

RankMe
Applicazioni e Servizi Web

Luca Rossi - 0000991926 {luca.rossi57@studio.unibo.it}
Davide Schiaroli - 0000981607 {davide.schiaroli2@studio.unibo.it}

01 Gennaio 2020

Introduzione

Il progetto consiste nella realizzazione di un nuovo social che punta a stilare la classifica degli utenti più popolari.

Ogni utente una volta registrato e loggato si troverà davanti due profili, ogni profilo è rappresentato da una foto e l'unica cosa che l'utente deve fare è sceglierne uno. Ad ogni voto corrisponde un punto, più un utente ha voti più scala la classifica e l'utente con più voti sarà il primo in assoluto.

Sarà possibile modificare varie componenti del profilo in modo tale da favorire una migliore user-experience.

La classifica potrà anche essere filtrata per generare delle sottoclassifiche più settoriali, ad esempio prendendo la classifica dei più popolari in una città o in una nazione.

Per permettere agli utenti di targettizzare il loro pubblico sono state create delle analytics che permettono loro di ottenere dati statistici dai voti ricevuti.

Chapter 1

Requisiti

In questo capitolo verranno discussi i requisiti funzionali e i requisiti non funzionali.

1.1 Requisiti funzionali

Per permettere un utilizzo corretto e naturale dell'applicazione si è deciso di utilizzare una struttura con poche pagine e funzioni molto precise e concise, che permettessero di avere una grafica omogenea per non distrarre o disorientare l'utente finale.

1.1.1 Gestione delle pagine

L'applicazione finale è il componimento delle seguenti pagine (i nomi saranno scritti in inglese, lingua usata all'interno dell'applicazione):

- Login
- Signup
- Home
- Profile
- Analytics
- Rank

Login

La pagina di Login punta ad essere il più pulita e semplice possibile in modo da non confondere l'utente ed essere il più diretta possibile. È il punto di ingresso dell'applicazione. (nel design parlare del colore e della struttura).

Signup

Le informazioni necessarie per avere un profilo completo sono molte, per questo motivo si è creata una pagina di Signup molto minimale, che consentisse una creazione veloce di un profilo e si è fatto in modo di poter aggiungere le info mancanti in più step che l'utente può iniziare in qualsiasi momento, in modo da invogliarlo a completare il suo profilo anziché demotivarlo.

Home

La Home come anche le altre pagine cerca di essere il più minimale possibile. È la componente principale dell'applicazione e da lì deve essere possibile accedere al Profilo, alle Analytics, al Ranking e fare logout. La Home deve permettere di votare un profilo dalla coppia proposta, facendone visualizzare un'altra quando il voto è andato a buon fine. Sarà inoltre possibile visionare i profili proposti visualizzando altre informazioni, come la posizione in classifica, social (facebook e instagram), biografia o data di nascita dell'utente in questione.

Profile

Dal profilo sarà possibile visualizzare le informazioni sopra descritte, più nello specifico il profilo di un utente è composto da:

- nome
- cognome
- username: nome univoco all'interno dell'applicativo
- email: unica all'interno dell'applicativo
- posizione in classifica
- biografia: spazio a disposizione dell'utente per personalizzare il proprio profilo, descrivere hobby, interessi o qualsiasi altra cosa
- città: città di residenza dell'utente
- stato: stato di residenza dell'utente (in Italia corrisponde alla regione)
- paese: paese di residenza dell'utente
- sesso
- facebook: link al profilo facebook, per farsi pubblicità
- instagram: link al profilo instagram, per farsi pubblicità
- badge: dà un di sfida per l'utente (gamification)

Questa pagina è estremamente importante e fa da piattaforma da cui è possibile farsi vedere e conoscere meglio.

Analytics

Pagina dedicata ai dati statistici dell'applicazione. I social sono ormai il mezzo principale e più redditizio per pubblicizzare prodotti, e sono nate nuove professioni che si basano sull'utilizzo di social proprio per raggiungere visibilità elevate, questa pagina serve a dare supporto a queste figure fornendo dati interessanti e utilizzabili per crescere.

Rank

Pagina che fornisce la classifica delle persone più votate. È il punto di raccordo da cui poter stilare anche classifiche più ristrette. Di default vengono mostrate le prime dieci persone al mondo, ma è possibile ottenere le n (n variabile a discrezione dell'utente) persone più votate in determinate:

- città
- stati
- paesi
- età (la classifica delle persone più votate che hanno 23 anni ad esempio)

È possibile creare un'infinità di filtri e modi diversi per visualizzare le classifiche e questa pagina potrebbe in futuro crescere per attirare più gente e creare un senso di competitività.

1.1.2 Notifiche

Le notifiche sono fondamentali per un corretto e sano coinvolgimento dell'utente e per mostrare la vera potenza dello stack MEAN [20], o come nel nostro caso, delle sue varianti (Mongodb, Express, Angular-React-Vue, Node).

A questo proposito sono state utilizzate due diverse tecnologie:

- email
- toast

Email

Le email sono senza dubbio lo strumento più diffuso per condividere informazioni. Vengono utilizzate quando l'utente deve ricevere e poter mantenere determinate informazioni, non sono sempre la scelta ottimale in ogni situazione, infatti, non sono immediate e per essere lette bisogna uscire dall'applicazione, o comunque metterla da parte. Sono tuttavia ottime per far sapere all'utente che determinate azioni sono avvenute con successo, che sono state prese delle decisioni, o più in generale per fornire comunicazioni di servizio.

Toast

I toast d’altro canto sono molto più coinvolgenti, l’utente è invitato a leggerli e una volta presa familiarità col sistema imparerà a riconoscerli quasi inconsciamente, riuscendo ad avere feedback immediati da determinate azioni. I toast sono utilizzabili anche per notificare l’utente che qualcosa sta accadendo al proprio profilo, come ad esempio l’essere stato votato, invitandolo, almeno auspicabilmente, a capire autonomamente che cosa sta accadendo e permettendo una visione più positiva sull’applicazione in generale. Utilizzo tipo: l’utente viene notificato di aver ricevuto n nuovi voti = \downarrow l’utente controlla il proprio profilo e la posizione in classifica = \downarrow controlla le analytics = \downarrow torna alla home e continua a votare = \downarrow il ciclo ricomincia. In generale questo tipo di notifiche dovrebbe piacere all’utente dando un senso di ricompensa che lo invogli a continuare ad usare l’applicazione.

1.2 Requisiti non funzionali

In questa sezione andremo invece ad analizzare alcuni tra i requisiti non funzionali su cui abbiamo deciso di soffermarci di più a livello di implementazione.

1.2.1 Usabilità

Questo requisito è stato il punto cardine di tutta l’implementazione ed è stato perseguito utilizzando metodologie HCI che verranno meglio ripresi nel capitolo 2 dove verranno illustrate le interfacce utente e l’architettura del sistema. In generale per perseguire questo requisito sono state utilizzate immagini per richiamare concetti (casa- \downarrow home) ed è stato fatto uno studio sui colori per rendere il sito omogeneo, coerente e supportare/suggerire inconsciamente l’utente su cosa la pagina deve trasmettere.

1.2.2 Efficienza

Nella nostra implementazione è stata favorita l’efficienza prestazionale a scapito di quella di spazio. Il sito in sé non è molto pesante e la maggior parte delle operazioni non sono particolarmente onerose ma quando entra in gioco la classifica e il suo filtraggio, non solo l’efficienza diventa cruciale ma risulta anche necessaria per non compromettere l’esperienza utente, con pause di computazione troppo lunghe o che potrebbero utilizzare troppe risorse. In un’applicazione web, l’efficienza è molto correlata all’implementazione, che verrà discussa nella sezione 1.2.3 ma è ancora più correlata alla strutturazione del db. Per questo motivo si è deciso di ridondare dei dati a livello di db per permettere un recupero dei dati più veloce e delle query molto più sintetiche.

1.2.3 Implementazione

Per favorire il riuso e la leggibilità si è cercato di fare uso delle best practice nel mondo di javascript (che per la maggior parte sono le best practice del mondo della programmazione in generale) e di utilizzare il più possibile una programmazione funzionale che, come abbiamo visto nel corso tenuto dal professor Viroli, è meno error prone e favorisce il riuso di codice.

1.2.4 Security

Avendo seguito il corso del professor D'Angelo abbiamo ritenuto fondamentale avere un ottimo livello di sicurezza all'interno del sito.

JWT

Per implementare controllo degli accessi e autorizzazioni e la gestione delle sessioni in maniera completamente stateless sono stati usati i JWT, in particolare al corretto conseguimento del login vengono restituiti i dati di un utente utili per la visualizzazione del profilo e vengono consegnati:

- un "accessToken"
- un "refreshToken"

L'access token scade dopo 15 minuti dalla sua creazione ed è necessario per ogni richiesta che viene inviata al server, se non è presente il server risponderà con un 401 unauthorized. Il refreshToken viene invece utilizzato per chiedere nuovi accessToken, per effettuare il logout è sufficiente cancellare questo refreshToken dal server. Il refreshToken è infatti l'unico token che viene ricordato dal server e la sua cancellazione equivale alla disconnessione dell'utente e dall'uscita dalla sessione.

Password hashate + sale

Per mantenere sicure le password, sul server non vengono memorizzate in chiaro ma ne viene conservato il valore hashato. Per il sale e il controllo delle password ci si affida ad una libreria esterna molto usata in questo ambito, ovvero **bcrypt**.

https

Attualmente qualsiasi sito è protetto da una connessione https e ormai nessuno si sentirebbe sicuro con una connessione http. L'https non solo è molto conveniente a livello di sicurezza, permettendo di garantire confidenzialità alla comunicazione, ma fornisce anche un aiuto psicologico all'utente che vedendo il lucchetto di fianco all'url si sentirà più sicuro.

Per l'implementazione è stato usato openssl, con il quale è stata creata una chiave RSA ed è stato firmato il certificato.

