PROGETTO CINEWEB

Applicazioni e Servizi Web

Bedei Andrea
matricola 0001126957 - andrea.bedei2@studio.unibo.it
Bertuccioli Giacomo Leo
matricola 0001136879 - giacomo.bertuccioli@studio.unibo.it)
Notaro Fabio
matricola 0001126980 - fabio.notaro2@studio.unibo.it

23 Dicembre 2024

Capitolo 1

INTRODUZIONE

Lo scopo principale del progetto è lo sviluppo di un portale di cinema in cui vengono programmate proiezioni di vari cinema ed è possibile prenotare posti e pubblicare recensioni, il tutto cercando di rispettare le buone pratiche e metodologie viste nei corsi di Tecnologie Web e Applicazioni e Servizi Web.

Le motivazioni che ci hanno spinto a scegliere proprio questo progetto sono molteplici e variegate:

- interesse per il dominio cinematografico
- rilevanza ed applicabilità → il settore dell'intrattenimento è rilevante nella vita quotidiana di persone, pertanto progettare e realizzare un sistema che faciliti l'interazione tra gestori di cinema e spettatori offre un'opportunità concreta di creare una piattaforma che possa avere applicazioni reali e un impatto positivo sull'esperienza utente
- sfida tecnica → il progetto proposto propone interessanti ostacoli (sia di progettazione che implementativi) che lo rendono molto stimolante dal punto di vista didattico, come la gestione di dati e aggiornamenti in tempo reale, l'integrazione con API esterne e la progettazione e realizzazione di un adeguato sistema di notifiche
- opportunità di mettersi alla prova con il rispetto delle best practices emerse nei corsi di Tecnologie Web e Applicazioni e Servizi Web.

Capitolo 2

REQUISITI

La presente sezione riporta l'elenco dei requisiti rilevanti che sono emersi in fase di analisi e che sono stati rispettati in fase di sviluppo.

Si noti che i requisiti possono essere distinti in funzionali (ossia quali sono le funzioni e funzionalità offerte dal sistema) e non funzionali (ossia quali caratteristiche deve possedere il sistema).

2.0.1 Requisiti funzionali

Il primo requisito funzionale emerso è il seguente:

1. suddivisione degli utenti del sistema in amministratori (proprietari e dipendenti di cinema) e clienti, ciascuno con differenti operazioni possibili e differenti viste degli stessi dati.

Tale requisito è di importanza rilevante in quanto consente di distinguere i successivi requisiti in base alla tipologia di utente che utilizzerà il sistema.

In particolare, tra le funzionalità offerte dal sistema agli amministratori è bene evidenziare:

2. login
3. possibilità di aggiungere/rimuovere/modificare le sale presenti, specificando il numero di posti disponibili in ogni sala
4. possibilità di aggiungere/rimuovere/modificare proiezioni di film, indicando giorno, orario, film proposto e prezzo del biglietto
5. possibilità di visualizzare feedback e recensioni pubblicate dai clienti.

Per quanto invece concerne le funzionalità offerte ai clienti, esse sono:

6. login

7. possibilità di consultare le proiezioni di film disponibili
8. possiblità di prenotare uno o più posti in una proiezione, vedendo in tempo reale quali posti sono liberi e quali sono occupati
9. possibilità di lasciare dei feedback/recensioni per le proiezioni a cui ha partecipato
10. possibilità di esprimere delle preferenze circa i generi di film preferiti.

Infine, un requisito funzionale che invece riguarda indistintamente sia gli amministratori che i clienti è il seguente:

11. il sistema deve prevedere un meccanismo di notifiche che, in tempo reale, avverta gli amministratori appena una recensione viene pubblicata e avverta i clienti appena viene pubblicato un film di loro interesse (in base ai generi di film che hanno espresso come di loro interesse) o in caso di modifiche ad una proiezione per la quale avevano fatto una prenotazione.

2.0.2 Requisiti non funzionali

Le caratteristiche e proprietà rilevanti che il sistema deve possedere sono le seguenti:

12. accessibilità → il sistema deve rispettare tutte le norme e best practices inerenti l'accessibilità che ci sono state presentate durante il corso
13. usabilità
14. sostenibilità → il sistema deve prevedere, ove possibile, l'adozione di piccoli accorgimenti consigliati durante il seminario sullo sviluppo sostenibile del web
15. reattività
16. compatibilità → l'applicazione web dev'essere fruibile in maniera accettabile sui più diffusi browser e indipendentemente dal dispositivo usato (sia smartphone che computer)
17. integrazione → il sistema deve prevedere integrazione con servizi terzi tramite API esterne
18. sicurezza → la registrazione ed il login devono prevedere una qualche forma di crittografia delle password
19. deployment realistico → l'applicazione web dev'essere deployata su un server vero e proprio, in modo che possa essere raggiunta dagli utenti in qualunque momento e da qualsiasi luogo.

Capitolo 3

DESIGN

3.1 Personas e scenarios

Preliminarmente alla fase di design, ma in realtà anche precedentemente alla fase di individuazione dei requisiti, sono state utilizzate le personas e gli scenarios per anticipare alcuni requisiti e alcuni aspetti di design rilevanti.

Di seguito si riportano le personas invididuate:

- Alberto è un ragazzo di 17 anni che frequenta il Liceo Classico. Ha svariati hobby come il calcio e il pianoforte. Tra i suoi hobby non c'è il cinema, tuttavia frequenta sporadicamente le sale della sua città con i suoi amici per godersi qualche nuova uscita di suo interesse. Non avendo ancora la patente, egli è stanco di dover ogni volta chiedere un passaggio a qualcuno solo per arrivare fino al cinema della sua città per capire se è uscito un film di suo interesse, pertanto gradirebbe che un sito web gli notificasse direttamente le nuove uscite che il cinema programma di proiettare.
- Barbara è una studentessa universitaria del corso DAMS. È appassionata di tutto ciò che riguarda l'arte, la musica, ma soprattutto è una grandissima appassionata di cinema. Un giorno sogna di diventare una sceneggiatrice, ossia di occuparsi della scrittura della trama e dei dialoghi dei film. Barbara sa che per raggiungere il suo sogno occorre impegnarsi tanto, infatti non si perde nemmeno un film tra quelli proiettati nel suo cinema di fiducia, non solo per svagare e divertirsi, ma per concentrarsi e studiare i film che guarda. Dopo aver guardato ogni film le piace approfondirlo facendo una moltitudine di ricerche sul web per vedere le sue caratteristiche e peculiarità, come durata, trama e genere. Per lei è scomodo però dover aprire svariati siti per trovare le informazioni che cerca (e spesso scoprire tra l'altro che non sono attendibili). Per agevolare il suo studio dei film le piacerebbe che fosse direttamente il portale del cinema a riportare le informazioni che cerca, così da poterle avere tutte insieme appena decide il prossimo film da guardare.

