

MovieParty  
Applicazioni e Servizi Web

Rei Beshiri - 950330  
[rei.beshiri@studio.unibo.it](mailto:rei.beshiri@studio.unibo.it)

Andrea Vaienti - 951454  
[andrea.vaienti2@studio.unibo.it](mailto:andrea.vaienti2@studio.unibo.it)

28 gennaio 2021

# Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
<b>2</b>	<b>Requisiti</b>	<b>3</b>
2.1	Requisiti di business . . . . .	3
2.2	Requisiti utente . . . . .	3
2.3	Requisiti funzionali . . . . .	4
2.4	Requisiti non funzionali . . . . .	6
2.5	Requisiti di implementazione . . . . .	6
<b>3</b>	<b>Design</b>	<b>8</b>
3.1	Design dell'architettura del sistema . . . . .	8
3.1.1	Componenti funzionali (React Hooks) . . . . .	8
3.1.2	JSX . . . . .	9
3.2	Metodologie di sviluppo . . . . .	9
3.3	Mockup iniziale . . . . .	10
3.4	Target User Analysis . . . . .	15
3.5	Storyboard . . . . .	17
3.6	Gamification . . . . .	34
<b>4</b>	<b>Tecnologie</b>	<b>36</b>
4.1	Redux . . . . .	36
4.2	Socket.io . . . . .	37
4.3	Axios . . . . .	38
4.4	Materialize . . . . .	39
<b>5</b>	<b>Codice</b>	<b>40</b>
5.1	Gestione stato con Redux . . . . .	40
5.2	Gestione comunicazione real-time con Socket.io . . . . .	41
<b>6</b>	<b>Test</b>	<b>43</b>
<b>7</b>	<b>Deployment</b>	<b>45</b>
7.1	Installazione . . . . .	45
7.2	Messa in funzione . . . . .	45
7.3	Conclusioni . . . . .	47

# Capitolo 1

## Introduzione

Il progetto **MovieParty** nasce dall'idea di realizzare un "salotto virtuale" in cui sia possibile divertirsi e rilassarsi, per rimanere in contatto con gli amici e non perdere le vecchie abitudini in questo periodo di lontananza forzata. MovieParty consiste in una Web App in cui è possibile visionare in compagnia o singolarmente i film presenti all'interno del catalogo offerto dal sito. La caratteristica principale che offre l'applicazione consiste nella creazione di una "stanza privata", denominata party, a cui posso essere invitati solamente gli amici. All'interno del party avverrà la riproduzione sincronizzata del video tra tutti i partecipanti. Chiunque potrà quindi fermare, avanzare o arretrare la riproduzione, in modo tale da riguardare i momenti salienti, avere tempo di preparare i pop-corn o saltare parti noiose. Ma non solo; il party è infatti provvisto di una chat per commentare in tempo reale ciò che si sta visionando, per poter condividere le proprie opinioni, pensieri o giudizi con tutti i partecipanti del party. MovieParty è presentata in stile Netflix, con una catalogazione dei prodotti in categorie. Sarà inoltre presente una barra di ricerca per cercare in modo facile e veloce contenuti multimediali non presenti all'interno delle categorie. L'applicazione dispone delle seguenti funzionalità:

- Sistema per la gestione sicura delle credenziali d'accesso. Solamente gli utenti autenticati potranno accedere ai servizi offerti dalla piattaforma.
- Sistema per la gestione della lista amici di ciascun utente.
- Sistema per la gestione di informazioni e servizi in real-time.
- Sistema di notifiche per la visualizzazione di richieste di amicizia o inviti al party.

# **Capitolo 2**

# **Requisiti**

Questo capitolo ha come scopo quello descrivere dettagliatamente tutti i requisiti del software implementato. La quasi totalità dei requisiti sono rimasti invariati sin dalle prime fasi del progetto, mentre alcuni sono stati leggermente modificati o eliminati. È bene precisare che qualunque requisito sottoelencato è stato selezionato in quanto verificabile.

## **2.1 Requisiti di business**

L'applicazione dovrà disporre delle seguenti caratteristiche:

- 1 Business.
  - 1.1 Possibilità di visionare contenuti multimediali presenti nella raccolta del sito in solitaria o in compagnia di utenti presenti nella lista amici.
  - 1.2 Chat in real-time durante la visualizzazione in compagnia dei contenuti multimediali.
  - 1.3 Riproduzione sincronizzata del video tra tutti i partecipanti del party.

## **2.2 Requisiti utente**

In particolare l'utente può usufruire dei seguenti aspetti:

- 2 Utente.
  - 2.1 Autenticazione al sito;
    - 2.1.1 Registrazione di un nuovo utente che non dispone delle credenziali;
    - 2.1.2 Login di un utente che dispone di credenziali di accesso;
    - 2.1.3 Logout di un utente;

- 2.2 Visualizzazione del catalogo dei film
  - 2.2.1 Disposizione dei film per categoria;
    - 2.2.1.1 Trending Now;
    - 2.2.1.2 Top Rated;
    - 2.2.1.3 Action Movies;
    - 2.2.1.4 Comedy Movies;
    - 2.2.1.5 Fantasy Movies;
    - 2.2.1.6 Animation Movies;
  - 2.2.2 Selezione di un film tramite apposita barra di ricerca.
- 2.3 Visualizzazione della lista amici
  - 2.3.1 Visualizzazione dello status (online, offline) per ogni utente presente nella lista amici;
- 2.4 Possibilità di aggiungere utenti alla lista amici tramite apposita richiesta di amicizia.
- 2.5 Visualizzazione delle richieste di amicizia ricevute tramite apposita sezione e notifica.
  - 2.5.1 Sezione per l'accettazione o il rifiuto della richiesta.
- 2.6 Visualizzazione dei badge ottenuti dall'utente.
  - 2.6.1 Distinzione grafica tra i badge posseduti e quelli ancora bloccati.
- 2.7 Fruizione del contenuto multimediale
  - 2.7.1 Possibilità di visualizzare il contenuto multimediale in solitaria
  - 2.7.2 Possibilità di visualizzare il contenuto multimediale con amici tramite creazione di un party
- 2.8 Possibilità di creazione di un party
  - 2.8.1 Scelta dei partecipanti presenti nella lista amici tramite apposito invito
  - 2.8.2 Visualizzazione di una lobby di attesa per gli utenti che accettano l'invito al party
    - 2.8.2.1 Possibilità di scambio di messaggi tramite chat per gli utenti presenti in lobby
    - 2.8.2.2 Visualizzazione del contenuto multimediale e della chat all'avvio del party

## 2.3 Requisiti funzionali

L'applicazione prodotta dovrà:

### 3 Funzionali.

- 3.1 Gestire l'autenticazione degli utenti al sito;