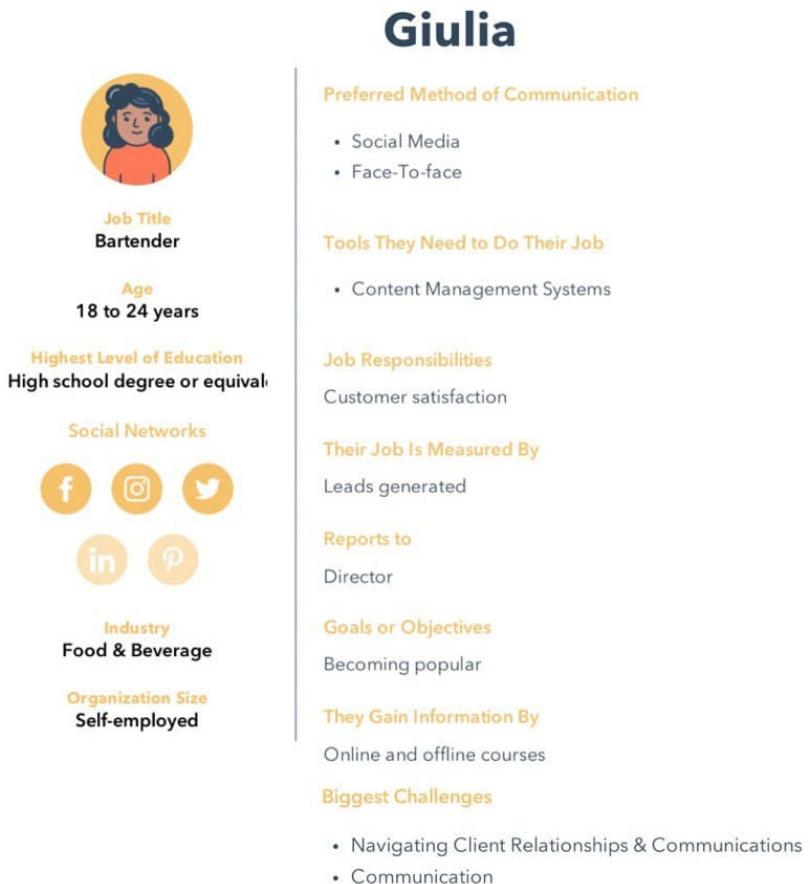


Figure 1.1: Giulia

1.3 Personas

Di seguito vengono proposte due Personas, Giulia 1.1: una ragazza, lavora come barista, vuole farsi conoscere per diventare famosa e aprire la sua attività e Olmo: un imprenditore digitale che lavora nel marketing e punta a pubblicizzare i prodotti dell'azienda in cui lavora.

Olmo



Figure 1.2: Olmo

Chapter 2

Design

Per non iniziare da zero ed avere una panoramica su come strutturare l'app e su quello che potrebbe servire per iniziare a sviluppare abbiamo deciso di partire creando dei mockup molto primitivi che ci aiutassero a buttare giù le prime linee di codice.

Nella fase di creazione dei mockup e di design abbiamo cercato di ispirarci il più possibile ad app già presenti sul mercato, famose per la loro semplicità d'uso. Abbiamo anche pensato che usare layout, icone e stili simili ad altri social già famosi potesse aiutare l'utente nel primo utilizzo dell'app facendolo sentire in qualche modo già "istruito" e abbassando al minimo quella barriera d'impatto che spesso viene a crearsi quando si inizia qualcosa di nuovo. Usare pattern già noti è anche alla base della user experience e aiuta a non disorientare l'utente e guidarlo verso un corretto utilizzo dell'app.

È stato favorito un approccio simil-agile, con un approccio iterativo che ci ha aiutati a creare l'app pezzo per pezzo. Iterativamente si andava a creare la parte di backend necessaria, si testava e se ne integrava il front-end. In questo modo abbiamo visto l'app crescere e prendere sempre più la sua forma definitiva.

Questo approccio non solo ci ha permesso di scrivere del codice più in fretta, ma ci ha anche concesso di testare l'app a piccoli blocchi, dando maggior consistenza al sistema nel complesso.

L'utilizzo dei mockup ci ha anche permesso di avere un parere in anteprima dai futuri utenti che hanno collaborato fin dalle prime fasi di sviluppo alla creazione di un ambiente confortevole e user-friendly.

2.1 Mockup

I mockup, come già anticipato, sono stati fondamentali per la creazione della nostra app. Di seguito ne verranno proposti alcuni.

2.1.1 Prima bozza

La prima schermata ad essere stata disegnata è ovviamente stata la home in figura 2.1 assieme alla sua versione desktop in figura 2.2.

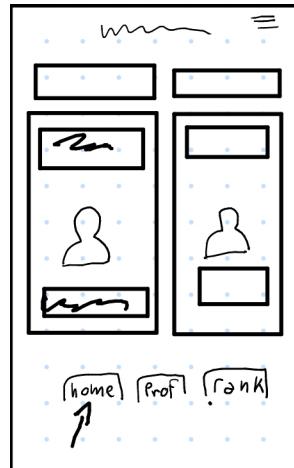


Figure 2.1: Home: prima bozza

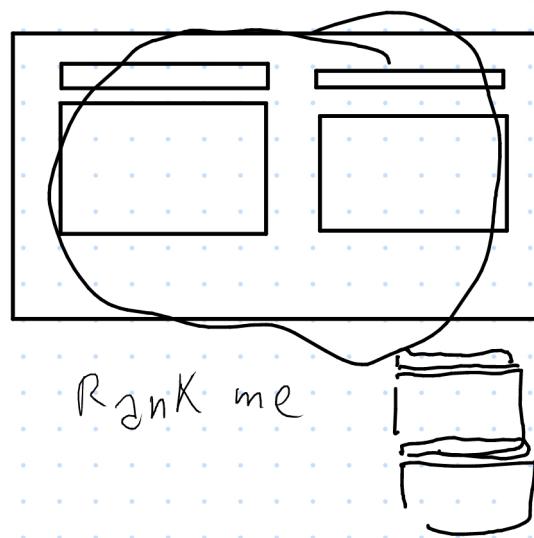


Figure 2.2: Home Desktop: prima bozza

Sono poi stati creati un profilo 2.3 e una pagina di ranking 2.4, che erano le altre due componenti principali della nostra app.

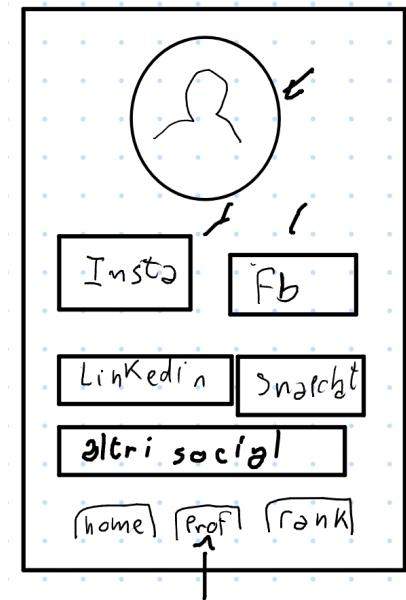


Figure 2.3: Profilo: prima bozza

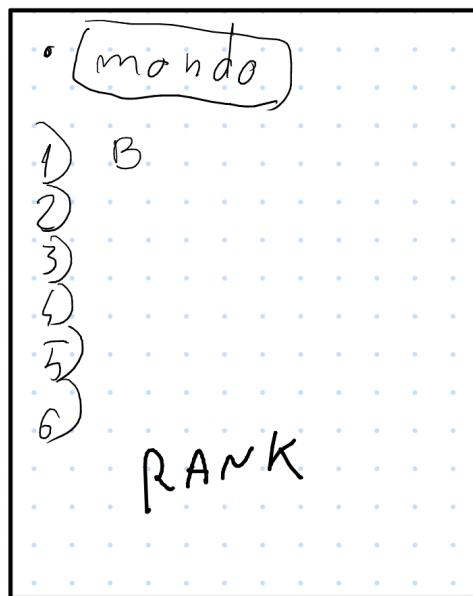


Figure 2.4: Ranking: prima bozza

2.1.2 Primi raffinamenti

Dopo i primi schizzi abbiamo provato a disegnare una versione dell'applicazione più fruibile e concreta.

Home

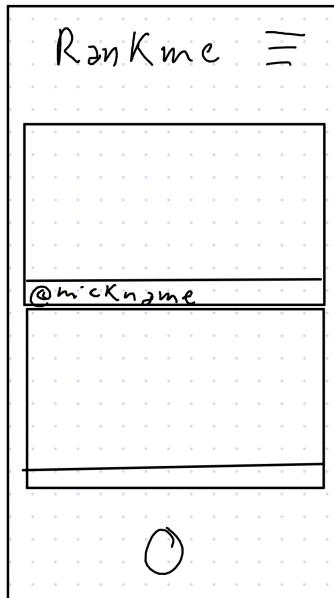


Figure 2.5: Home: raffinata

Profile

Nell'immagine 2.6 possiamo trovare il raffinamento proposto per il profilo, con più info personali e meno social.

Rank

In figura 2.7 troviamo la proposta di raffinamento della classifica mentre in figura 2.8 un singolo record della classifica.

2.2 Design finale del sito

Di seguito proponiamo il design finale del sito.

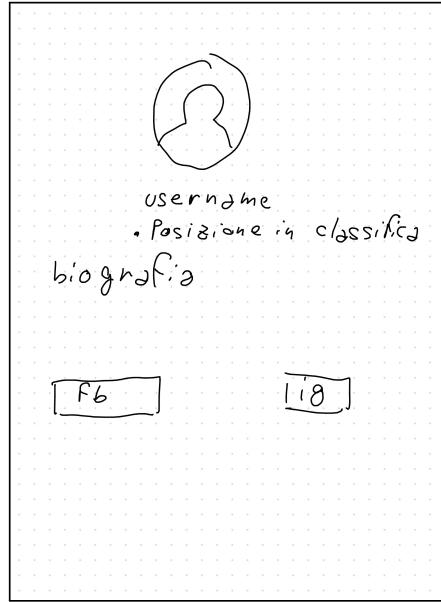


Figure 2.6: Profilo: raffinato

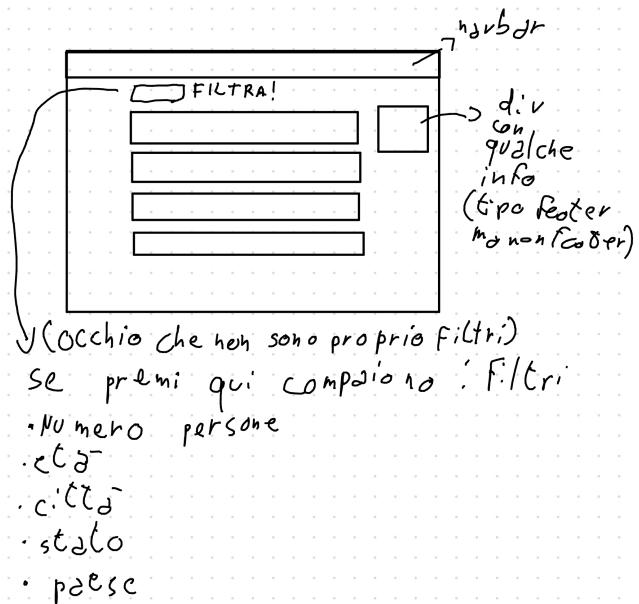


Figure 2.7: Rank: raffinata



Figure 2.8: Rank field

2.2.1 Home

In figura 2.9 troviamo la home mentre in 2.10 la sua versione mobile.