- Carlo è un impiegato di mezza età non molto avvezzo alla tecnologia. Si vanta di essere ancora uno dei pochi italiani rimasti a leggere il giornale su carta e si scandalizza per ogni articolo legato all'inflazione che legge. Carlo è infatti un impiegato bancario, pertanto ha solide basi di finanza ed economia. Provando un certo astio per tutto ciò che è tecnologico, non ci tiene ad usare un sito o un'app per prenotare il suo posto al cinema tuttavia, come il suo background culturale suggerisce, è amante del risparmio e delle ottimizzazioni di tempo e sforzi, pertanto apprezzerebbe se nel sito dei cinema riportassero chiaramente il prezzo del biglietto, in modo da poter fare agevolmente dei confronti e capire senza troppi sforzi in quale cinema gli conviene andare. Ci tiene a precisare che nel caso il biglietto, però, lo comprerebbe dal vivo, parlando con la cassiera e pagando rigorosamente in contanti.

- Daniela è una professoressa di storia delle scuole superiori, nonché responsabile del cineforum scolastico. Sta organizzando l'annuale uscita scolastica al cinema, nella quale ogni classe del suo istituto va a vedere un film educativo ed istruttivo. Per quest'anno è stato scelto come film "The Imitation Game" tuttavia, per organizzare al meglio le uscite, Daniela vorrebbe sapere quali proiezioni hanno sufficienti posti liberi per ospitare una o più classi e quali no. Siccome la tecnologia lo permette, si chiede perché debba continuare a organizzare dal vivo, perdendo il suo tempo, questa uscita, quando dalla comodità di casa potrebbe monitorare lo stato delle proiezioni in calendario, individuare quelle adatte ad ospitare le sue classi e prenotare.

- Emiliano è un anziano pensionato. Non va al cinema siccome è parecchio scontroso e odia quando qualcuno gli dice di fare silenzio. Suo malgrado, per fare contenta la sua nipotina di 5 anni, è andato a vedere l'ultimo film con i minions. L'esperienza non è stata per nulla buona: mangiando un popcorn gli si è scheggiata la dentiera, la luminosità dello schermo era talmente eccessiva che non ha potuto godersi il film e come se non bastasse usciti dal cinema ha litigato con il proprietario perché è stato ripetutamente richiamato al silenzio e a non disturbare durante la proiezione. Adesso Emiliano è talmente furibondo che vorrebbe imparare ad usare uno smartphone per lasciare una recensione molto negativa all'esperienza vissuta.

- Francesca è una giovane mamma di 2 gemelli di 4 anni. È recentemente andata al cinema della sua città con i suoi due figli, ai quali il nuovo film della Pixar è veramente molto piaciuto. Francesca vorrebbe dunque poter lasciare una recensione per dire che il film è molto piaciuto ai suoi figli e che è adatto ad un pubblico giovane.

- Giulio è il proprietario del cinema Eliseo di Cesena. Un lavoro ripetitivo di cui si occupa e che vorrebbe evitare è fare pubblicità sui social non appena decide di calendarizzare una nuova proiezione. È veramente stressante e

dispendioso, dice, dover pagare un SMM per creare le locandine delle nuove proiezioni e pubblicarle su tutti i social. Anche in ottica di risparmio, gradirebbe che nel sito del suo cinema ci fosse un meccanismo che gli consenta di calendarizzare le prossime proiezioni e contemporaneamente avvisare gli utenti che un nuovo film sarà proiettato.

- Holly è la proprietaria della multisala "UCI Cinema" di Savignano sul Rubicone. Trattandosi di una multisala non è raro aggiungere/rimuovere sale o aggiungere/rimuovere i posti che compongono una sala. Il sito che attualmente usa per gestire le prenotazioni online dei biglietti è però statico, ossia non consente nessun cambiamento per le sale, cosa che invece sarebbe molto comoda a Holly, anche perché sono sempre più frequenti le inconsistenze dovute alle prenotazioni online (accade spesso che qualcuno prenoti un posto non più esistente a causa di modifiche alla disposizione dei sedili nella sala).

- Ignazio è il proprietario del cinema Cineflash di Forlimpopoli. La sua attività, dopo la pandemia, è molto in difficoltà: non riesce ad attrarre nuovi clienti, è sull'orlo della chiusura e non riesce a capire come mai visto che la concorrenza, pur avendo i medesimi prezzi e servizi, non è nella sua stessa situazione. Prima che sia troppo tardi, intende investigare le motivazioni dietro alle sue perdite pertanto, oltre a chiedere di persona pareri ai clienti a seguito delle visioni, vorrebbe anche aggiungere un'apposita sezione sul sito del cinema in cui i clienti possono scrivere delle recensioni/feedback sulle proiezioni a cui hanno assistito e lui vorrebbe venire notificato immediatamente appena una nuova recensione viene pubblicata, siccome il tempo stringe ed ogni secondo è prezioso per evitare la chiusura.

Infine, di seguito si riportano gli scenarios ipotizzati:

- Alberto vuole vedere quali film propone il cinema e venire notificato se viene aggiunto al calendario un film di suo interesse. Per fare ciò apre il sito. Subito gli compare la schermata di registrazione. Inserisce tutti i dati richiesti per creare un nuovo account, tra i quali anche quali sono le categorie di film di suo gradimento e su cui quindi desidera essere notificato. A questo punto accede alla homepage del sito e subito si trova davanti a tutti i film che il cinema proietterà nei prossimi giorni, in modo da poter vedere se c'è qualcosa di suo interesse. Tramite il menù in alto, Alberto decide poi di dare un'occhiata alla sezione dedicata alle sue notifiche, dove può controllare le ultime notifiche relative ai film che potrebbero piacergli e che sono stati calendarizzati. Se poi trova un film di suo interesse può, cliccandoci sopra, essere redirezionato alla pagina dedicata al film, dove è possibile vederne tutte le caratteristiche ed eventualmente prenotare un biglietto.

- Barbara, essendo appassionata di cinema, non solo vorrà agevolmente prenotare dei biglietti, ma anche navigare i film presenti per studiarli, guardando trama, durata, voto, genere e tutte le informazioni che ritiene rile-

vanti. Dopo essersi registrata, anche lei può navigare nell'homepage del sito per vedere tutti i film in proiezione nei prossimi giorni. La ricerca è agevolata non solo da filtri, ma anche da una barra di ricerca situata in alto nella schermata, che permette di ricercare un film per nome. Cliccando poi su un film, Barbara apre la pagina dedicata, dove può trovare tutte le caratteristiche rilevanti. Volendo poi procedere con l'acquisto del biglietto, Barbara preme l'apposito pulsante, che la redirezione ad un'altra schermata del sito nella quale può scegliere un giorno e un orario in cui la proiezione si farà, potendo vedere anche lo stato della sala fino a quale momento (posti liberi e posti occupati). Cliccando su un posto Barbara lo sceglie, in modo da prenotarlo, e procede al pagamento.