- 3.1.1 Registrazione di utenti che non possiedono credenziali;
  - 3.1.1.1 Richiesta obbligatoria delle seguenti informazioni: username, email, password e conferma della password;
  - 3.1.1.1.1 Verifica unicità dello username;
  - 3.1.1.1.2 Verifica semantica dell'email;
  - 3.1.1.1.3 Verifica che la password e la conferma della password coincidano;
  - 3.1.1.1.4 Hashing della password per la memorizzazione nel database;
- 3.1.2 Login di utenti che possiedono credenziali
  - 3.1.2.1 Richiesta di email e password per l'autenticazione al sito;
  - 3.1.2.1.1 Verifica dell'esistenza di un utente con l'email specificata
  - 3.1.2.1.2 Verifica che la password inserita coincida con la password dell'utente che possiede l'email specificata.
- 3.1.3 Logout di utenti autenticati al sito;
- 3.2 Visualizzare una notifica in seguito a specifici eventi;
  - 3.2.1 Visualizzazione di una notifica pop-up nel caso l'utente interessato sia online;
    - 3.2.1.1 Ricezione di una richiesta di amicizia;
    - 3.2.1.2 Accettazione di una richiesta di amicizia spedita precedentemente;
    - 3.2.1.3 Richiesta di amicizia ad un utente non esistente
    - 3.2.1.4 Richiesta di amicizia già presente per uno specifico utente
    - 3.2.1.5 Guadagno di un nuovo badge
  - 3.2.2 Visualizzazione delle richieste di amicizia ricevute mentre l'utente era offline;
- 3.3 Visualizzare una notifica in seguito a specifici eventi;
  - 3.3.1 I badge sbloccati possiedono le seguenti caratteristiche: icona univoca, titolo descrizione;
  - 3.3.2 I badge bloccati possiedono le seguenti caratteristiche: icona standard equivalente per tutti i badge non posseduti, descrizione;
- 3.4 Gestione della lista amici di ogni utente;
  - 3.4.1 Possibilità di aggiungere nuovi amici tramite richiesta di amicizia. Il destinatario della richiesta di amicizia deve essere specificato attraverso lo username;
    - 3.4.1.1 Il destinatario della richiesta di amicizia deve esistere nel database.

- 3.4.1.2 Il destinatario della richiesta di amicizia non deve essere già presente nella lista amici.
- 3.4.2 Visualizzazione della lista amici, come riportato in 2.3;
- 3.5 Possibilità di ricercare uno specifico film all'interno del catalogo attraverso la modalità descritta in 2.2;
- 3.6 Creazione di un party selezionando i partecipanti dalla propria lista amici.
  - 3.6.1 Il creatore del party potrà selezionare quali utenti online presenti nella sua lista amici invitare.
  - 3.6.2 Possibilità di accettare o declinare l'invito da parte di chi ha ricevuto la richiesta di partecipazione al party;
    - 3.6.2.1 L'utente che accetta la richiesta di partecipazione entrerà in lobby in attesa che il creatore avvi il party;
    - 3.6.2.1.1 Possibilità di scambi di messaggi tra i partecipanti della lobby attraverso una chat dedicata;
  - 3.6.3 Avvio del party con conseguente riproduzione del contenuto multimediale e chat in real-time;
    - 3.6.3.1 Possibilità per chiunque partecipi al party di effettuare azioni sul contenuto multimediale che verranno sincronizzate con tutti gli utenti;
      - 3.6.3.1.1 Mettere in pausa;
      - 3.6.3.1.2 Avanzare nella riproduzione del video
      - 3.6.3.1.3 Indietreggiare nella riproduzione del video
    - 3.6.3.2 Possibilità per chiunque partecipi al party di usufruire della chat real-time
      - 3.6.3.2.1 Invio di messaggi;
      - 3.6.3.2.2 Ricezione di messaggi;

## 2.4 Requisiti non funzionali

- 4 Non funzionali.
  - 4.1 Interfaccia utente responsive.

## 2.5 Requisiti di implementazione

- 5 Implementazione.
  - 5.1 L'applicazione verrà sviluppata utilizzando lo stack MERN.
  - 5.2 Utilizzo della libreria Socket.IO per la realizzazione di comunicazioni real-time

5.3 Le informazioni riguardanti i film saranno ottenute attraverso l'utilizzo dell'API fornita da <https://developers.themoviedb.org/>

# Capitolo 3

## Design

### 3.1 Design dell'architettura del sistema

L'applicativo è stato costruito sulla base dello stack MERN, che come l'acronimo descrive comprende le seguenti tecnologie:

- MongoDB
- Express
- React
- Node.js

MERN è una delle numerose varianti dello stack MEAN (dove il tradizionale framework frontend Angular.js viene sostituito con React.js) che ci ha permesso di costruire facilmente un'architettura a 3 livelli (frontend, backend, database) utilizzando interamente Javascript e JSON.

All'interno del progetto sono state utilizzate diverse tecnologie, che differiscono da frontend a backend. Per la parte frontend sono state utilizzate Axios, Redux e React; mentre per la parte backend sono state utilizzate express e moongoose. Queste tecnologie sono descritte in dettaglio nel capitolo 5. Infine, è importante precisare che MongoDB grazie alla memorizzazione di documenti JSON, ha facilitato notevolmente l'archiviazione, la manipolazione e la rappresentazione dei dati ad ogni livello dell'applicazione.

#### 3.1.1 Componenti funzionali (React Hooks)

In questo elaborato sono stati utilizzati gli Hook, una nuova aggiunta disponibile da React 16.8. Gli hook consentono di mantenere semplici i componenti, in quanto non è necessario gestire lo stato del componente e tutti i suoi effetti collaterali, evitando così di introdurre bug e incongruenze. Sostanzialmente gli hook sono funzioni che consentono di “agganciarsi” allo stato di React e al ciclo

di vita dei componenti senza utilizzare classi. Tipicamente gli hook più utilizzati all'interno del progetto sono stati:

- useState: consente di aggiungere uno stato ad un componente React;
- useEffect: consente di eseguire effetti collaterali sui componenti funzionali.

### 3.1.2 JSX

All'interno del progetto il team ha deciso di utilizzare la sintassi JSX fornita da React. Si è optato per l'utilizzo di JSX in quanto essendo un'estensione di Javascript è semplice ed intuitiva, e cosa non meno importante, consente di usufruire di tutta la sua potenza. Infatti, invece di separare artificialmente le tecnologie inserendo il codice di markup e la logica applicativa in file separati, React consente di separare le responsabilità utilizzando unità debolmente accoppiate chiamate “componenti” che contengono entrambi (cit. React). In dettaglio, è possibile vedere come JSX è stato applicato nel capitolo 5.x.