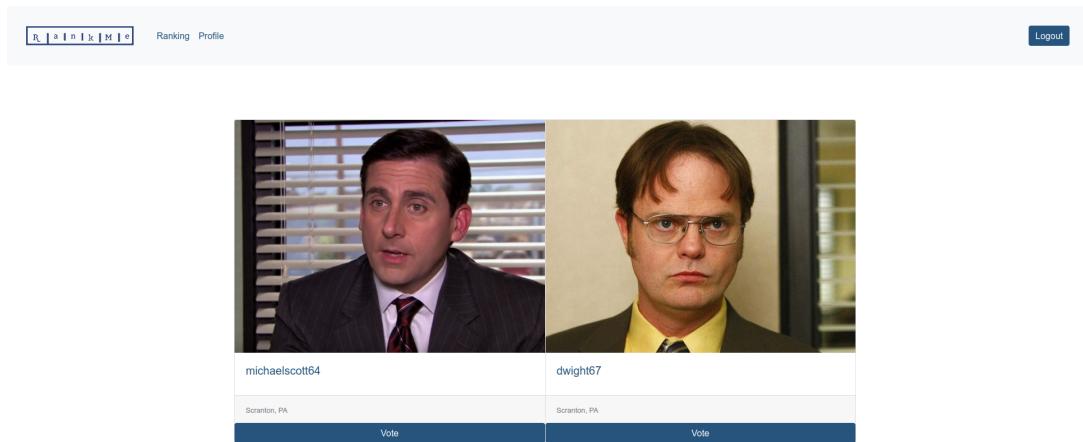


Figure 2.9: Home

2.2.2 Profile

In figura 2.11 troviamo il profilo mentre in 2.12 la sua versione mobile.

2.2.3 Ranking

In figura 2.13 troviamo la classifica, in figura 2.14 la sua versione mobile e in figura 2.15 possiamo vedere i vari filtri applicabili.



Figure 2.10: Home, versione mobile

The mobile profile screen for Michael Scott shows his circular profile picture at the top left. To the right of the picture, the text '@michaelscott64 - Michael Scott' is displayed, followed by 'Rank: 1'. Below this, a quote reads: 'Would I rather be feared or loved? Easy. Both. I want people to be afraid of how much they love me.' To the right of the quote are two small icons: a sun-like symbol and a bar chart. Below the profile picture are two social media sharing buttons: a blue 'f' for Facebook and a red 'o' for Instagram. At the bottom of the screen is a horizontal progress bar with three segments: 'Beginner', 'Average', and 'Awesome'. Each segment has a corresponding goal: 'Get one vote!', 'Get 10 votes!', and 'Get 100 votes!'. The 'Average' segment is currently highlighted with a blue bar.

Figure 2.11: Profile

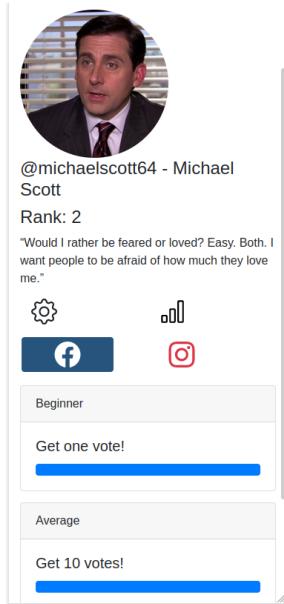


Figure 2.12: Profile mobile

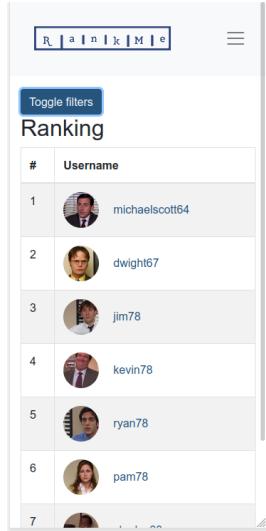
A screenshot of a ranking page. At the top left is a navigation bar with 'Ranking' and 'Profile' tabs, and a 'Logout' button on the right. Below is a table titled 'Ranking' with a 'Toggle filters' button above it. The table has columns for '#', 'Username', and a small profile icon. The data is as follows:

#	Username
1	michaelscott64
2	dwight67
3	jim78
4	kevin78
5	ryan78
6	pam78
7	stanley68
8	angela

Figure 2.13: Ranking

2.2.4 Analytics

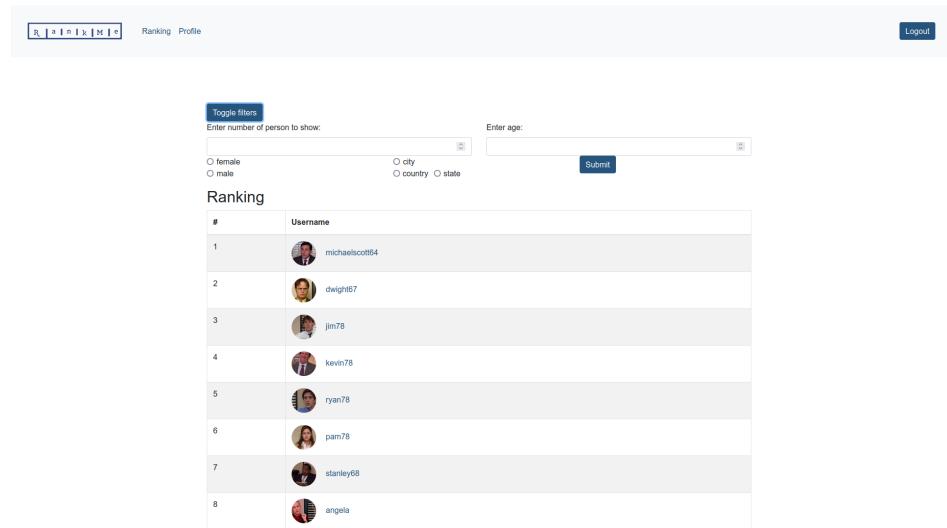
In figura 2.16 possiamo vedere la schermata di analytics.



Ranking

#	Username
1	michaelscott64
2	dwight67
3	jim78
4	kevin78
5	ryan78
6	pam78
7	stanley68

Figure 2.14: Ranking, versione mobile



Ranking

#	Username
1	michaelscott64
2	dwight67
3	jim78
4	kevin78
5	ryan78
6	pam78
7	stanley68
8	angela

Figure 2.15: Ranking, with forms

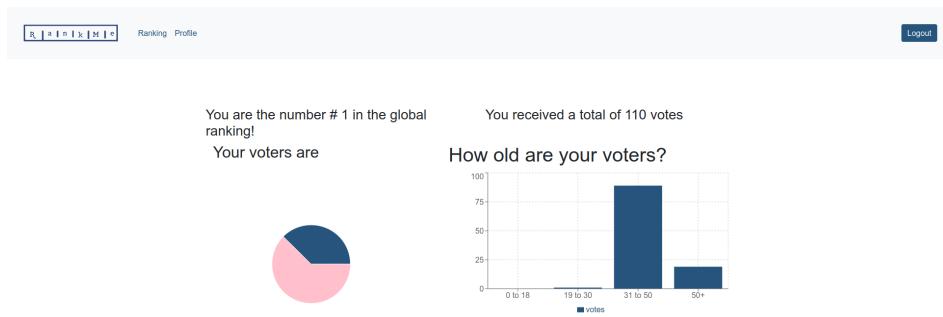


Figure 2.16: Analytics

Chapter 3

Tecnologie

Principalmente possiamo suddividere le tecnologie utilizzate in due categorie: quelle utilizzate dal server e quelle utilizzate dal client ed entrambe le categorie verranno espansse in questo capitolo.

3.1 Solution Stack

Tra i vari stack proposti abbiamo deciso di utilizzare MERN, ovvero MongoDB, Express, React e Node.

3.1.1 MongoDB [7]

MongoDB è un database documentale che supporta nativamente JSON. È caratterizzato da un query language molto espressivo e facile da imparare ed usare con molte funzionalità built-in. Viene attualmente usato in moltissime app moderne ed è ampiamente usato anche da colossi quali Google, verizon e molti altri. Tra i vari servizi offerti vi è inoltre un db online, che ci ha permesso di avere versioni consistenti del db durante lo sviluppo anche lavorando su due pc diversi.



Figure 3.1: MongoDB logo

3.1.2 Express [5]

Express è un framework open source per applicazioni web per Node.js. È stato progettato per creare web application e API ed ormai definito il server framework standard de facto per Node. L'autore lo descrive come un'infrastruttura di base minimale, estendibile con innumerevoli plugin. Express è fa parte del backend.



Figure 3.2: Express logo

3.1.3 React [15]

React viene definito come una libreria open-source per costruire interfacce utente utilizzando Javascript. È stato sviluppato da Facebook per uso interno e poi rilasciato verso la comunità, che ora si occupa di mantenerlo e aggiornarlo. Occupandosi solo del rendering dei componenti, sono necessarie librerie per il routing e la gestione dello stato, le più utilizzate sono React-router e Redux. Un'applicazione React è costituita da uno o più componenti, che vengono renderizzati utilizzando la sintassi JSX.

React vs Vue vs Angular

Dal punto di vista didattico, abbiamo pensato di utilizzare il framework (o libreria) che offrisse il maggior supporto possibile agli utenti (cioè noi sviluppatori), per questo motivo, nonostante la sua duttilità e facilità di utilizzo, abbiamo scelto di non utilizzare Vue. Inoltre abbiamo pensato visto la natura del progetto più orientato al microservizio di usare React rispetto ad Angular poiché più flessibile. Inoltre in ottica futura React permette di espandere il progetto in versione mobile, grazie a React Native.