- A Carlo interessa solo vedere i prezzi dei biglietti per effettuare confronti con altri cinema ma, essendo Carlo piuttosto contrario alla tecnologia, egli preferisce non registrarsi al sito per non diffondere i suoi dati, pertanto entra come ospite, senza un account, e si ritrova direttamente nella homepage del sito, da qui chiaramente non ha la possibilità di fare tutte le operazioni che farebbe un utente registrato (come lasciare recensioni o comprare biglietti), tuttavia può comunque aprire la scheda di un film di interesse per vederne il prezzo.

- Daniela ha la necessità di prenotare un numero elevato di biglietti, pertanto si registra al sito, nella homepage seleziona il film "The imitation game" cercandolo grazie all'apposita barra di ricerca posta in alto, apre la scheda dedicata e si dirige nella sezione di acquisto. Qui Daniela naviga un po' le varie giornate per vedere in quale ci sono più posti disponibili. Individuata la giornata adatta seleziona 20 posti, uno per ogni suo studente, e compra i biglietti per la giornata del cineforum scolastico.

- Emiliano ha l'impellente necessità di lasciare una cattiva recensione ed è talmente arrabbiato che intende addirittura iscriversi al sito pur di poterlo fare. Pertanto si connette al sito, si registra inserendo i suoi dati ed entra nella homepage. Subito nota che può accedere nella sua area personale, dove sono riportati i film che ha visto. Dall'elenco seleziona il film legato alla sua pessima esperienza e già dall'elenco compare il suo agognato obiettivo, il bottone "Scrivi recensione". Premendolo, Emiliano viene riportato nella pagina dedicata al film, ove può compilare la sua recensione. Carlo lascia un voto discreto al film, ma pessimo all'esperienza e nell'area di testo dedicata scrive i suoi pareri e i motivi che lo hanno spinto a dare quel giudizio. Terminata la sua recensione preme il bottone per pubblicare la recensione.

- Francesca ha, come nel caso precedente, la necessità di scrivere una recensione, tuttavia positiva. Esegue gli stessi passi di Emiliano, tuttavia non nota che è possibile scrivere la recensione dalla sua area personale, pertanto raggiunge la scheda dedicata del film tramite la barra di ricerca posta in alto. La restante interazione è del tutto analoga al caso sopra.

- Giulio, essendo il proprietario del cinema, deve semplicemente aggiungere un nuovo film al calendario delle proiezioni cosicché i clienti sappiano che è in programma e chi ne ha fatto richiesta venga notificato. Giulio non ha la necessità di registrarsi per entrare sul sito, essendo admin del cinema ha un ingresso dedicato in cui deve specificare solo le sue credenziali. Nella sua homepage Giulio trova subito l'elenco dei film che sono in programma, da cui può aggiungere un nuovo film e specificarne tutte le caratteristiche: caratteristiche del film, prezzo, orari, sale. Confermando la nuova programmazione, Giulio viene avvisato che l'operazione è andata a buon fine e che le notifiche sono state spedite ai clienti che potrebbero essere interessati al film appena inserito dall'admin.

- Holly, anch'essa proprietaria di cinema, ha la necessità di cambiare la disposizione dei posti di una sua sala. Ipotizzando che sia già iscritta, nella homepage si direziona nella sezione dedicata alle sue sale, dove ha un riepilogo delle sale che compongono il suo cinema (per ognuna un identificativo, il numero di posti, numero di righe e di colonne). È possibile modificare i parametri di una sala dall'apposito bottone situato di fianco all'identificativo. Premendolo Holly specifica ad esempio che la sua sala ha una fila in più composta da 10 nuovi posti. Chiaramente Holly non può cambiare la disposizione di una sala per la quale ci sono già delle prenotazioni attive. Comunque, salvando la modifica, i clienti vedranno la nuova sala con la disposizione aggiornata quando vorranno prenotare i biglietti.

- Ignazio vuole semplicemente essere notificato non appena qualcuno lascia una recensione. L'interazione è del tutto analoga ai due casi precedenti, tuttavia Ignazio appena entra nella sua homepage viene avvertito che qualcuno ha lasciato una recensione. Allora Ignazio apre la sezione dedicata alla recensioni e consulta le ultime recensioni non lette, che poi verranno eliminate dalla sezione notifiche, in modo da non venire ripresentate alla successiva apertura. Ignazio può comunque rileggere una recensione nella sezione delle notifiche già lette o aprendo la scheda dedicata ad un film.

Come detto, le personas e gli scenarios sono stati riportati in quanto ci hanno permesso di anticipare, rilevare e formalizzare i principali requisiti (sia funzionali che non) ed aspetti di design.

3.1.1 Focus group

Durante la fase di design è stata anche organizzata una sessione di focus group, alla quale hanno partecipato tre sviluppatori (noi) e due nostri amici in veste di utenti finali.
Come suggerito, il focus group è durato poco, meno di un'ora, e in questa sessione sono emersi, oltre ad ulteriori funzionalità (già inserite nell'elenco dei requisiti funzionali), anche alcuni importanti aspetti di design dell'architettura dell'applicazione web e della sua interfaccia.
In particolare noi sviluppatori avevamo due dubbi principali, che abbiamo risolto

chiedendo ai due utenti finali cosa avrebbero preferito.
In particolare è emersa:

- una preferenza circa le notifiche non bloccanti (effetto luminoso che segnala la notifica) rispetto a quelle bloccanti (degli alert che interrompono bruscamente il flusso di navigazione)
- una preferenza di stile "film-centrico" (nella homepage è meglio vedere subito i film, come accade anche in Netflix, e dopo aver selezionato il film si scelgono le operazioni) rispetto a uno stile "operazione-centrico" (nella homepage si sceglie l'operazione e dopo si seleziona il film interessato).

3.1.2 Design dell'architettura

Il design dell'architettura proposto si è concentrato sui seguenti aspetti:

- stack architetturale MEVN → requisito richiesto dalla professoressa che richiede un full-stack Javascript composto da MongoDB, Exress.js, Angular e Node.js
- Single Page Application → la nostra applicazione web è progettata per caricare una sola pagina HTML iniziale e aggiorna dinamicamente il suo contenuto senza ricaricare l'intera pagina ma solo i componenti necessari, offrendo un'esperienza utente più fluida, veloce e reattiva
- security by design → dovrà essere garantita tramite suddivisione role-based dei permessi tra amministratori e clienti e crittografia delle password con funzioni hash e sale
- sustainabilty by design → dovranno essere tenuti in considerazione alcuni aspetti emersi durante il seminario sullo sviluppo sostenibile del web, ossia immagini in formato .webp (più sostenibile in quanto compressione più efficiente) e test dei punteggi di sostenibilità.