## 3.2 Metodologie di sviluppo

Per la creazione dell'elaborato è stata utilizzata UCD (User Centered Design) come metodologia di design. La scelta è ricaduta su questa metodologia in quanto sin dalle prime fasi di progettazione, il team di sviluppo ha messo in primo piano la HCI (Human Computer Interaction), cercando di focalizzarsi sulle esigenze degli utenti e sull'ottenimento di una buona usabilità (andando così a migliorare la User Experience). Il team di sviluppo si è infatti focalizzato sulla applicazione da implementare, con l'obiettivo di rispondere ai bisogni e alle richieste degli utenti, sulla base delle loro caratteristiche. Gli utenti a cui ci si sta riferendo erano “virtualizzati”, ovvero una serie di Personas che rispecchiano il target dell'applicativo. Oltre a ciò, per aumentare la user experience sono stati utilizzati i seguenti principi:

- Responsive Design: sono state realizzate pagine Web in grado di adattarsi graficamente e in modo automatico ai dispositivi coi quali vengono visualizzati;
- Mobile First: sono state progettate prima le interfacce per i dispositivi più limitanti (smartphone e tablet) per poi passare alla progettazione di interfacce per i PC;
- KISS: le interfacce sono state realizzate per contenere solamente gli elementi fondamentali ed in modo facilmente accessibile all'utente.

Questi principi e metodologie di design sono state poi integrate all'interno della metodologia di sviluppo AGILE. Tutto questo è stato poi integrato con la metodologia di sviluppo AGILE che ha permesso la realizzazione del progetto per fasi, denominate sprint, ognuna delle quali focalizzata su nuove funzioni. I

meeting tra i componenti del team sono avvenuti con frequenza regolare mediante la piattaforma Microsoft Teams. All'inizio di ogni settimana sono stati effettuati in modo congiunto la Sprint Review e lo Sprint Planning. In questo tipo di meeting è stato passato in rassegna il lavoro svolto nella settimana appena conclusa e sono stati determinati gli obiettivi e i compiti da svolgere per ciascun membro del team. Ad ogni giornata di lavoro si è svolto il daily scrum, in cui i membri del team si aggiornavano sui progressi svolti e le difficoltà riscontrate, in maniera tale da studiare insieme possibili soluzioni.

### 3.3 Mockup iniziale

Per la realizzazione dell'applicazione dal punto di vista dello user interface design, in particolare per quanto riguarda la progettazione visiva e dell'interazione si è cercato di mantenere uno stile il più semplice ed intuitivo possibile.

In un prima fase di design si è fatto ricorso dell'uso di mockup al fine di avere una massima dell'applicativo finale. Si è fatto uso del tool di sviluppo Balsamiq per la creazione di wireframes. L'app è progettata per essere frutta al massimo delle sue potenzialità in modalità desktop, in quanto più adeguato alla riproduzione di film e risorse multimediali, in aggiunta di una chat real-time. Tuttavia si è prestata attenzione anche all'interfaccia per dispositivi tablet e mobile in modo da rendere l'esperienza il più godibile possibile. In seguito a primi sketch e feedback ricevuto da alcuni utenti l'interfaccia del sito è riassunta come segue:

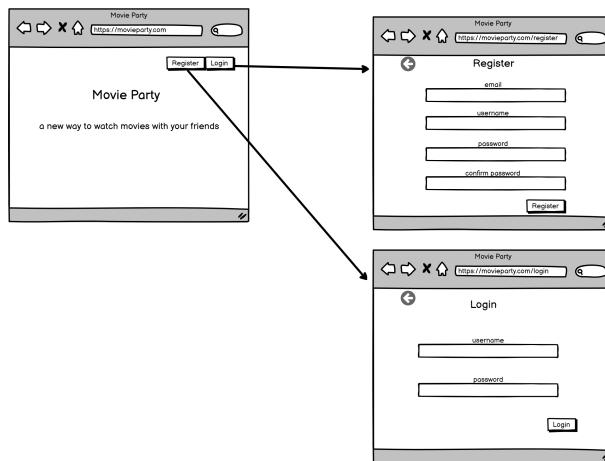


Figura 3.1: login-register

Il sito si presenta con una schermata minimal e dalle funzionalità ridotte, l'intento di questa schermata è quella di far registrare un utente, o nel caso in cui abbia a disposizione delle credenziali la possibilità di eseguire il login.

Una volta effettuato il login si viene reindirizzati alla dashboard principale, che si presenta come segue:

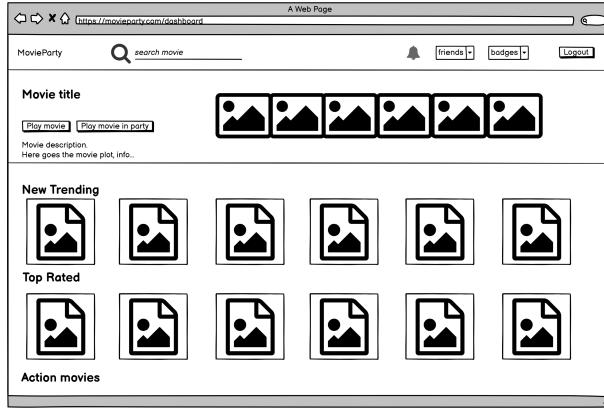


Figura 3.2: dashboard

Nella dashboard vi è presente una navbar che permette di utilizzare diverse funzioni come ad esempio la ricerca di un film o visualizzare la lista di amici dell'utente. Da qui si potranno inviare richieste di amicizia per ampliare la lista amici o visualizzare le notifiche perse. Inoltre, vi sono presenti i badges, ovvero collezionabili che l'utente potrà sbloccare eseguendo specifiche azioni, aggiunto come elementi di gamification.

Direttamente sotto alla navbar si visualizza il film cercato nella barra di ricerca mentre al di sotto sono presenti un insieme selezionato di film in base alla categoria, ad esempio nei trending vi saranno presenti i film di successo dell'ultimo periodo. Selezionando un poster da una sezione si aprirà un banner dove si vedranno informazioni riguardanti il film e la possibilità di riprodurre lo stesso.

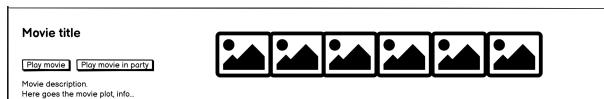


Figura 3.3: banner

Il banner, Fig. 3.3, presenta il titolo del film, un suo poster a ricoprire il background la descrizione della trama e la possibilità di riprodurre il trailer o di visionare il film in modalità party. Selezionando il trailer si aprirà un player interno adibito alla riproduzione di quest'ultimo, come rappresentato in figura 3.4, invece, se viene avviato il party, l'utente verrà spedito nella lobby fig. 3.5.