Figure 3.3: Comparison between framework

React rendering

Questo è l'esempio più semplice, creiamo un componente che contiene una stringa "Hello, world!":

```
const element = <h1>Hello, world!</h1>;
```

Per renderizzare questo elemento, posso utilizzare ReactDOM.Render :

```
ReactDOM.render(  
  element,  
  document.getElementById('root')  
) ;
```

React gestisce quasi autonomamente la renderizzazione degli elementi, in modo da aggiornare la UI ogni volta che un elemento viene aggiornato. In più React riesce ad aggiornare solo l'elemento del DOM che richiede di essere aggiornato, non andando a modificare ogni volta l'interno albero.

React props e state

Posso anche aggiungere un'espressione all'interno di un elemento utilizzando le parentesi graffe, come ad esempio:

```
const element = <img src={user.avatarUrl}></img>;
```

Una funzionalità fondamentale di React è quella di poter passare informazioni tra componenti padre a componenti figli, utilizzando le props. Ad esempio:

```
function Ciao(props) {  
  return <h1>Ciao, {props.nome}</h1>;  
}  
  
const elemento = <Ciao nome="Sara" />;  
  
ReactDOM.render(  
  elemento,  
  document.getElementById('root')  
) ;
```

I componenti di React hanno un ciclo di vita, in cui possono essere specificati determinati comportamenti quando il componente viene "montato" oppure eliminato dal DOM. Oltre a dare informazioni ad un componente dall'esterno, utilizzando le props, è spesso utile avere uno stato indipendente all'interno della applicazione. Questo può essere fatto utilizzando le classi componenti oppure utilizzando le funzioni componenti, ma in due modi diversi. Prima vediamo come nelle classi:

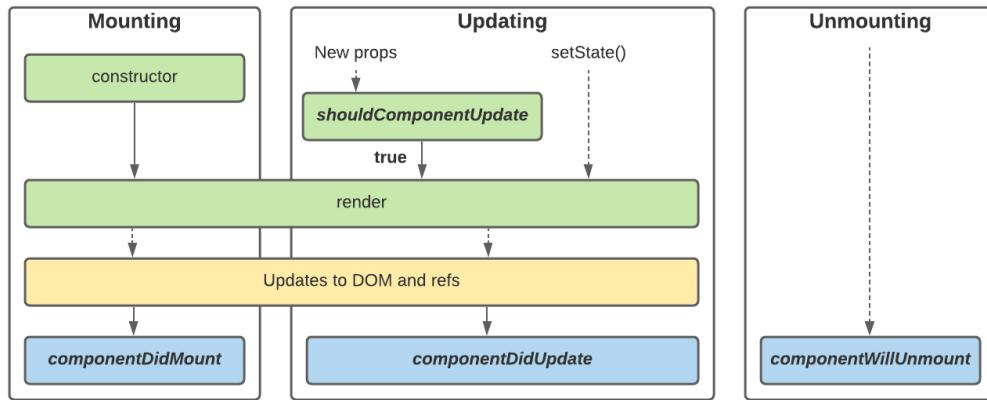


Figure 3.4: React component lifecycle

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  render() {
    return (
      <div>
        <h2>Sono le {this.state.date.toLocaleTimeString()}</h2>
      </div>
    );
  }
}
```

mentre nelle funzioni componenti, basta utilizzare le variabili solite di javascript, oppure è consigliato utilizzare gli hooks e useState() In questo esempio, la data da mostrare è incapsulata nello stato e ogni accesso avviene tramite "this.state". E' importante notare che non è possibile modificare lo stato direttamente, ma è necessario utilizzare "this.setState()"

React class

Mostrate queste caratteristiche di React, possiamo vedere un esempio di come si interfaccino in un componente completo:

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
```

```

        this.state = {date: new Date()};
    }

componentDidMount() {
    this.timerID = setInterval(
        () => this.tick(),
        1000
    );
}

componentWillUnmount() {
    clearInterval(this.timerID);
}

render() {
    return (
        <div>
            <h1>Ciao, mondo!</h1>
            <h2>Sono le {this.state.date.toLocaleTimeString()}.</h2>
        </div>
    );
}
}

```

Questa classe serve a renderizzare un orologio, creando un timer quando l'elemento viene montato ed eliminandolo quando l'elemento viene eliminato.

React accessibility

React utilizzando la sintassi JSX permette di includere gli attributi aria (Accessible Rich Internet Applications) nell' HTML

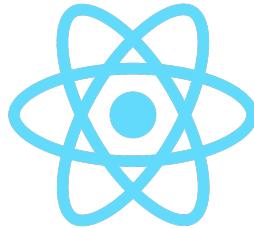


Figure 3.5: React logo

Per mantenere invece valida la semantica degli elementi in React è possibile utilizzare i `<Fragment>` oppure raggruppare elementi all'interno di "empty tag".

Inoltre, React da la possibilità di dare una label ad ogni elemento dei form, utilizzando l'attributo "htmlFor"

Node [10]

Essendo un runtime di JavaScript asincrono e event-driven, è progettato per costruire applicazioni web scalabili. È caratterizzato da un'architettura single threaded e un comportamento non bloccante dell'I/O. È estremamente veloce e leggero, fa un uso efficiente delle risorse e permette di avere un forte controllo della logica dell'app e dell'ambiente, il che permette di gestire diversi utenti con poche risorse.



Figure 3.6: Node logo

3.2 Server

Di seguito verranno elencate le tecnologie e gli strumenti utilizzati per l'implementazione del server.

3.2.1 OpenSSL [13]

È l'implementazione open-source di protocolli TLS e SSL che è stata usata per la generazione della chiave RSA e per la firma dei certificati per ottenere una connessione sicura e privata (https).



Figure 3.7: OpenSSL logo

3.2.2 bcrypt [2]

Pacchetto npm utilizzato per la gestione delle password. Permette di generare hash sicuri e valori di sale appropriati, senza bisogno di salvarli in chiaro nel db, né di recuperarli al momento di confronto delle password.

3.2.3 dotenv [4]

Pacchetto che permette di separare i segreti del db dal codice sorgente. Utile per avere un punto comune dove tenere tutti i segreti e non usare delle stringhe hard-coded nel codice sorgente (logo in figura 3.8).



Figure 3.8: .env logo

3.2.4 jsonwebtoken [6]

Jsonwebtoken (in figura 3.9) è uno standard sicuro ed autocontenuto per la trasmissione sicura di oggetti json. Le informazioni possono essere verificate e da considerarsi fidate perché sono firmate digitalmente. I jsonwebtoken sono stati usati per implementare l'access control e l'authorization nella nostra applicazione. In particolare viene fornito un access-token che ha una validità di quindici minuti e permette l'accesso alle funzioni del sito, e un refresh-token che permette di creare nuovi access-token. L'utente ha bisogno di conservare questi due token che sono molto piccoli e hanno al loro interno l'id dell'utente, in questo modo si permette di limitare moltissimo il peso dei pacchetti scambiati tra client e server e di renderli sicuri e verificabili. Tecnologia molto interessante e utile.



Figure 3.9: JWT logo

3.2.5 Mongoose [8]

È uno strumento di modellazione (in figura 3.10) elegante per oggetti da inserire in mongodb. Permette una modellazione veloce, supporta la tipizzazione e ha funzioni integrate che rendono più semplice e veloce la lettura e la scrittura delle query.



Figure 3.10: Mongoose logo

3.2.6 Multer [9]

Multer è un middleware di node.js che permette di maneggiare multipart/form-data, prevalentemente usato per l'upload di file. È stato usato per caricare sul server le immagini di profilo degli utenti.



Figure 3.11: Multer logo

3.2.7 Nodemailer [11]

Nodemailer è un modulo per Node.js che permette di mandare mail in maniera semplice e veloce. È molto facile configurarlo e rende altrettanto semplice l'invio di semplici mail.

3.2.8 Nodemon [12]

Nodemon è un'utility che permette di monitorare i cambiamenti nel codice sorgente facendone automaticamente il restart. Utility già sfruttata durante i laboratori ed estremamente comoda in fase di sviluppo.



Figure 3.12: Nodemailer logo



Figure 3.13: Nodemon logo

3.2.9 Seedrandom [18]

Seedrandom è un generatore di numeri casuali per javascript. Per avere una buona casualità nella scelta delle coppie da votare il generatore di numeri casuali di default di javascript non era abbastanza, in particolare troppo spesso si vedevano le stesse sequenze di scelta.

Per questo motivo è stato usato un generatore esterno, che oltre ad avere una possibile inizializzazione tramite seed, è anche capace di accumulare entropia ed usarla per generare numeri davvero casuali, permettendo una migliore distribuzione di coppie da scegliere.

3.2.10 Sharp [19]

È un modulo ad alta velocità per node.js per la manipolazione di immagini. Viene tipicamente usato per la conversione di grandi immagini in immagini più piccole e facilmente utilizzabili sul web. Essendo molto ottimizzata riesce ad essere quattro o cinque volte più veloce degli altri principali manipolatori di immagini su operazioni come il resizing. Questo pacchetto è stato usato per fare il resizing delle immagini del profilo inviate dal client, facendo in modo di avere sul server immagini dello stesso formato e che occupino meno delle originali.