3.1.3 Design delle interfacce

Invece, per quanto concerne il design delle interfacce utente, esso si è concentrato sui seguenti aspetti:

- accesibility by design → progettazione accessibile di tabelle, immagini (tramite descrizione alternativa), modali (tramite role, aria-labelledBy e aria-modal) e colori (conformi a standard)
- mobile first → l'applicazione web è progettata e sviluppata dando la precedenza ai dispositivi mobile e solo successivamente ai dispositivi con schermo landscape
- layout responsive → il layout si deve adattare al dispositivo su cui si sta utilizzando l'applicazione.

Come suggerito dalle best practices la fase di design delle interfacce va accompagnata dalla realizzazione dei mockup.
E' possibile vedere tutti i mockup realizzati al seguente link.

Di seguito riportiamo alcuni mockup rilevanti, sia nella versione web che mobile:

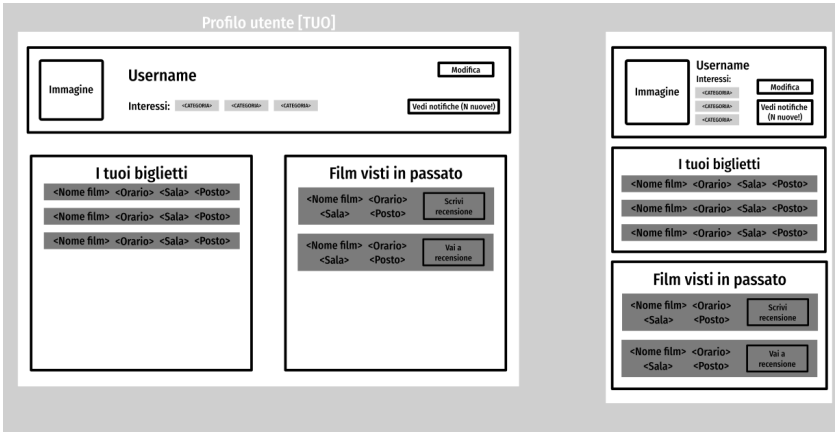

Figura 3.1: mockup del profilo dell'utente non amministratore

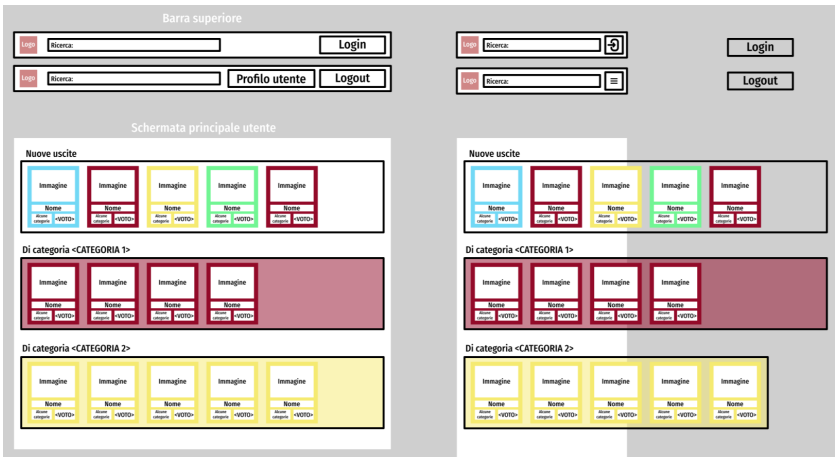

Figura 3.2: mockup della homepage di un utente non amministratore.

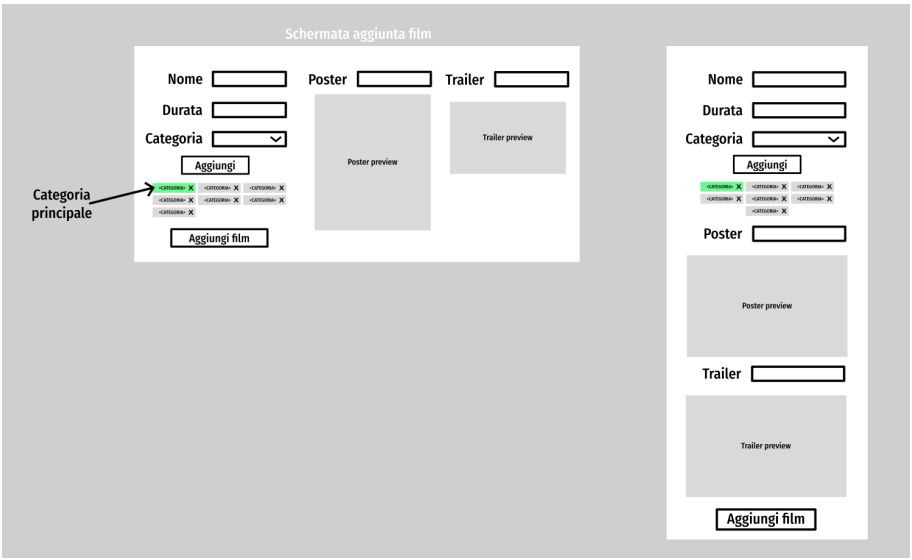

Figura 3.3: mockup per l'inserimento di un film.

Capitolo 4

TECNOLOGIE

Di seguito si riporta l'elenco delle tecnologie impiegate nel progetto e le motivazioni che hanno portato al loro utilizzo:

- tutte le tecnologie previste dallo stack architetturale MEVN (MongoDB, Express.js, Vue.js e Node.js) → compongono un'architettura full-stack moderna, flessibile e scalabile, che consente di sviluppare applicazioni web in modo rapido ed efficiente, utilizzando il linguaggio Javascript sia su front-end e back-end
- linguaggio Typescript → utile perchè aggiunge tipizzazione statica a JavaScript, migliorandone leggibilità, manutenzione del codice e riduzione degli errori durante lo sviluppo di applicazioni complesse
- framework Tailwind CSS → è utile perché rappresenta un'evoluzione di Bootstrap che permette di creare interfacce utente mobile first e altamente personalizzabili in modo rapido, sfruttando classi utility predefinite che eliminano la necessità di scrivere CSS personalizzato per ogni pagina
- libreria Mongoose → semplifica l'interazione con MongoDB fornendo un'interfaccia intuitiva per definire schemi, validare dati e gestire operazioni sul database in modo strutturato ma al contempo efficiente
- libreria Socket.io → libreria che agevola la comunicazione a bassa latenza, bidirezionale, e ad eventi tra client e server
- Axios → libreria che semplifica le richieste HTTP con una sintassi chiara e funzionalità avanzate
- Pinia → usata perchè fornisce una gestione dello stato degli utenti semplice, reattiva e scalabile per le applicazioni Vue.js, semplificando la condivisione e la gestione dei dati tra i componenti
- CryptoJS → libreria per implementare algoritmi di crittografia, particolarmente utile per cifrare e decifrare le password degli utenti.