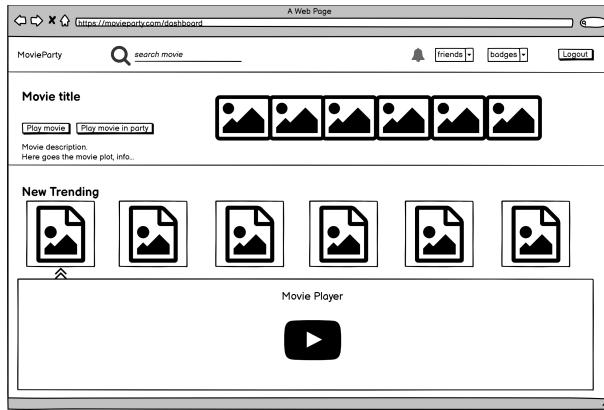


Figura 3.4: dashboard-to-player

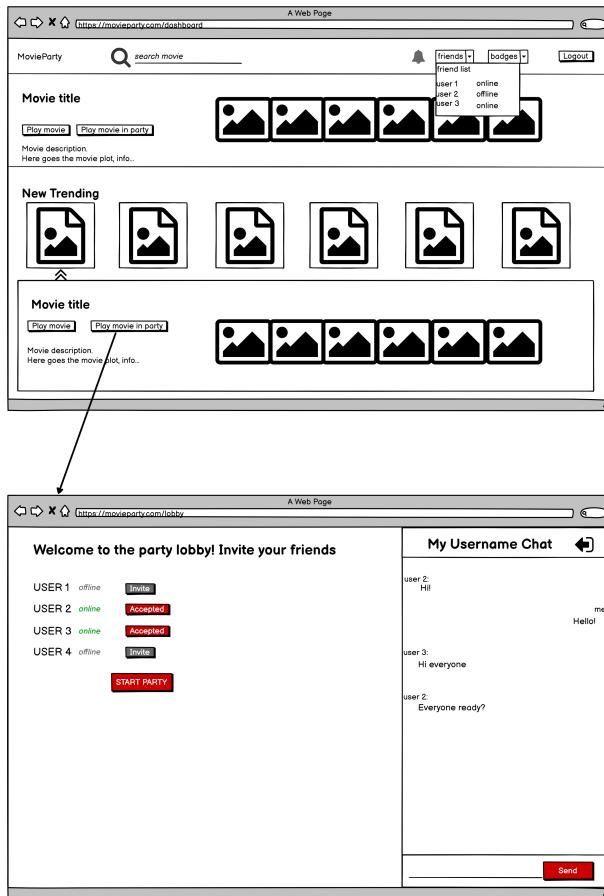


Figura 3.5: dashboard-to-lobby

Nella lobby il leader potrà invitare gli utenti online appartenenti alla sua lista amici. Vi è anche presente una chat che dà la possibilità ai partecipanti del party di interagire tra di loro, in attesa che il creatore del party decida di avviare la riproduzione sincronizzata del film.

Una volta che viene avviata la riproduzione i partecipanti visualizzano la seguente schermata in Fig. 3.6.

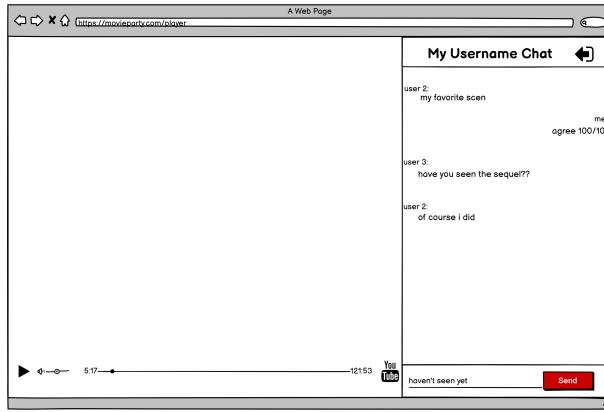


Figura 3.6: player with chat

Similmente a prima vi è presente la chat dove gli utenti possono comunicare durante la visione del film grazie ad un player interno. Durante la visione sincronizzata gli utenti possono interagire con il player mettendolo in pausa, avanzando di tempo o indietreggiando e le azioni verranno propagate a tutti i partecipanti del party.

La versione mobile non si differenzia molto da quella desktop, anche se è presente qualche rivisitazione, dovuta alla ridotta dimensione dello schermo. La schermata della dashboard risulta come segue in Fig 3.7.

I dispositivi mobile vengono solitamente utilizzati in verticale, e utilizzando il dispositivo posizione verticale si è cercato di strutturare l'applicazione in maniera tale da sfruttare tutta l'area disponibile mantenendo comunque l'interfaccia semplice ed intuitiva, perdendo il minor numero di informazioni disponibili rendendo la user experience godibile tanto quanto la versione desktop.

Per quanto riguarda la lobby pre party si ha il risultato riportato in Fig 3.8.

Anche in questo caso si è sfruttata la verticalità del dispositivo per costruire una struttura che desse all'utente la sensazione che ogni parte dello schermo fosse utilizzata. Similmente viene anche costruita la schermata del player, Fig 3.9.

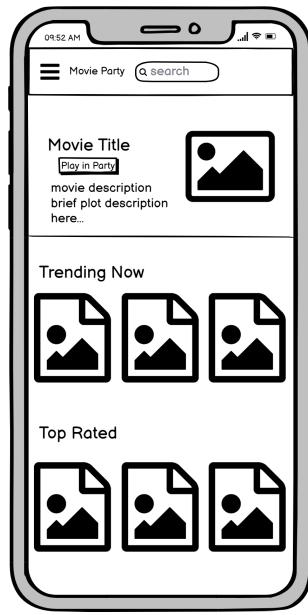


Figura 3.7: mobile dashboard

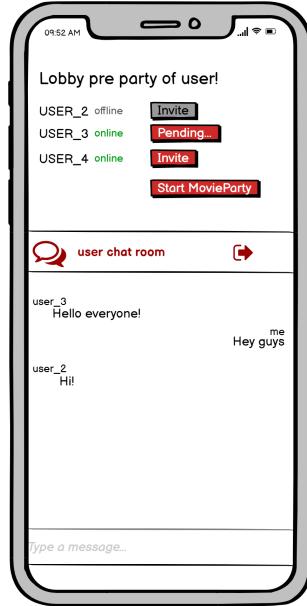


Figura 3.8: mobile lobby



Figura 3.9: mobile player

### 3.4 Target User Analysis

L'idea di business, dovrà essere rivolta ad uno specifico gruppo di potenziali clienti. Il segmento di clienti individuato e scelto in quanto si reputi il più adatto a recepire e, soprattutto, a desiderare l'applicazione.

Il target quindi è il pubblico formato da individui con caratteristiche, esigenze e necessità in comune con quelle individuate e scelte dalle funzionalità del sito web. Identificare e segmentare il pubblico non in base alla quantità ma alla qualità in quanto sono le persone interessate al prodotto o al servizio ad avere più possibilità di diventare dei clienti effettivi. Avere un target definito e ben delineato serve a impostare una strategia di marketing precisa e specifica, perciò si segue l'analisi considerando i seguenti punti:

- Contenuti specifici: sapere qual è il pubblico a cui si è rivolti permette di creare messaggi specifici e appropriati, comunicare con il pubblico sulla base dei loro interessi è la maniera migliore per convertire potenziali clienti in clienti effettivi.
- Ottimizzare il sito: bisogna pensare a come naviga il target audience scelto, se da pc, tablet, smartphone o addirittura tutti e tre i dispositivi. Se il target ha dimestichezza con il mondo online o ha difficoltà, e sapere come si muove tra le schermate del sito è importante perché lo si può ottimizzare per favorire una migliore esperienza ai potenziali clienti.