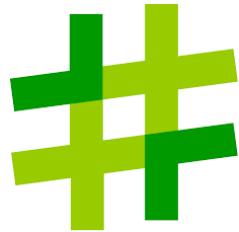


Figure 3.14: Sharp logo

3.2.11 Postman [14]

Postman è uno strumento utilissimo per il testing delle api, che permette di fare richieste al server e controllarne le risposte. Verrà meglio discusso nel capitolo 5



POSTMAN

Figure 3.15: Postman logo

3.3 Client

3.3.1 Axios [1]

Axios è una libreria che permette di fare richieste HTTP basata sull'utilizzo delle Promise. Questo permette di scrivere codice asincrono con facilità, oltre ad evitare il problema noto della "Pyramid of Doom". Axios, se utilizzato a



Figure 3.16: Axios logo

livello server utilizza il modulo "http" nativo di NodeJs, mentre da browser, cioè quando viene utilizzato nei framework lato client, esegue richieste XHR, ovvero XMLHttpRequest

3.3.2 Redux [17]

Redux è uno "state container" che permette cioè di salvare lo stato di una applicazione per permettergli di comportarsi coerentemente in diversi ambienti. Spesso Redux viene utilizzato insieme a React, per sopperire ad alcune mancanze, come ad esempio il "Prop Chain Problem". Redux è composto principalmente di 3 componenti. Il primo è lo Store. Lo Store è una sorta di Database dove vengono salvati i dati che lo sviluppatore vuole che siano coerenti tra i diversi componenti dell'applicazione. Lo store ha diversi compiti, innanzitutto deve fornire dei getter (getState()) e dei setter (actions, che vedremo tra poco) e permettere di registrare dei listener per poter essere notificati delle modifiche. Il secondo sono le Action. Le Action sono dei Javascript Objects (Json) che rappresentano l'informazione che deve essere modificata all'interno dello store. Sono composti da un campo type, che contiene una stringa che definisce che azione deve essere compiuta e uno più valori di payload (può contenere un json a sua volta). L'ultimo componente sono i Reducers. I Reducers sono delle funzioni che determinano il cambio dei valori all'interno dello Store, e conseguentemente, dello stato dell'applicazione. E' importante notare come i Reducers siano delle funzioni pure, ovvero non cambiano direttamente lo stato di una variabile, ma ritornano un nuovo stato. Come vengono combinati questi tre componenti? Innanzitutto abbiamo bisogno di un evento, qualcosa che dica all'applicazione di cambiare stato. Chiamiamo questa azione trigger e questo trigger causa uno dispatch(action(payload)). A questo punto il reducer smisterà l'action in base al suo campo "type" e ritornerà un nuovo stato elaborando il payload passato all'action.



Figure 3.17: Redux logo

3.3.3 Bootstrap [3]

Per poter scrivere un applicazione mobile-first, responsive in modo agevole, ma comunque avendo a cuore la sua resa grafica, abbiamo pensato di utilizzare la libreria Bootstrap. In particolare, utilizzando un framework frontend che utilizzava JSX, abbiamo pensato di non sovraccaricare ulteriore codice Jquery, necessario per bootstrap utilizzando la libreria "react-bootstrap", questo ci ha permesso di utilizzare componenti bootstrap direttamente a partire dal codice HTML, come ad esempio nel caso dei

```
<Button>  
al posto del classico  
<button type="button" class="btn btn-primary">
```



Figure 3.18: Bootstrap logo

3.3.4 React Router [16]

React Router è una collezione di strumenti per il routing all'interno di applicazione React che permette di utilizzare uno stile dichiarativo nella scrittura delle Routes

3.3.5 JSX

JSX è una particolare sintassi con cui il codice HTML viene convertito in codice Javascript, che a sua volta produce componenti React. JSX è diventato popolare infatti proprio con React, anche se adesso viene utilizzato con altri framework e linguaggi diversi dal Javascript, come TypeScript. JSX viene considerato sicuro come sintassi, poiché effettua, prima di essere compilato in Javascript, una sanificazione del codice, rendendolo praticamente immune ad attacchi tipo XSS (cross-site-scripting)

3.3.6 Typescript

Typescript è un superset di Javascript creato dalla Microsoft che mira a risolvere alcuni problemi noti di Js, come il typechecking. Innanzitutto, essendo un superset, è perfettamente compatibile con tutte i framework js esistenti, tra cui React che abbiamo scelto per il progetto ed è possibile adottarlo gradualmente nei progetti. Typescript introduce la definizione statica dei tipi, che permette di validare il codice prima che venga eseguito, risolvendo molti errori e permettendo di velocizzare il processo di debug e testing, senza scrivere codice aggiuntivo. Per poter sperimentare con questo nuovo linguaggio, abbiamo riscritto due componenti funzione della nostra applicazione adottando questo linguaggio e si sono notate subito alcune potenziali fallacie, risolte tramite questo strumento.



Figure 3.19: Typescript logo

3.3.7 Docker

Per poter sviluppare all'interno di container l'applicazione, abbiamo scelto di utilizzare Docker. Docker utilizza il kernel Linux per poter eseguire dei container isolati dal sistema operativo in cui solo alcune risorse vengono utilizzate (come la scheda di rete). I container Docker possono essere compilati a partire da una immagine Linux (le distribuzioni più utilizzate sono Busybox, Alpine e Ubuntu) oppure da zero (scratch) e le istruzioni per poter scaricare pacchetti o eseguire comandi vengono impartite nel file "Dockerfile". E' poi possibile compilare questo file in una immagine, che a sua volta è possibile eseguire quando è necessario utilizzare l'applicazione. Inoltre, Docker-compose permette di scrivere applicazioni basate su microservizi specificando come due o più container si interfacciano tra di loro, specificando le impostazioni di rete (come le porte utilizzate).



Figure 3.20: Docker logo

Chapter 4

Codice

Di seguito verranno discussi alcuni tra gli aspetti più rilevanti del codice scritto

4.1 Frontend

4.1.1 React Hooks

All'inizio dello sviluppo del progetto, non avendo mai sviluppato applicazioni con React, si è pensato di sviluppare ogni componente utilizzando classi componenti. Questo ha permesso la scrittura dei primi componenti, il collegamento con l'app e il routing. Studiando ulteriormente la documentazione di React ci siamo accorti che nelle ultime versioni questo framework aveva introdotto una nuova tecnologia chiamata React Hooks. Questa introduzione ha permesso di "riciclare" i vecchi componenti scritti come classi in semplici funzioni, scrivendo codice più pulito, funzionale e risolvendo alcuni problemi noti di React stesso. Come ad esempio il riutilizzo della logica tra componenti diversi e la semplificazione dei componenti complessi. Ad esempio, un possibile confronto tra classi e funzioni potrebbe essere il seguente, usando i React Hooks:

```
function ShowCount(props) {
  const [count, setCount] = useState();

  useEffect(() => {
    setCount(props.count);
  }, [props.count]);

  return (
    <div>
      <h1> Count : {count} </h1>
    </div>
  );
}
```

mentre usando le classi:

```
class ShowCount extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0
    };
  }
  componentDidMount() {
    this.setState({
      count: this.props.count
    })
  }

  render() {
    return (
      <div>
        <h1> Count : {this.state.count} </h1>
      </div>
    );
  }
}
```

La classe è notevolmente più lunga, poichè le funzioni non hanno ovviamente bisogno di costruttore e la chiamata a render() non viene effettuata, mentre viene restituito direttamente il codice JSX. Un altro fattore da tenere in considerazione è che la gestione del ciclo di vita dei componenti React viene effettuata attraverso l'hooks useEffect(). Altri elementi importanti:

- Non è più necessario dover accoppiare le funzioni ai componenti (function binding)
- Non bisogna più indicare gli elementi della class con la keyword "this"
- E' più facile disaccoppiare la logica dalla UI, rendendo entrambi più usabili

4.2 Backend

Lo sviluppo lato backend ha cercato di accentuare e sfruttare il più possibile la natura asincrona del framework per permettere un corretto funzionamento dell'applicazione. Nella scrittura di codice si è sempre cercato di tenere in conto il potenziale costo degli accessi al db cercando di minimizzarli quando possibile e di evitare computazioni troppo onerose cercando comunque di demandare al db più computazione possibile in modo tale da tenere il server libero, pronto a rispondere ad eventuali nuove richieste. Una delle parti più importanti è ovviamente il recupero della classifica e quella qui sotto è l'implementazione attualmente usata.

```

const myId = req.user._id
const city = req.query.city
const state = req.query.state
const country = req.query.country
const age = req.query.age
const gender = req.query.gender

let numberofPeopleToRank
if(req.query.n){
    numberofPeopleToRank = req.query.n
} else {
    numberofPeopleToRank = 10
}

let filter = {}
if (gender) {
    filter["gender"] = gender
}
if(city) {
    filter["city"] = city
} else if (state) {
    filter["state"] = state
} else if (country) {
    filter["country"] = country
}
let users = []
let userFound = false
User.find(filter, '_id username picture birthDate').sort({'numberofVotes': -1})
    .then(rankedUsers => {
        if (!rankedUsers) {
            console.log("users not found")
            return res.send(users)
        }
        let i = 1;
        //filter age
        if (age && typeof age !== undefined) {
            rankedUsers = rankedUsers.filter(u => getAge(u.birthDate) == age)
        }
        for (let u of rankedUsers){
            if(i <= numberofPeopleToRank){
                if (u.username === user.username) {
                    userFound = true
                }
                users.push(createUser(u._id, i, u.username, u.picture))
            } else if (!userFound) {
                if (u.username === user.username) {

```

```

        userFound = true
        users.push(createUser(u._id, i, u.username, u.picture))
        break;
    }
}
i++
}
return res.send({"array" : users})
})

```

Ci sarebbe stato modo di tenere il server ancora meno impegnato a livello di computazione utilizzando la funzione di aggregazione **\$limit** ma si è preferito favorire l'esperienza utente, aggiungendo tra i risultati la posizione dell'utente in classifica, anche se l'utente richiede di vedere solo i primi cinque ed è ventesimo, sarà in grado di vedere la sua posizione.

Chapter 5

Test

5.1 Server

I test per il corretto funzionamento delle chiamate al server sono state fatte tramite Postman. Queste chiamate non sono servite solamente per controllare il corretto funzionamento della webapp, ma ci hanno permesso attraverso delle collezioni condivise di avere un punto di controllo condiviso. L'utilizzo di questo strumento hanno permesso una corretta coordinazione tra backend e frontend, che sono potuti procedere in parallelo. Il backend sviluppava una nuova funzionalità, la testava e quando pronta la pushava e scriveva la richiesta da utilizzare sulla collezione condivisa su Postman insieme ad una risposta in modo da far capire al frontend che risposta aspettarsi. In figura 5.1 troviamo l'esempio di una richiesta.

5.2 Testing con utenti

Si è cercato di coinvolgere il più possibile gli utenti durante il processo di sviluppo, cercando di seguire un processo agile e ascoltando sempre il loro parere senza mai scrivere troppo codice in completa autonomia. Più che fare delle sessioni a posteriori di confronto con gli utenti si è cercato di svilupparlo proprio insieme a loro e per farlo si chiedeva a più persone possibile di valutare il lavoro svolto in una determinata pagina e cosa secondo loro non andasse, accontentandoli il più possibile cercando anche di rendere la webapp il più user-friendly possibile.

