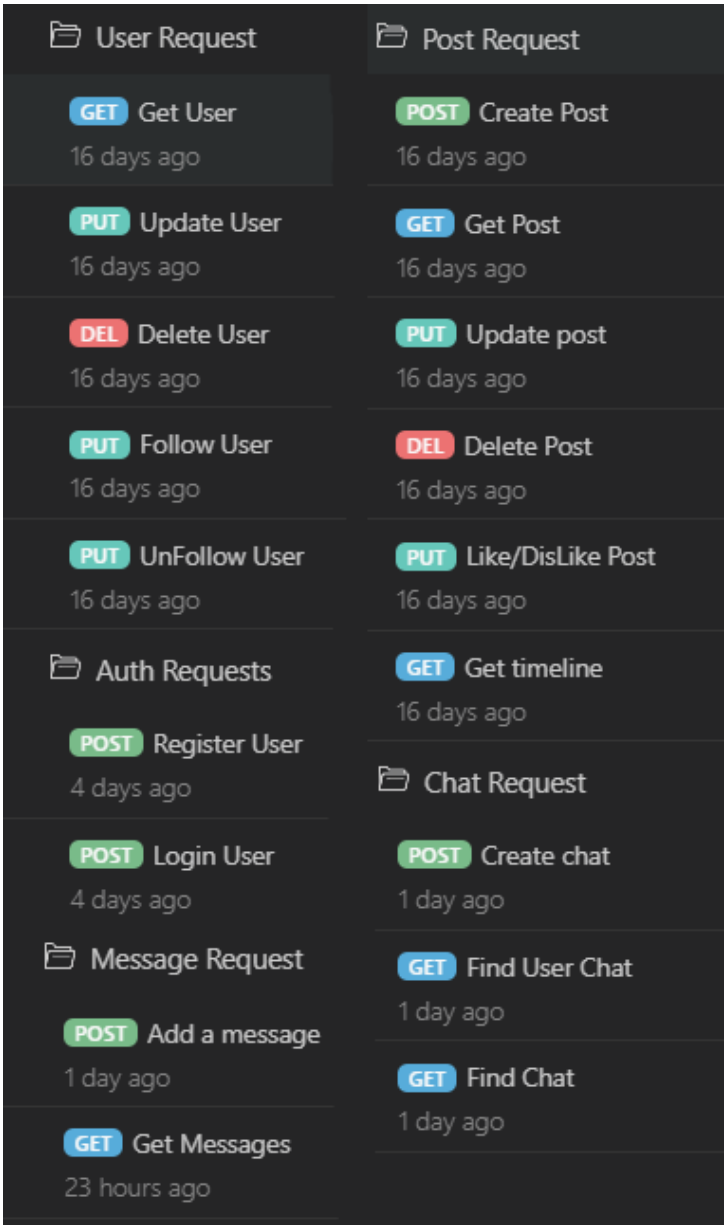

Figura 10: Lista di API testate con Thunder Client

7 Deployment

Una volta clonato il repository ed avere installato Node.js sarà sufficiente eseguire a seconda del sistema operativo usato uno dei seguenti script:

- "UnixInstallRun.sh" per i sistemi Linux e MacOS
- "WinInstallRun.bat" per i sistemi Windows

Si noti che una connessione internet è sempre necessaria al fine di collegarsi al database ed installare i node_modules.

8 Conclusioni

8.1 Risultati Ottenuti

L'applicazione svolge i compiti per cui è stata ideata, consente agli utenti di creare post mettere "like" ad essi, modificare le info personali, seguire e smettere di seguire gli altri utenti, chattare con altri utenti.

8.2 Sviluppi futuri

Un possibile sviluppo futuro sarebbe quello di integrare funzionalità già presenti in social come Instagram o Facebook, come per esempio la creazione di gruppi privati, creazione di "storie" che dopo 24h vengono eliminate, possibilità di commentare i post e taggare altri utenti, possibilità di caricare video.