- Concentrarsi sul potenziale: non è necessario né utile spendere tempo e risorse per raggiungere tutti bensì bisogna concentrare l'attenzione su un pubblico più propenso ad acquistare. L'idea di base è quella di seguire un potenziale pubblico bene piuttosto che tutti ma male.
- Strategie di successo: quando si sa a quale pubblico si mira automaticamente si facilita anche il compito di scegliere quali media e quali tecniche usare e facilita l'applicazione strategie di successo, pertinenti e coerenti con esso.

Definite le principali guidelines da tenere in considerazione si è fatta un'analisi sulla target audience. I target di riferimento dell'applicazione risultano quindi essere quei potenziali clienti che si trovano nella necessità o desiderio di visionare un prodotto multimediale in compagnia con altri utenti, o amici online. Non risulta comodo per gli utenti per la visione di un film utilizzare qualche metodo per la coordinazione come potrebbero essere chiamate su piattaforme come Discord oppure Skype per gestire la sincronicità dell'avvio del film. Inoltre nel momento in cui un utente abbia intenzione di mettere in pausa il video per necessità personale, ha bisogno di un'ulteriore coordinazione con gli amici per eseguire queste semplici operazioni. L'applicazione mira a ridurre questi compiti tediosi che l'utente deve eseguire, non dovrà quindi preoccuparsi della sincronizzazione del video, ad esempio coordinandosi con un suo amico per la partenza del video. Inoltre vengono demandate all'applicazione anche tutte le operazioni di seek forward, seek backward e pausa del video stesso. Vi è anche a disposizione una chat real-time utilizzabile per la comunicazione e l'interazione tra gli utenti partecipanti ad un party. Tutte queste operazioni non sono divise su diverse piattaforme, come potrebbe essere ad esempio l'uso di Netflix per la visione e quello di Discord per l'interazione tra gli utenti bensì è accorpato tutto sulla stessa applicazione.

Definito un pubblico di riferimento si sono costruiti diversi personas ovvero personaggi immaginari creati per rappresentare utenti che potrebbero utilizzare il sito.

#### **Personas:** Alessandro

Alessandro è un ragazzo appassionato di cinema e Serie TV e viene a conoscenza del sito.

Scenario d'uso: Alessandro vuole fruire di prodotti multimediali, e talvolta anche in compagnia del suo gruppo di amici, tuttavia non disdegna la visione in solitaria

1. Alessandro accede alla piattaforma
2. Si iscrive registrandosi con i propri dati
3. Sceglie il film da visionare
4. Nessun amico è momentaneamente online, allora Alessandro decide di avviare comunque il film

### **Personas:** Francesca

Francesca è una ragazza che solitamente si raggruppa con le amiche per la visione delle sue serie tv preferite. A causa delle regolamentazioni dovute al covid, però, non è riuscita ancora a vedere l'ultima stagione della sua serie tv preferita con le amiche.

Scenario d'uso: Francesca è venuta a conoscenza della piattaforma e informa anche le sue amiche

1. Francesca accede alla piattaforma
2. Si iscrive registrandosi con i propri dati
3. Aggiunge alla “lista amici” tutte le sue amiche
4. Avvia la lobby del party con la sua serie tv preferita
5. Invita le amiche a partecipare al party
6. Avvia il party, visionando la serie tv con le sue amiche

## **3.5 Storyboard**

Il prodotto finale non si discosta molto dai mockup iniziali, risulta tuttavia revisionato tenendo in considerazione il feedback degli utenti che hanno avuto un accesso anticipato di un primo prototipo dell'applicazione.

La pagina iniziale si mostra come segue in Fig 3.10.



Figura 3.10: home-desk

Da questa schermata l'utente potrà registrarsi nel caso in cui non abbia ancora le credenziali per l'accesso al sito, Fig 3.11.

Se l'utente durante la registrazione al sito, commette errori esso viene notificato visivamente, nello specifico si ha un warning visivo per gli errori riguardanti l'inserimento di uno username già presente nel database, l'inserimento di una

The screenshot shows a registration form titled "Register". At the top right is a "BACK TO HOME" link and a "Log in" link. The form has four input fields: "Name" (with placeholder "utente1"), "Email" (with placeholder "utente1@gmail.com"), "Password" (with placeholder "\*\*\*\*\*"), and "Confirm Password" (with placeholder "\*\*\*\*\*"). Below the fields is a "SIGN UP" button.

Figura 3.11: register page

mail semanticamente non valida, l'utilizzo di una password che non supera certi criteri di sicurezza e l'errore riguardante il match tra la password inserita e la riconferma della stessa. Di seguito verranno riportate tali schermate (da Fig 3.12 a Fig 3.15)

The screenshot shows the same registration form as Fig 3.11. The "Name" field contains "utente1", which is highlighted with a red border. To its right, the text "Username already exists" is displayed in red. The other fields (Email, Password, Confirm Password) and the "SIGN UP" button are visible below.

Figura 3.12: username not exists

Nel caso in cui si possiedano le credenziali sarà possibile eseguire l'accesso al sito dalla schermata di login come mostrato in Fig 3.16

Così come per la registrazione anche per il login saranno presenti warning visivi nel caso in cui le credenziali non rispecchiano nessun utente disponibile nel database, o dell'invalidità della mail (Fig 3.17 e 3.18)

Dalla schermata di login è possibile navigare verso la schermata di registrazione e viceversa, e da entrambe è possibile tornare alla schermata di home.

← BACK TO HOME

## Register

Already have an account? [Log in](#)

Name

Email  
  
Email is invalid

Password

Confirm Password

[SIGN UP](#)

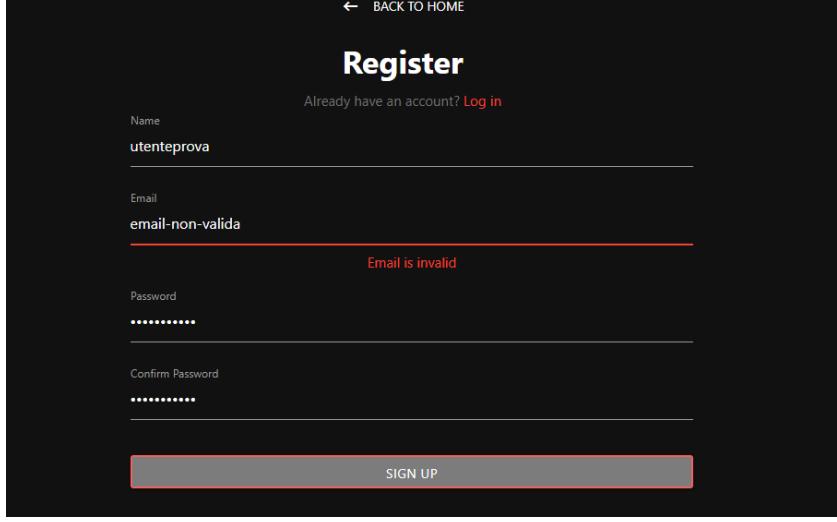


Figura 3.13: non valid mail

← BACK TO HOME

## Register

Already have an account? [Log in](#)