Una volta finito il sito abbiamo invece chiesto agli utenti di svolgere il seguente questionario (costruito grazie ai moduli di Google che hanno permesso di automatizzare la raccolta di risultati e una migliore presentazione rispetto a quella del pdf allegato):

Attrattività

Valutazione estetica della webapp

*Campo obbligatorio

1. Ti piace la schermata di login? *

Contrassegna solo un ovale.

1	2	3	4	5	
Poco	<input type="radio"/> Molto				

2. Ti piace la schermata di signup? *

Contrassegna solo un ovale.

1	2	3	4	5	
Poco	<input type="radio"/> Molto				

3. Ti piace la pagina del profilo? *

Contrassegna solo un ovale.

1	2	3	4	5	
Poco	<input type="radio"/> Molto				

4. Ti piace la home? *

Contrassegna solo un ovale.

1	2	3	4	5	
Poco	<input type="radio"/> Molto				

5. Ti piace la pagina delle analytics? *

Contrassegna solo un ovale.

1 2 3 4 5

Poco Molto

6. Qual è la tua pagina preferita? Come mai?

7. Qual è la pagina che ti è piaciuta di meno? Come mai?

Esegui dei task

prova ad eseguire questi task

Prova a registrarti

8. È stato facile registrarsi? *

Contrassegna solo un ovale.

1 2 3 4 5

Per niente Facilissimo

Fai logout

9. È stato facile fare logout? *

Contrassegna solo un ovale.

1 2 3 4 5

Per niente Facilissimo

Prova a fare login

10. È stato facile? *

Contrassegna solo un ovale.

1 2 3 4 5

Per niente Facilissimo

Prova a votare qualcuno

11. È stato facile? *

Contrassegna solo un ovale.

1 2 3 4 5

Per niente Facilissimo

Prova a cambiare la tua immagine di profilo

12. È stato facile? *

Contrassegna solo un ovale.

1 2 3 4 5

Per niente Facilissimo

Ti sei divertito?

13. Ti è piaciuto usare la nostra app? *

Contrassegna solo un ovale.

1 2 3 4 5

Per niente Moltissimo

14. La consiglieresti ad un tuo amico? *

Contrassegna solo un ovale.

sicuramente

probabilmente no

Altro: _____

15. Ti piacciono i colori usati? *

Contrassegna solo un ovale.

1 2 3 4 5

per niente Moltissimo

16. Il sito ti è sembrato intuitivo? *

Contrassegna solo un ovale.

1 2 3 4 5

Per niente Moltissimo

17. Ti senti al sicuro utilizzando il nostro sito? *

Teniamo moltissimo alla sicurezza dei nostri utenti e ci siamo impegnati per rendere il sito il più sicuro possibile.

Contrassegna solo un ovale.

1 2 3 4 5

Per niente Moltissimo

18. Hai riscontrato problemi utilizzando l'app? Se sì quali? se non hai riscontrato problemi puoi saltare questa domanda

19. Usa questo spazio per scrivere un commento sull'applicazione

Grazie mille della disponibilità!

grazie dallo staff di RankMe

Questi contenuti non sono creati né avallati da Google.

Google Moduli

5.2.1 Risultati

Dal questionario sono stati raccolti i seguenti risultati:

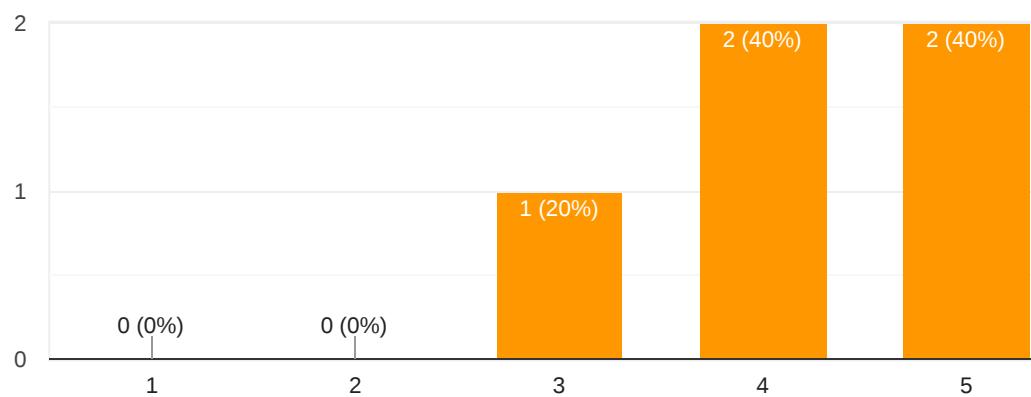
Attrattività

5 risposte

[Pubblica i dati di analisi](#)

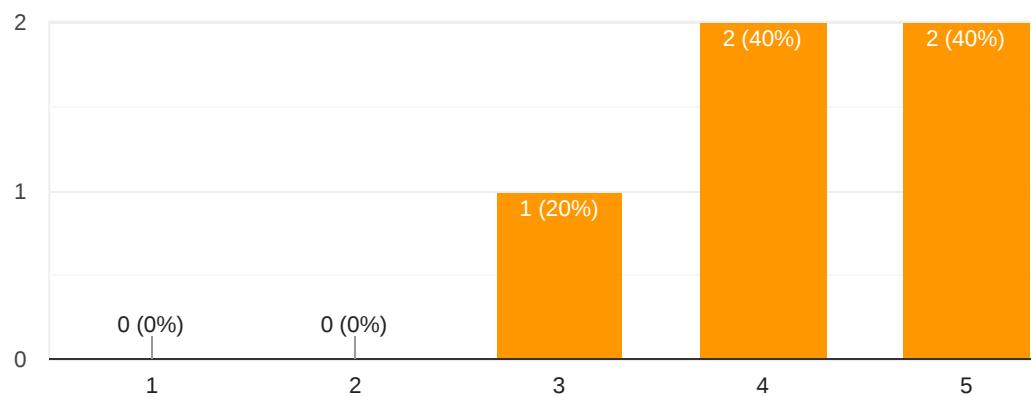
Ti piace la schermata di login?

5 risposte



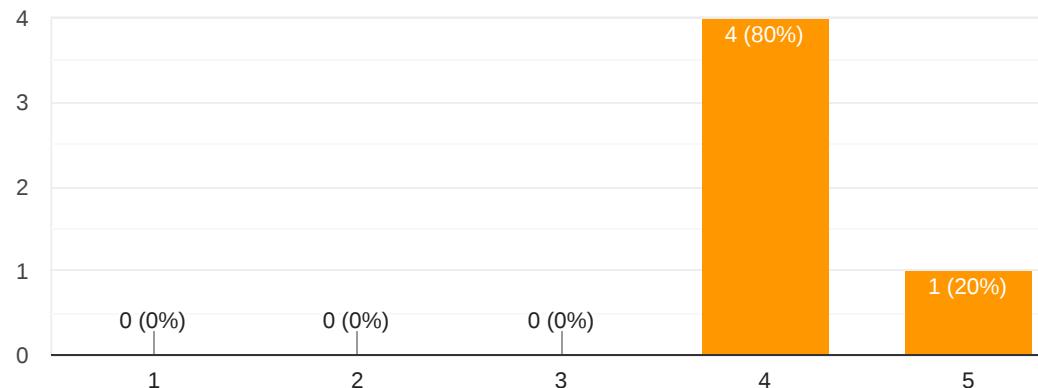
Ti piace la schermata di signup?

5 risposte



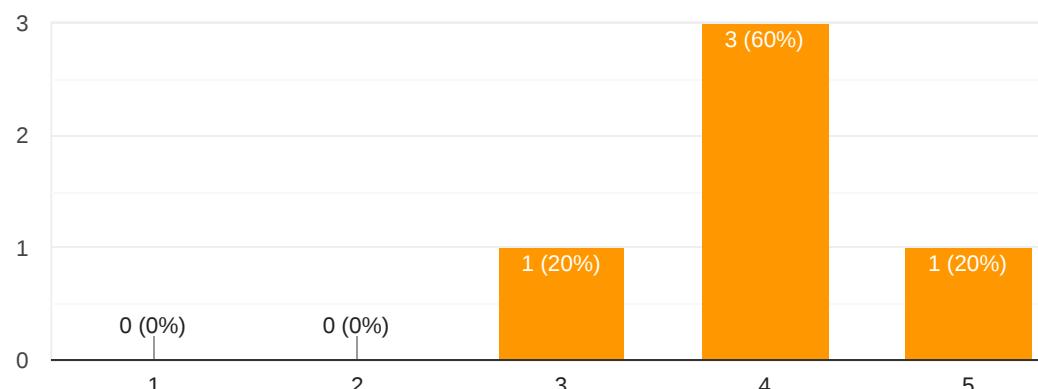
Ti piace la pagina del profilo?

5 risposte



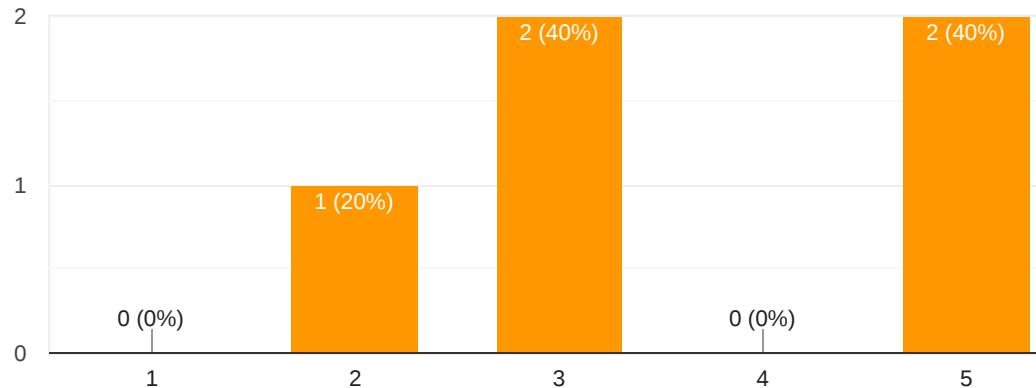
Ti piace la home?

5 risposte



Ti piace la pagina delle analytics?

5 risposte



Qual è la tua pagina preferita? Come mai?

5 risposte

La home in quanto è breve e concisa, si capisce in fretta quello che richiede.

La home dove posso scegliere tra i profili

Analytics

Profilo molto carino

Analitics, perché ci sono tante informazioni utili



Qual è la pagina che ti è piaciuta di meno? Come mai?