Capitolo 5

CODICE

Nel presente capitolo sono riportati frammenti rilevanti di codice.

5.1 Notifiche

Un requisito obbligatorio del progetto prevedeva l'implementazione di un qualche tipo di notifiche in tempo reale.

Nel nostro caso, abbiamo identificato i seguenti eventi associati a notifiche:

- quando un cliente pubblica una recensione occorre notificare tutti gli amministratori
- quando un amministratore aggiunge un film occorre notificare tutti i clienti che hanno tra i loro generi preferiti quello del film appena inserito
- quando un amministratore elimina o modifica una proiezione occorre notificare tutti i clienti che avevano prenotato dei posti per quella proiezione.

Di seguito si riporta l'esempio dell'emissione della notifica riferita all'aggiunta di un nuovo film (le altre due sono del tutto analoghe):

```
1  // AddMovieMoald.vue
2  <script setup lang="ts">
3  ...
4  async function addMovie() {
5      try {
6          const response = await axios.post('http://
               localhost:3001/movies', {
7              title: name.value,
8              productionYear: year.value,
9              trailerLink: extractVideoCode(trailer.value),
10             duration: duration.value.toString(),
11             genres: selectedGenres.value,
12             poster: image.value,
13         });
```

```
14
15          if (response.status === 200) {
16              user.socket.emit('newFilm', { movie: name.value,
                    genres: selectedGenres.value });
17              emit('close', true);
18          } else {
19              msgUser.value = "Errore durante l'aggiunta del
                    film.";
20              check.value = false;
21          }
22      } catch (error) {
23          console.error('Errore di connessione:', error);
24          msgUser.value = "Errore di connessione durante l'
                aggiunta del film.";
25          check.value = false;
26      }
27  }
28  </script>
```

Le notifiche sono state implementate sfruttando la libreria Socket.io e in particolare il comportamento a seguito di una ricezione è consultabile nel file index.js:

- definizione delle strutture dati → adminSockets è una mappa che associa un ID di amministratore a un'istanza di socket associata alla connessione e similmente userSockets è una mappa che associa un ID di utente a un'istanza di socket associata alla connessione
- definizione degli eventi su Socket.io → in caso di evento registerAdmin, un amministratore invia l'ID al server per registrarsi ed il nuovo socket viene salvato nella mappa adminSockets, invece in caso di evento registerUser si agisce similmente a registerAdmin, ma registra l'utente nella mappa userSockets
- gestione delle notifiche → quando viene ricevuta una nuova recensione (reviewData) tutti gli amministratori connessi vengono notificati con un evento newReviewNotification e il relativo contenuto, quando invece viene aggiunto un nuovo film vengono estratti i suoi generi e per ciascuno il server ottiene l'elenco degli ID utente interessati a quel genere, in modo poi da notificarli con un evento newFilmNotification, infine quando c'è un'eliminazione o cambiamento di proiezione (screening) il server ottiene l'elenco degli ID utente che hanno prenotato quella proiezione, in modo che poi essi vengono notificati con un evento newScreeningNotification
- gestione della disconnessione → quando un socket si disconnette il server tenta di rimuovere l'utente dalla mappa appropriata
- avvio del server sulla porta 3001.

```
1  // index.js
2  /**
```

```
3   * @type {Map<String, Socket<DefaultEventsMap, DefaultEventsMap
        , DefaultEventsMap, any>>}
4   */
5   let adminSockets = new Map();
6   /**
7   * @type {Map<String, Socket<DefaultEventsMap, DefaultEventsMap
        , DefaultEventsMap, any>>}
8   */
9   let userSockets = new Map();
10  io.on('connection', (socket) => {
11      // Riconosci gli amministratori
12      socket.on('registerAdmin', (id) => {
13          if (adminSockets.has(id)) {
14              const oldSocket = adminSockets.get(id);
15              if (oldSocket.connected) {
16                  oldSocket.disconnect();
17              }
18              adminSockets.delete(id);
19          }
20          adminSockets.set(id, socket);
21      });
22
23      socket.on('registerUser', (id) => {
24          if (userSockets.has(id)) {
25              const oldSocket = userSockets.get(id);
26              if (oldSocket.connected) {
27                  oldSocket.disconnect();
28              }
29              userSockets.delete(id);
30          }
31          userSockets.set(id, socket);
32      });
33
34      socket.on('newReview', (reviewData) => {
35          adminSockets.forEach((adminSocket, adminId) => {
36              if (adminSocket.connected) {
37                  try {
38                      adminSocket.emit('newReviewNotification',
                            reviewData);
39                  } catch (error) {
40                      console.error(`Errore nello invio della
                            notifica a admin ID: ${adminId}`, error
                            );
41                  }
42              } else {
43                  console.warn(`Socket non connesso per admin ID:
                         ${adminId}`);
44              }
45          });
46      });
47
48      socket.on('newFilm', async (film) => {
49          genres = film.genres;
50          const userPromises = genres.map(async (genreId) => {
51              try {
52                  const response = await axios.get(`http://
                        localhost:3001/users/genre/${genreId._id}`)
```

```
                ;
53              const userIds = response.data.map(user =>
                    user._id);
54              return userIds;
55          } catch (error) {
56              console.error(`Errore durante la chiamata per
                    il genere ${genreId._id}:`, error);
57              return []; // Restituisci un array vuoto in
                    caso di errore
58          }
59      });
60
61      try {
62          const usersByGenre = await Promise.all(userPromises
                );
63          const allUsers = [...new Set(usersByGenre.flat())];
64          userSockets.forEach((userSocket, userId) => {
65              if (userSocket.connected) {
66                  if (allUsers.includes(userId)) {
67                      try {
68                          userSocket.emit('
                                newFilmNotification', film);
69                      } catch (error) {
70                          console.error(`Errore nello invio
                                della notifica a user ID: ${
                                userId}`, error);
71                      }
72                  }
73              } else {
74                  console.warn(`Socket non connesso per user
                        ID: ${userId}`);
75              }
76          });
77      } catch (error) {
78          console.error('Errore durante l elaborazione delle
                notifiche:', error);
79      }
80  });
81
82  socket.on('changeScreening', async (screening) => {
83      try {
84          const response = await axios.get(`http://
                localhost:3001/reservations/screening/${
                screening.screening}`);
85          const userIds = response.data.map(data => data.user
                );
86          userSockets.forEach((userSocket, userId) => {
87              if (userSocket.connected) {
88                  if (userIds.includes(userId)) {
89                      try {
90                          userSocket.emit('
                                newScreeningNotification',
                                screening.screening);
91                      } catch (error) {
92                          console.error(`Errore nello invio
                                della notifica a user ID: ${
                                userId}`, error);
```

```
                        }
                    }
                } else {
                    console.warn(`Socket non connesso per user
                        ID: ${userId}`);
                }
            });
        } catch (error) {
            console.error(`Errore durante la chiamata:`, error)
                ;
            return []; // Restituisci un array vuoto in caso di
                errore
        }

    });

    // Gestione della disconnessione
    socket.on('disconnect', () => {
        if (isAdmin && adminSockets.delete(socket)) {
            console.log('Un utente admin si e disconnesso');
        } else if (userSockets.delete(userId)) {
            console.log('Un utente si e disconnesso');
        }
    });
});

server.listen(3001, () => {
    console.log('Server listening on port 3001');
});
```