Name

Email

Password  
  
Password must be at least 6 characters

Confirm Password

[SIGN UP](#)

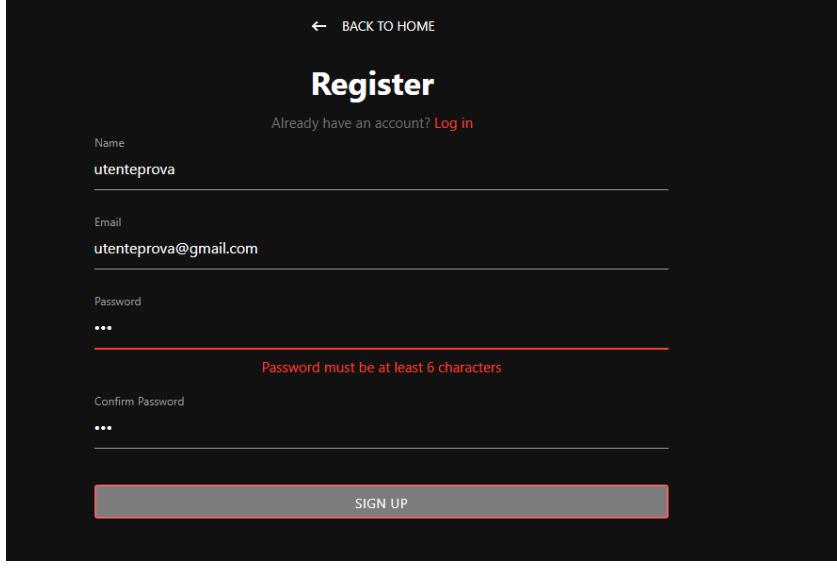


Figura 3.14: password not valid

← BACK TO HOME

## Register

Already have an account? [Log in](#)

Name

Email

Password

Confirm Password

Passwords must match

[SIGN UP](#)

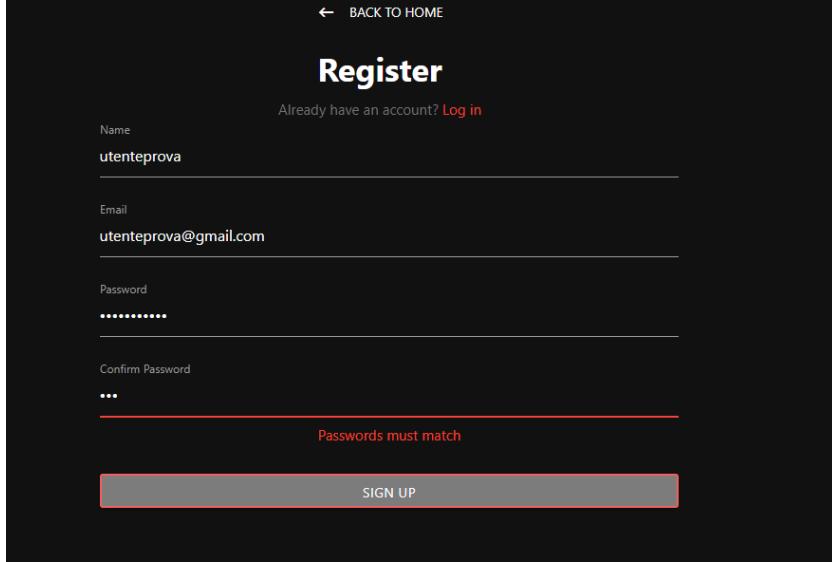
A screenshot of a registration form on a dark-themed website. The form includes fields for Name, Email, and Password. The Password and Confirm Password fields do not match, resulting in a red error message below them. A large red border highlights the 'SIGN UP' button.

Figura 3.15: password does not match

← BACK TO HOME

## Login

Don't have an account? [Register](#)

Email

Password

[LOGIN](#)

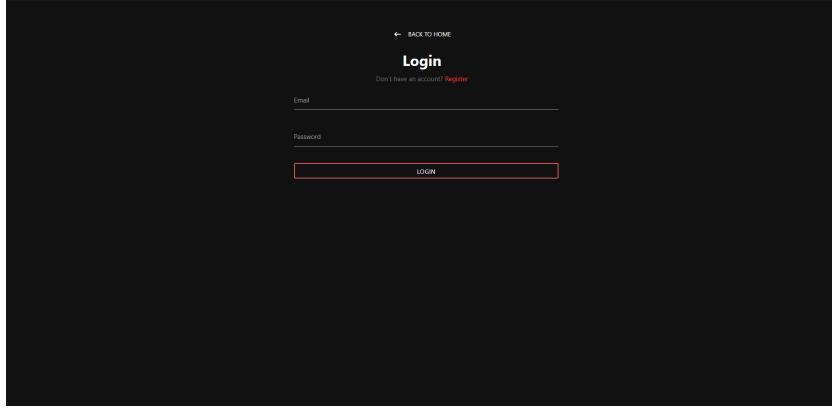
A screenshot of a login form on a dark-themed website. It features fields for Email and Password, and a 'LOGIN' button with a red border.

Figura 3.16: login page

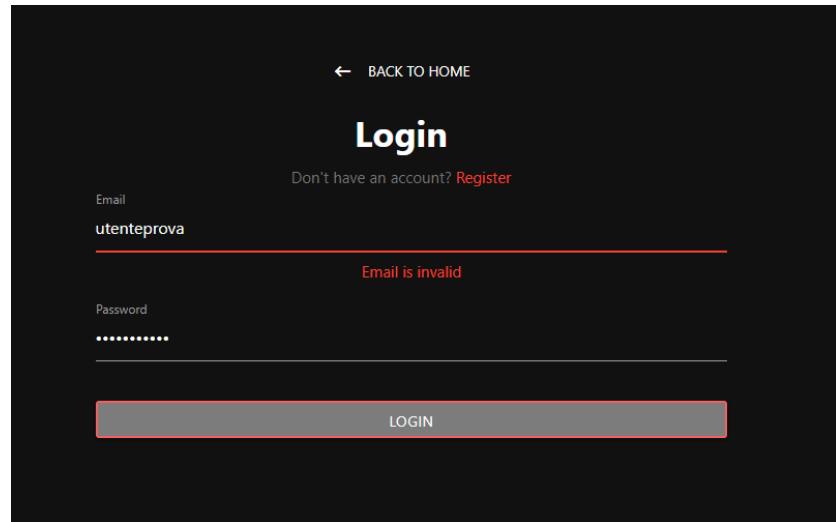


Figura 3.17: login mail not valid

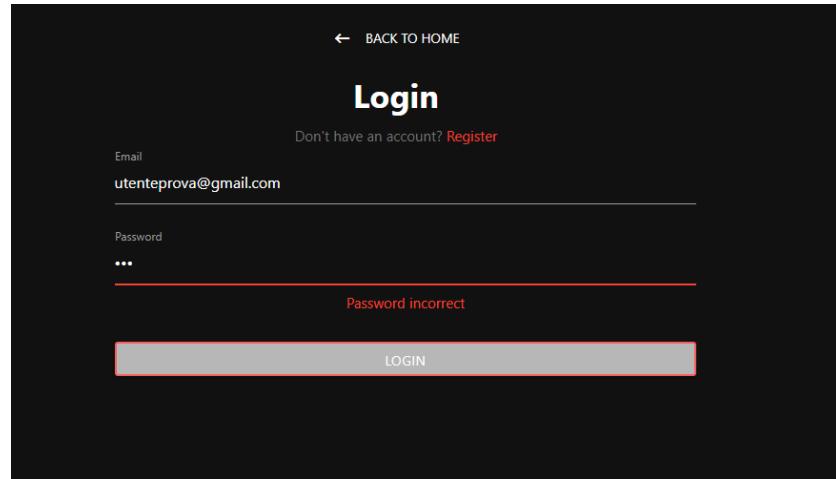


Figura 3.18: login password incorrect

Una volta che un utente registrato abbia eseguito l'accesso al sito esso verrà riportato alla dashboard, come viene riportata dalla Fig 3.19.