5 risposte

Analytics, in quanto essendo appena iscritta non ho ancora persone che mi abbiano votato, di conseguenza la pagina risulta vuota e quello che c'è è molto ammazzato

Analytics un po' spoglie

Profile

Analytics non mi sono piaciuti i grafici un po' spogli

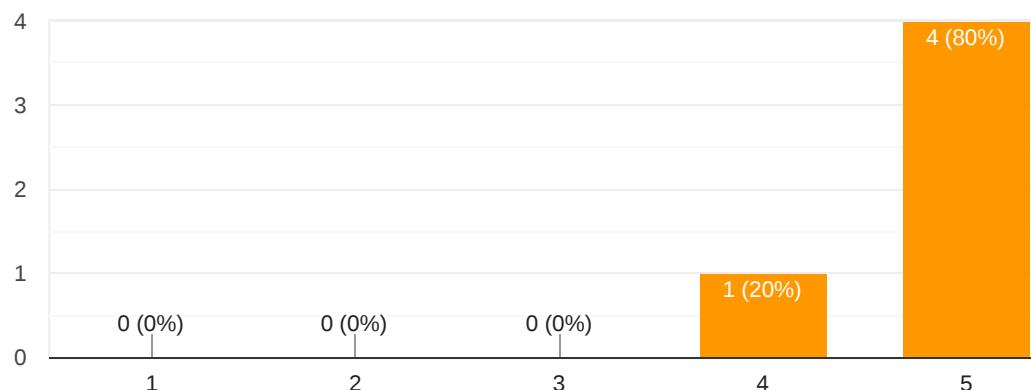
home, troppo attaccate le foto su pc

Esegui dei task

Prova a registrarti

È stato facile registrarsi?

5 risposte

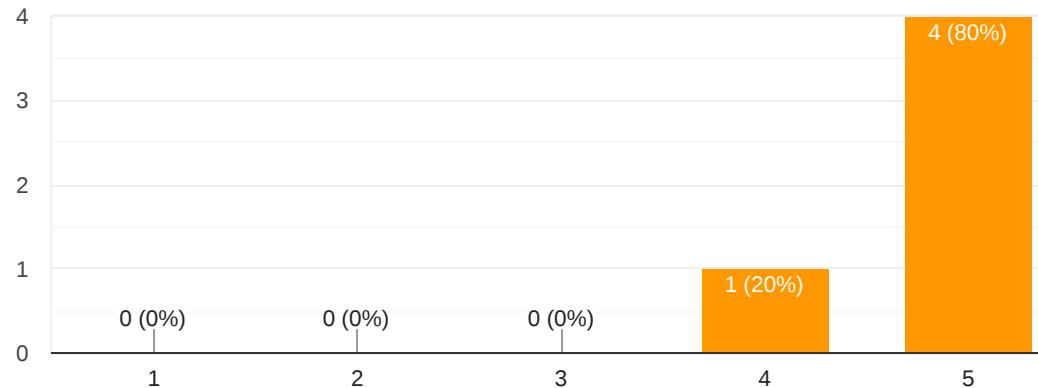


Fai logout



È stato facile fare logout?

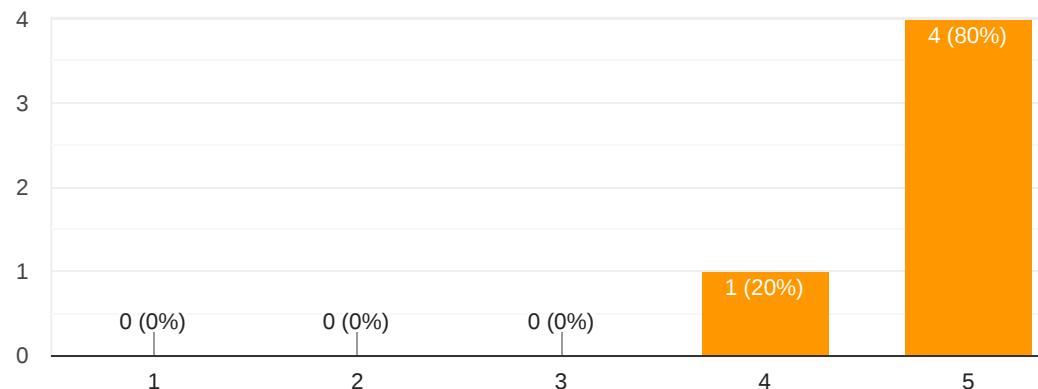
5 risposte



Prova a fare login

È stato facile?

5 risposte

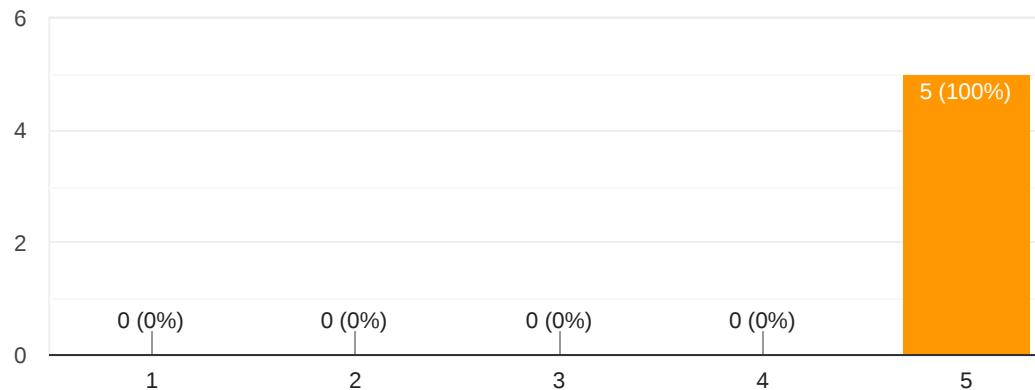


Prova a votare qualcuno



È stato facile?

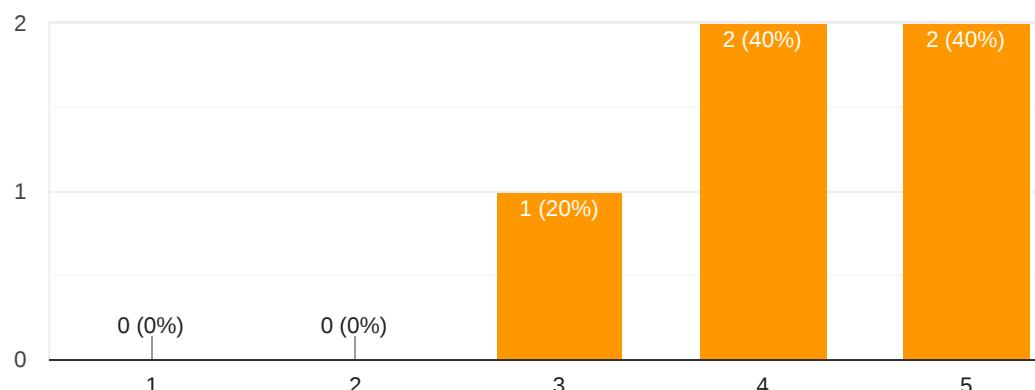
5 risposte



Prova a cambiare la tua immagine di profilo

È stato facile?

5 risposte

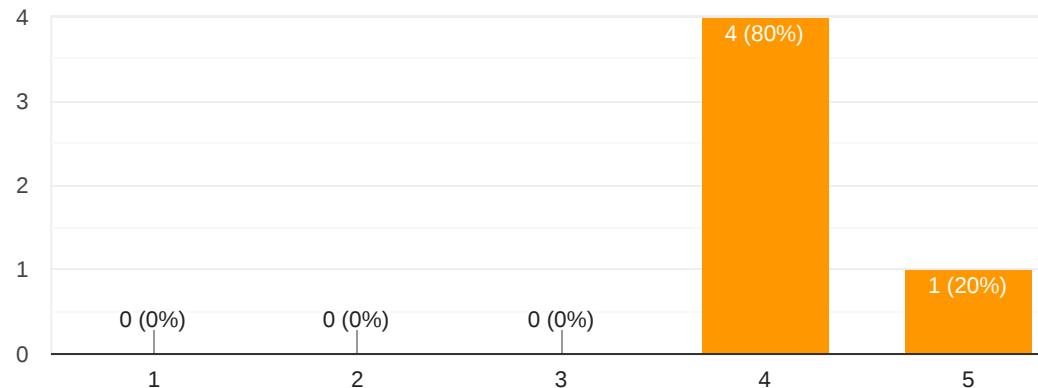


Ti sei divertito?



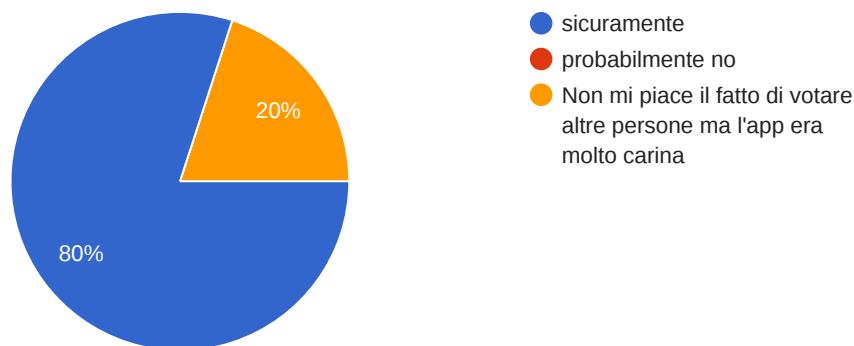
Ti è piaciuto usare la nostra app?

5 risposte



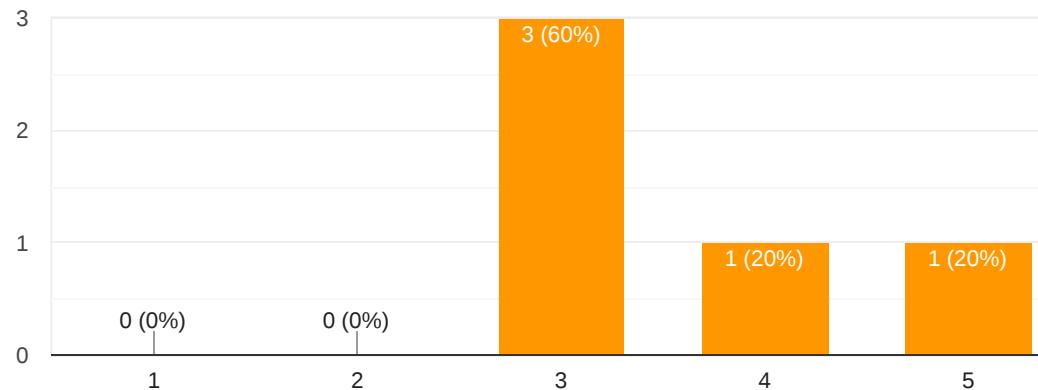
La consigliresti ad un tuo amico?