Nel codice seguente viene specificato come viene modificato il componente PageHeader in caso di noifica:

```
// PageHeader.vue
<script setup lang="ts">
const hasNewNotification = ref(false);

onMounted(() => {
    user.socket.on('newReviewNotification', () => {
        hasNewNotification.value = true;
        setTimeout(() => {
            hasNewNotification.value = false;
        }, 60000);
    });
    user.socket.on('newFilmNotification', () => {
        ...
    });
    user.socket.on('newScreeningNotification', () => {
        ...
    });
});

function goToNotify() {
    hasNewNotification.value = false; // Resetta lo stato
    router.push('/notify');
}
...
</script>
```

```
26
27  <template>
28      ...
29      <SimpleButton color="primary" :title="expanded ? 'Chiudi
            menu azioni' : 'Apri menu azioni'"
30          rounding="full" :handle-click="toggleExpandedMenu"
                size="small" :bold="true"
31          class="aspect-square !transition-all" :class="{ '
                -rotate-90 m-1': expanded }">
32    <EllipsisVerticalIcon v-if="!hasNewNotification" class="py-0"
          />
33    <BellIcon v-else class="py-0" />
34  </SimpleButton>
35  </template>
```

5.2 Prenotazioni

Il frammento di codice sotto riportato è indicativo di quanta attenzione sia stata messa anche nell'accessibilità di componenti complessi.
In particolare, il componente MovieRoom.vue utilizza un template HTML dinamico per generare una griglia di pulsanti che rappresentano i posti disponibili in una sala.
Ogni posto è rappresentato da un pulsante interattivo disposto in righe e colonne.

```
1   // MovieRoom.vue
2   <template>
3       <div class="...">
4           <div class="..."
5               role="grid" aria-label="Tabella dei posti">
6               <div v-for="row in Array.from(Array(rows).keys())"
                    :key="row"
7                   class="..." role="row">
8                   <button v-for="col in Array.from(Array(cols).
                        keys())" :key="`${row}-${col}`"
9                       :disabled="!interactive || isSpotOccupied(
                            row, col)" @click="toggleSelection(
                            Number(row), col)"
10                      class="..."
11                      :class="[colors(selectedSpotsIds.has(`${row
                            }-${col}`), isSpotOccupied(row, col))"
12                      :title="'Riga ' + Number(row + 1) + ',
                            colonna ' + (col + 1)" role="gridcell"
13                      :aria-disabled="!interactive ||
                            isSpotOccupied(row, col)"
14                      :aria-selected="selectedSpotsIds.has(`${row
                            }-${col}`)"
15                      :aria-label="'Riga ' + Number(row + 1) + ',
                             colonna ' + (col + 1)">
16                      {{ col + 1 }}
17                  </button>
18              </div>
19          </div>
20      </div>
```

```
21 </template>
```

Come si nota, il codice include diverse pratiche per garantire che la griglia sia accessibile anche per utenti con disabilità o limitazioni:

- uso dei ruoli ARIA e degli attributi dinamici aria-disabled, aria-select ed aria-label
- titoli e descrizioni per ogni pulsante
- la disabilitazione è segnalata sia visivamente (:disabled) sia semanticamente (aria-disabled).

5.3 Database

Per quanto concerne l'organizzazione del database con MongoDB sono state seguite le best practice suggerite a lezione, come ad esempio la suddivisione tra schema, controller e router, in cui:

- model definisce la struttura e le regole per i dati degli utenti nel database utilizzando Mongoose
- controller contiene la logica applicativa per gestire le richieste HTTP, come creare, leggere, aggiornare o eliminare utenti, separando il codice del server dalla gestione dei dati
- routes specifica invece le URL endpoint e le associa ai metodi appropriati del controller, in modo da instradare le richieste degli utenti verso la logica applicativa corretta.