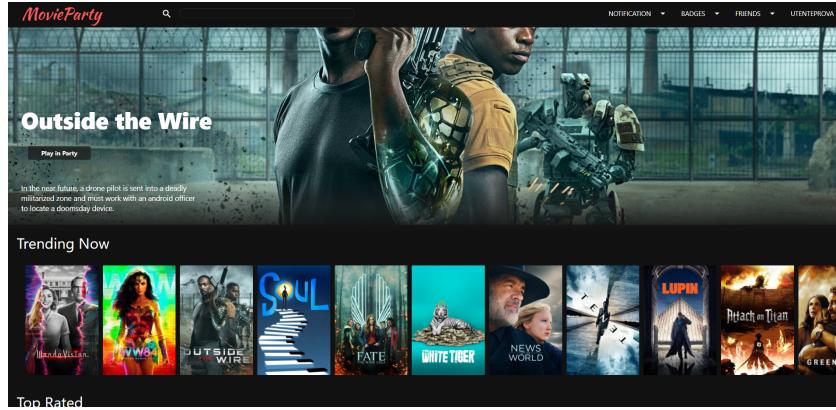


Figura 3.19: Dashboard desktop view

Da questa schermata è possibile tramite la navbar collocata in alto della pagina cercare un film, che apparirà nel banner principale, visualizzare le notifiche, i badge sbloccati completando le missioni oppure vedere la lista amici, con l'opzione di aggiungerne di nuovi.

Per aggiungere nuovi amici, una volta selezionata la voce “Friends” e “add friend” è sufficiente inserire nel form il nome utente dell’amico e inviare la richiesta (Fig 3.20)

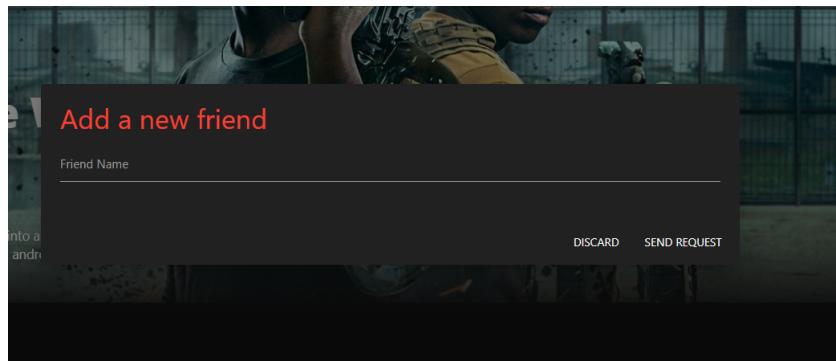


Figura 3.20: Dashboard add friends

All’invio della richiesta se l’utente risulta online gli verrà visualizzata la notifica pop-up nella schermata, dove potrà accettare o rifiutare la richiesta di amicizia (Fig. 3.21)

Nel caso in cui la richiesta venga accettata dall’utente, aprendo la finestra “Friend” si ritroverà con il nuovo utente come amico (Fig 3.22)



Figura 3.21: Notification pop-up

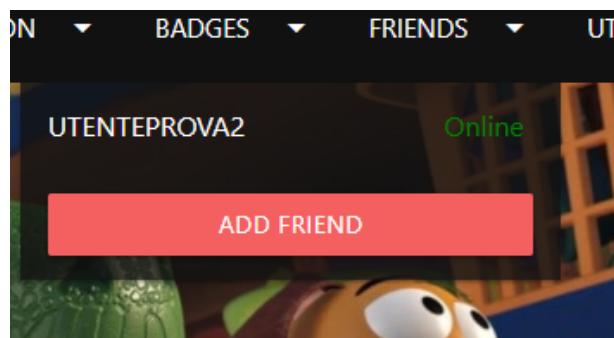


Figura 3.22: online friend

Mentre se vi è un utente offline verrà visualizzata l'icona offline (Fig 3.23)

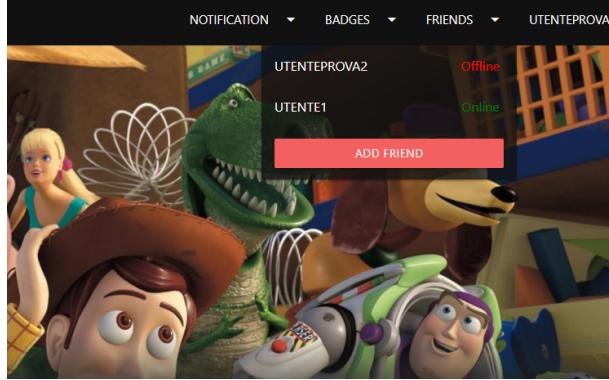


Figura 3.23: offline friend

Nel caso un cui l'utente non abbia visto la notifica in tempo o se non era online all'invio della richiesta questa verrà successivamente visualizzata nelle "Notification". Verrà indicata come "new" quando si ha una nuova notifica non è ancora stata visualizzata dal destinatario (Fig 3.24)

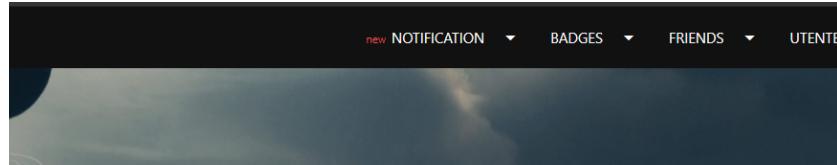


Figura 3.24: new notification

A questo punto il destinatario ricevendo la notifica di invito potrà quindi decidere, similmente a prima se accettare o rifiutare l'amicizia (Fig 3.25)

Nella versione mobile dell'applicazione si avranno le viste in Fig 3.26 e 3.27.