5 risposte



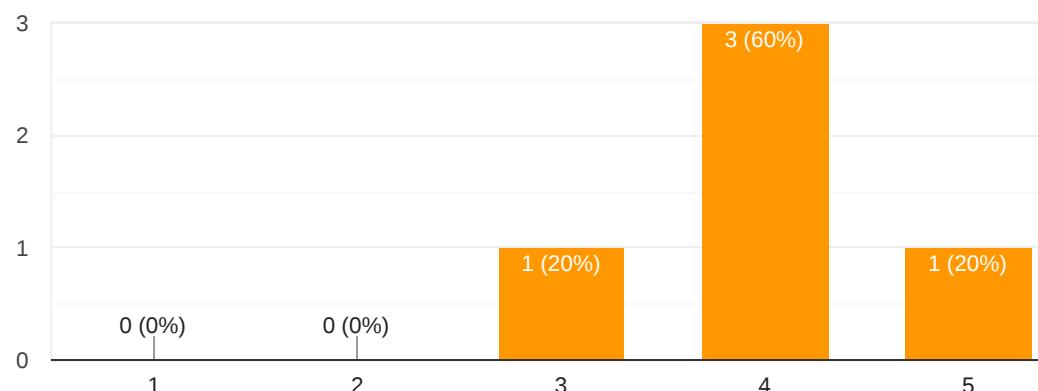
Ti piacciono i colori usati?

5 risposte



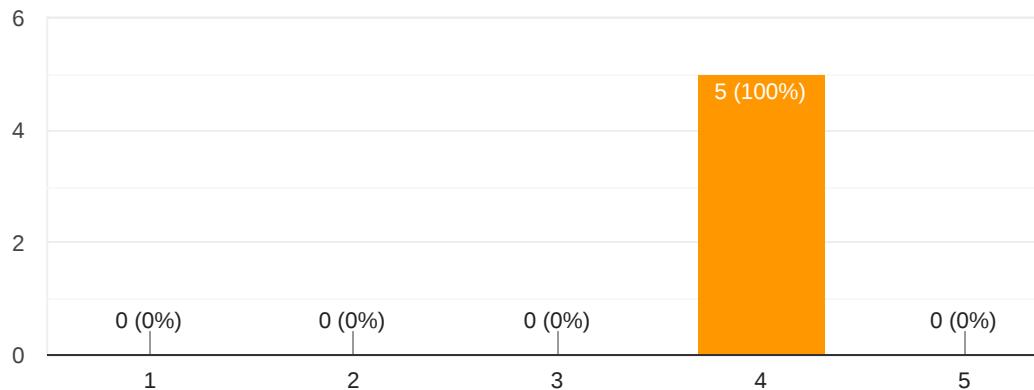
Il sito ti è sembrato intuitivo?

5 risposte



Ti senti al sicuro utilizzando il nostro sito?

5 risposte



Hai riscontrato problemi utilizzando l'app? Se si quali? se non hai riscontrato problemi puoi saltare questa domanda

1 risposta

Non ho capito come caricare la foto

Usa questo spazio per scrivere un commento sull'applicazione

3 risposte

Bell'idea e sito ben comprensibile ed utilizzabile

Mi è piaciuta molto, alcune cose non le ho capite al volo ma nel complesso mi è piaciuta

Molto carino questo social network

Grazie mille della disponibilità!

Questi contenuti non sono creati né avallati da Google. [Segnala una violazione](#) - [Termini di servizio](#) - [Norme sulla privacy](#)



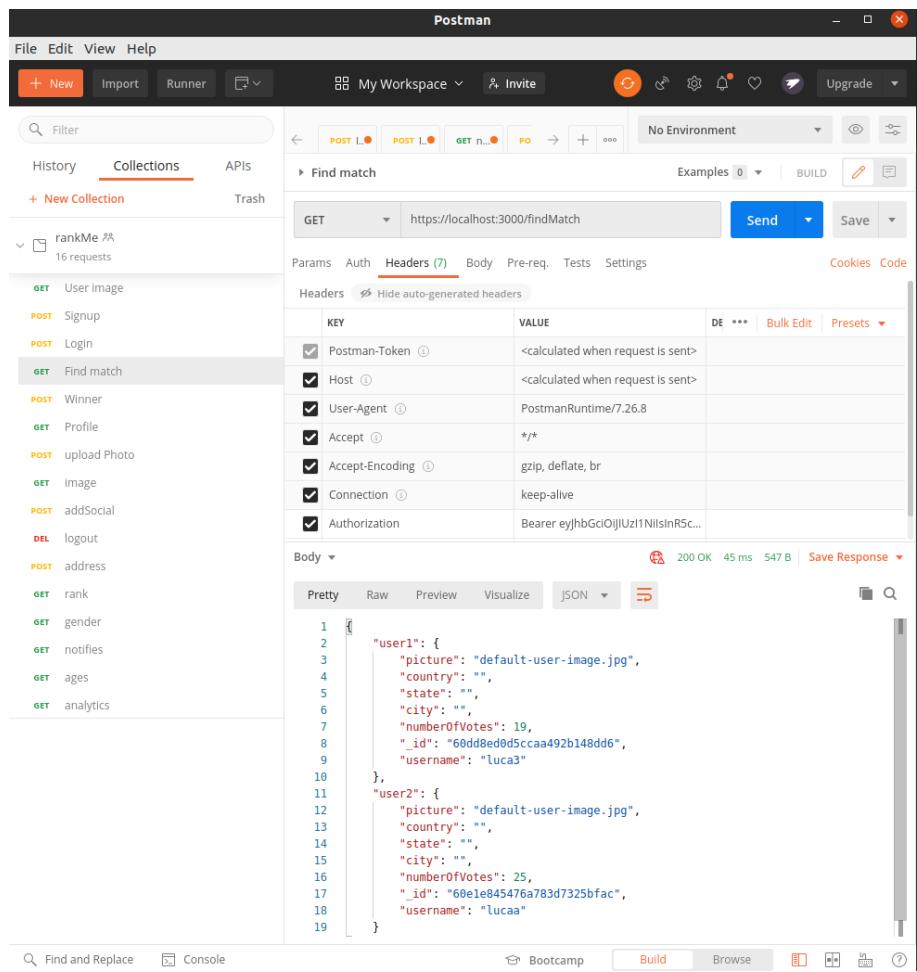


Figure 5.1: Esempio di richiesta con Postman

Chapter 6

Deployment

Per consentire un rilascio agile dell'applicazione abbiamo predisposto una multi-applicazione utilizzando il servizio Docker-compose di Docker. Per fare ciò abbiamo definito un file docker-compose.yml dove abbiamo indicato i tre servizi principali dell'applicazione, server, client e database MongoDB, per essere eseguiti ciascuno in un container e una rete che permettesse il collegamento tra essi. Per fare ciò ogni servizio è definito un file Dockerfile che specifica l'immagine base per ogni applicazione, (alpine per server e client, ubuntu per mongodb) e le istruzioni necessarie a metterli in esecuzione. Una possibile alternativa sarebbe stata quella di utilizzare un servizio di hosting o di noleggio macchine software (aruba o ec2 di Amazon Web Services) per poter mettere online l'applicazione, ma questo avrebbe causato dei problemi come ad esempio dover aggiornare l'applicazione ogni volta che venivano effettuate delle modifiche. Per questo motivo abbiamo deciso di continuare lo sviluppo sempre a livello locale, visto anche il fatto che per l'utilizzo iniziale dell'applicazione non era necessario che i membri del gruppo avessero la stessa copia del database.

Per poter eseguire l'applicazione da qualsiasi dispositivo in cui è installato il runtime Docker è necessario lanciare i seguenti comandi:

```
//Per compilare le immagini dei servizi  
docker-compose build  
  
//Per lanciare l'applicazione in background "-d"  
docker-compose up -d
```

Chapter 7

Conclusioni

Il progetto da noi realizzato richiede sicuramente una buona base di utenti iniziale per poter essere messo in commercio oppure distribuito verso il pubblico, ciò nonostante, i test da noi eseguiti su una scala di 15-20 persone virtuali da noi create hanno dimostrato che l'applicazione funziona in tutti i suoi elementi principali. È stato interessante lavorare su un progetto che usa javascript sia lato frontend che backend e sicuramente formativo, questo stack ci ha aiutati a scrivere del codice molto velocemente e la natura asincrona di node ci ha permesso di scrivere codice in stile funzionale molto leggibile e riutilizzabile.

7.0.1 Lavori futuri

Per il futuro abbiamo pensato ad alcune modifiche per rendere più social la nostra applicazione aggiungendo ulteriori funzionalità che per noi potrebbero essere più coinvolgenti, come ad esempio nuove analytics e nuovi badge da ottenere da parte degli utenti insieme anche a classifiche già filtrate che permettano agli utenti di consultarle senza doverle creare manualmente. Per la distribuzione abbiamo pensato che il prossimo passo sia quello di noleggiare un macchina su AWS ed eseguire lì il nostro container Docker.

Un altro passo fondamentale sarà una sessione di testing con gli utenti del sito attuale, per permetterci di raccogliere tutte le critiche e le informazioni necessarie per rendere l'applicazione il più semplice utilizzabile possibile.

Bibliography

- [1] Axios - github.
- [2] bcrypt - npm.
- [3] Bootstrap for react.
- [4] dotenv - npm.
- [5] Express.
- [6] jsonwebtoken - npm.
- [7] Mongodb.
- [8] mongoose - npm.
- [9] multer - npm.
- [10] Node.
- [11] Nodemailer.
- [12] Nodemon.
- [13] Openssl.
- [14] Postman.
- [15] React.
- [16] React-router.
- [17] Redux.
- [18] seedrandom - npm.
- [19] sharp - npm.
- [20] Wikipedia. Mean (solution stack).