Di seguito si riportano gli esempi per gli utenti, ma la struttura è del tutto analoga per ogni concetto presente nel database:

```
1  // usersModel.js
2  const mongoose = require('mongoose');
3
4  const userSchema = new mongoose.Schema({
5      name: String,
6      surname: String,
7      birthdate: Date,
8      email: { type: String, unique: true },
9      password: String,
10     salt: String,
11     isAdmin: Boolean,
12     favoriteGenres: {
13         type: [{ type: mongoose.Schema.Types.ObjectId, ref: '
               Genres' }],
14         validate: {
15             validator: function (array) {
16                 return array.length <= 5;
17             },
18             message: 'Un utente puo avere al massimo 5 generi
                   preferiti'
```

```
        }
    },
    profilePicture: { type: String, default: 'profile.webp' }
});

const usersModel = mongoose.model('User', userSchema);

module.exports = { usersModel };

// usersController.js
const { usersModel } = require('../models/usersModel');
const crypto = require('crypto');

...

exports.getUserByID = (req, res) => {
    usersModel.findById(req.params.id)
        .populate('favoriteGenres', 'name')
        .then(doc => {
            if (!doc) {
                return res.status(404).send('User not found');
            }
            doc.password = undefined;
            res.json(doc);
        })
        .catch(err => {
            res.status(500).send(err);
        });
}

exports.createUser = async (req, res) => {
    req.body.isAdmin = false;
    req.body.salt = crypto.randomBytes(32).toString('hex');
    function isFutureDate(dateToCheck) {
        const today = new Date();
        const inputDate = new Date(dateToCheck);
        return inputDate.setHours(0, 0, 0, 0) > today.setHours
            (0, 0, 0, 0);
    }

    if (isFutureDate(req.body.birthdate)) {
        return res.status(204).send('La data di nasciata non
            puo essere futura');
    }

    const newUser = new usersModel(req.body);
    newUser.save()
        .then(doc => {
            res.json(doc);
        })
        .catch(err => {
            res.status(500).send(err);
        });
}

exports.updateUser = (req, res) => {
    ...
```

```
74    };
75
76  exports.deleteUser = (req, res) => {
77      usersModel.findByIdAndDelete(req.params.id)
78          .then(doc => {
79              if (!doc) {
80                  return res.status(404).send('User not found');
81              }
82              res.json({ message: 'User deleted' });
83          })
84          .catch(err => {
85              res.status(500).send(err);
86          });
87  }
88
89  exports.findAdmins = (req, res) => {
90      ...
91  }
92
93  exports.findEmail = (req, res) => {
94      ...
95  }
96
97  exports.authenticateUser = (req, res) => {
98      usersModel.find()
99          .where('email').equals(req.query.email)
100         .then(docs => {
101             const user = docs[0];
102             if (req.body.password === user.password) {
103                 res.json(user._id);
104             } else {
105                 res.status(401).send('Password incorretta');
106             }
107         })
108         .catch(err => {
109             res.status(500).send(err);
110         });
111 }
112
113 exports.authenticateUserById = (req, res) => {
114     ...
115 }
116
117 exports.getUserInterests = (req, res) => {
118     usersModel.findById(req.params.id)
119         .populate('favoriteGenres', 'name')
120         .then(user => {
121             if (!user) {
122                 return res.status(404).send('User not found');
123             }
124             if (user.isAdmin) {
125                 return res.status(403).send('Admins do not have
                        favorite genres');
126             }
127             res.json(user.favoriteGenres);
128         })
129         .catch(err => {
```

```
            res.status(500).send(err);
        });
};

exports.findUsersByGenre = (req, res) => {
    ...
};


// usersRoutes.js
const express = require('express');
const router = express.Router();
const controller = require('../controllers/usersController');

router.route('/')
    .get(controller.usersList)
    .post(controller.createUser);

router.route('/email')
    .get(controller.findEmail)
    .post(controller.authenticateUser);

router.route('/admins')
    .get(controller.findAdmins);

router.route("/users/users/:id/interests")
    .get(controller.getUserInterests);

router.route('/genre/:genreId')
    .get(controller.findUsersByGenre);

router.route('/:id')
    .get(controller.getUserByID)
    .post(controller.authenticateUserById)
    .put(controller.updateUser)
    .delete(controller.deleteUser);

module.exports = router;
```

5.4 Esempio di API esterna

L'ultimo frammento di codice rilevante che riportiamo riguarda un esempio di integrazione con API esterne:

```
// MovieDetails.vue
<script setup lang="ts">
...
async function fetchMovieDetails(title: string, year: string) {
  try {
    const response = await axios.get(
      `http://www.omdbapi.com/?t=${encodeURIComponent(title)}&
          y=${year}&apikey=e2b4e2eb`
    );
    movieDetails.value = response.data;
  } catch (error) {
```

```
11        console.error("Errore nel recupero dei dettagli del film:",
              error);
12      } finally {
13        isLoading.value = false;
14      }
15    }
16    ...
17    </script>
```

Il frammento di codice sopra definisce una funzione asincrona, fetchMovieDetails, che utilizza Axios per richiedere i dettagli di un film specifico dall'API OMDB in base al titolo e all'anno forniti.

Si noti infine che, seppur non sia stata riportata, è prevista anche integrazione con le API del traduttore di Google per tradurre la trama restituita da OMDB dall'inglese all'italiano.

Capitolo 6

TEST

Per verificare la correttezza dell'applicazione web, sono stati effettuati diversi test e sono state sfruttate diverse tecniche di validazione del design delle interfacce.
In particolare le verifiche possono essere così suddivise:

- co-discovery learning per validazione del prototipo
- verifica del rispetto delle euristiche di Nielsen
- questionari su usabilità e user experience
- test di validazione dei campi
- test di sostenibilità.

Si noti dunque che sono state sfruttate tecniche di validazione e test non solo nella fase finale del progetto, ma anche durante tutto il suo sviluppo.

Vediamone i dettagli nelle sezioni che seguono.

6.1 Co-discovery learning

Per validare la correttezza e le interfacce di un prototipo della web application si è sfruttato approccio co-discovery learning con think aloud protocol, in cui sono stati coinvolti due utenti con background abbastanza differente (due studenti universitari, uno di matematica e uno di astrofisica), a cui sono stati presentati diversi task da svolgere e durante lo svolgimento sono stati richiesti commenti, sensazioni ed emozioni.

I commenti emersi sono riportati nella tabella seguente:

TASK PROPOSTO	DIFFICOLTA'	COMMENTI
Registrazione come cliente	Facile	Coerente con altri form tipici del web
Login come cliente	Facile	
Aggiungere dei generi di tuo interesse	Facile	Facile trovare la sezione, ma il bottone salva scompare dal mio schermo se metto troppi interessi
Vedere i dettagli del film Toy Story	Facile	Mi aspettavo che cliccando sul logo del sito mi rimandasse alla homepage
Lasciare una recensione al film Toy Story	Facile	Avrei preferito essere avvertito subito della scala dei voti fino a 5 e non mi era chiaro che la data non la dovessi mettere io ma fosse automatica
Cambiare la propria password, uscire dal sito e verificare il cambio password	Facile	Cliccato per sbaglio login
Prenotare un posto per il film Jurassic Park	Facile	Intuitivo, ma avrei preferito vedere chiaramente le lettere/numeri delle file della sala
Cambiare la propria foto profilo	Facile	
Login come admin	Facile	
Aggiungere una sala	Facile	Un po' di confusione con aggiunta/modifica
Leggere le proprie notifiche	Facile	
Aggiungere il film Cars	Facile	Come utente normale è poco intuitivo ma per un admin sarebbe comprensibile
Aggiungere una proiezione per il film Cars	Facile	Non avevo notato che dovevo specificare il cinema prima di salvare
Leggere tutte le recensioni del film Pulp Fiction	Facile	

Gli errori emersi sono stati corretti ed i suggerimenti emersi, ove ritenuti adottabili, sono stati implementati.

6.2 Euristiche di Nielsen

Uno dei membri del team di sviluppo, in qualità di valutatore esperto, ha certificato il rispetto delle euristiche di Nielsen.

6.3 Questionari di usabilità e user experience

In questa fase sono stati coinvolti 4 utenti esterni, a cui è stato chiesto di utilizzare in autonomia per del tempo il nostro sito e infine di compilare dei questionari.

Ecco i risultati ottenuti:

- User Experience Questionnaire →

Scale means per person					
Attrattività	Apprendibilità	Efficienca	Controllabilità	Stimolazione	Originalità
3,00	3,00	2,75	3,00	1,25	1,75
2,17	3,00	3,00	2,50	1,50	-0,25
1,00	1,50	1,25	1,75	0,25	-1,00
1,33	2,25	1,25	2,75	0,75	0,00

Figura 6.1: media per persona (range da -3 a +3).

UEQ Scales (Mean and Variance)		
Attrattività	1,875	0,80
Apprendibilità	2,438	0,52
Efficienca	2,063	0,89
Controllabilità	2,500	0,29
Stimolazione	0,938	0,31
Originalità	0,125	1,35

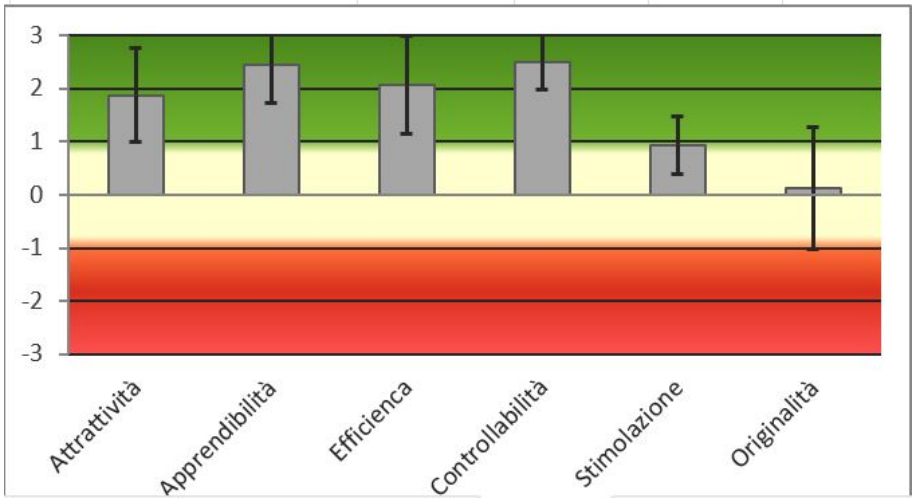

Figura 6.2: grafici per ambito.

- System Usability Scale → punteggi 75/100, 88/100, 100/100, 80/100.

6.4 Test di validazione dei campi

Sono stati effettuati approfonditi test per verificare la correttezza delle interfacce in risposta a input sia corretti che scorretti (date passate, date future, mail mal formattate, numeri al posto di stringhe e viceversa...).
Gli eventuali problemi emersi sono stati tutti risolti.

6.5 Test di sostenibilità

Come suggeritoci durante il seminario sullo sviluppo sostenibile del web, abbiamo utilizzato dei siti per scoprire il punteggio di sostenibilità del nostro sito:

Sebbene il punteggio ottenuto non sia ottimo e vi siano ampi margini di miglioramento, è emblematico il fatto che abbiamo raggiunto il medesimo punteggio ottenuto dal sito web dell'Università di Bologna.
Inoltre il sito non tiene in considerazione che il server su cui è raggiungibile il sito è alimentato prevalentemente da energia solare, il che fa aumentare il punteggio da 49 a 80, di gran lunga superiore alla media dei siti oggi disponibili.

Capitolo 7

DEPLOYMENT

L'applicazione web è accessibile in ogni momento e da ogni luogo siccome si trova su un server remoto di proprietà del team di sviluppo ed è raggiungibile al seguente link: https://abedo.ddns.info (in caso di problemi si prega di contattare gli sviluppatori via mail).

Il repository contenente il codice sorgente è consultabile al seguente link: https://github.com/AndreaBedei/cineweb.

Se si preferisce invece avviare una versione locale del sito occorre seguire i passi sotto riportati:

- aprire un terminale nella root del progetto
- spostarsi nella cartella frontend ed eseguire il comando npm install → spostarsi poi nella cartella backend e inserire lo stesso comando
- poi tornare nella cartella frontend ed eseguire il comando npm run dev
- infine aprire un altro terminale, spostarsi nella cartella backend ed eseguire il comando node index.js.

Se si vogliono provare le funzonalità offerte esclusivamente agli amministratori di sistema è possibile accedere utilizzando le seguenti credenziali: andreabedei@libero.it password.

Capitolo 8

CONCLUSIONI

Il progetto realizzato, stando ai feedback ottenuti, risulta piacevole, semplice ed intuitivo, unico aspetto negativo è la mancanza di originalità.

Per quanto riguarda i requisiti iniziali siamo riusciti a rispettarli tutti, in particolare:

- accessibility by design è stata raggiunta cercando di tenere in considerazione aspetti di accessibilità sin dalla fase di progettazione delle interfacce utente → da un punto di vista più implementativo è stato seguito uno sviluppo accessibile di tabelle, immagini (tramite descrizione alternativa), modali (tramite role, aria-labelledBy e aria-modal) e colori (conformi a standard), integrato da un approccio di sviluppo mobile first e responsive
- la reattività richiesta è stata ottenuta struttrando l'applicazione web come Single Page Application e sfruttando i benefici di Vue, che tra le altre cose consente di vedere gli aggiornamenti della pagina in tempo reale, senza dover per forza ricaricarla
- sustainability by design ottenuta facendo il deployment su un server alimentato prevalentemente ad energia solare ed utilizzando esclusivamente immagini in formato .webp
- security by design ottenuta adottando securty role-based (clienti e amministratori hanno viste differenti sugli stessi dati e possono fare operazioni differenti), il protocollo HTTPS e la crittografia delle password con funzioni hash e sale
- integrazione a servizi terzi tramite API esterne soddisfatta grazie alle comunicazioni con il servizio di OMDB (che ci fornisce i dettagli di ogni film richiesto), PayPal (che permette il pagamento tramite la loro piattaforma) e Google Traduttore per tradurre automaticamente in italiano le trame restituite da OMDB prima di mostarrle agli utenti.

Tra i limiti del nostro sito occorre invece citare:

- il fatto che come titolo del film vada messo per forza il titolo originale del film, altrimenti non si possono sfruttare le API messe a disposizione da OMDB
- assenza di statistiche utili per gli amministratori (ad esempio report su incassi giornalieri o periodici, capienza delle sale ecc)
- assenza di una qualche forma di machine learning, ad esempio come sistemi di recommendation o prediction.

Ad ogni modo il team di sviluppo si riserva di espandere il progetto e superare i limiti sopra riportati tramite sviluppi futuri.