I badges rappresentano un elemento di gamification aggiunto all'applicazione in modo da coinvolgere gli utenti, cercando di divertirli durante gli accessi quotidiani attraverso il gioco, con sfide che potranno essere aggiunte, facendo sbloccare icone di raggiungimento dell'obiettivo in modo da fidelizzare la clientela e rendere l'esperienza con l'app il più piacevole possibile. I badges inizialmente risultano bloccati; non sarà quindi possibile vederne l'icona o il titolo, visibili solamente una volta raggiunto l'obiettivo e aver quindi sbloccato definitivamente il badge (Fig 3.28)

Nella versione mobile si ha una vista come in Fig 3.29.

Nella dashboard è possibile anche navigare tra le categorie (Trending, Top Rated...), dove è possibile selezionare un poster dalla quale si aprirà un banner dove si vedranno informazioni riguardanti il film e la possibilità di riprodurre lo stesso (come mostrato in Fig 3.30)

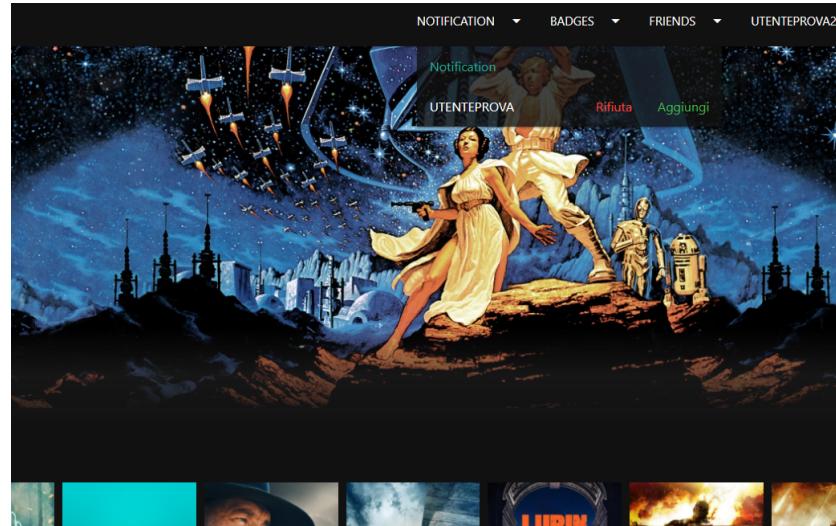


Figura 3.25: handle new notification

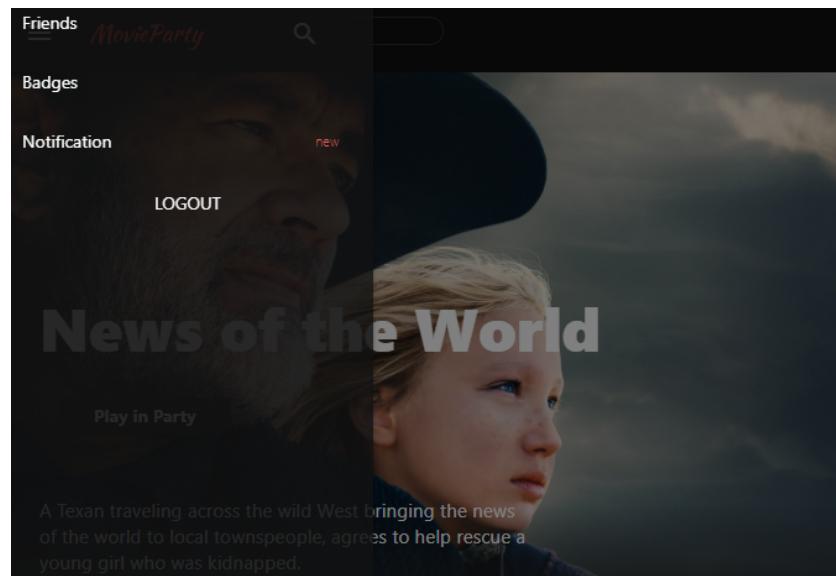


Figura 3.26: new notification from mobile view

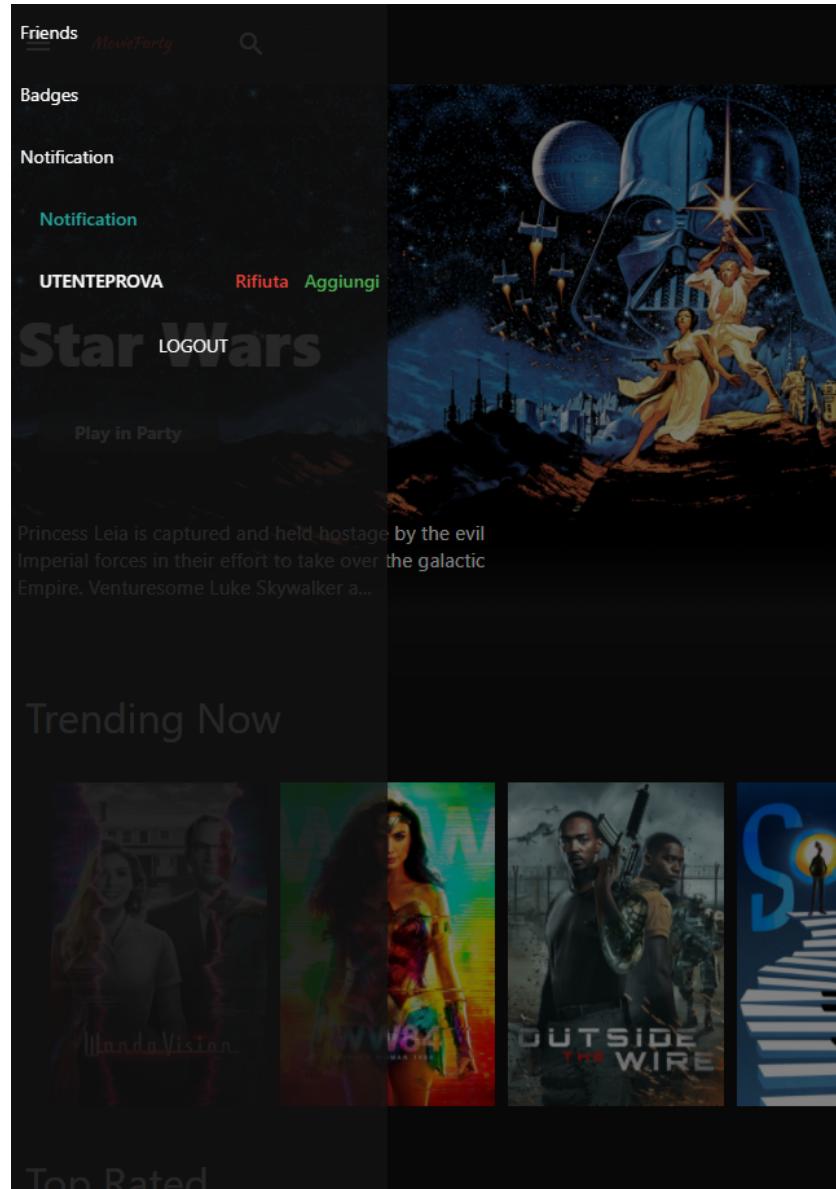


Figura 3.27: handle new notification from mobile view

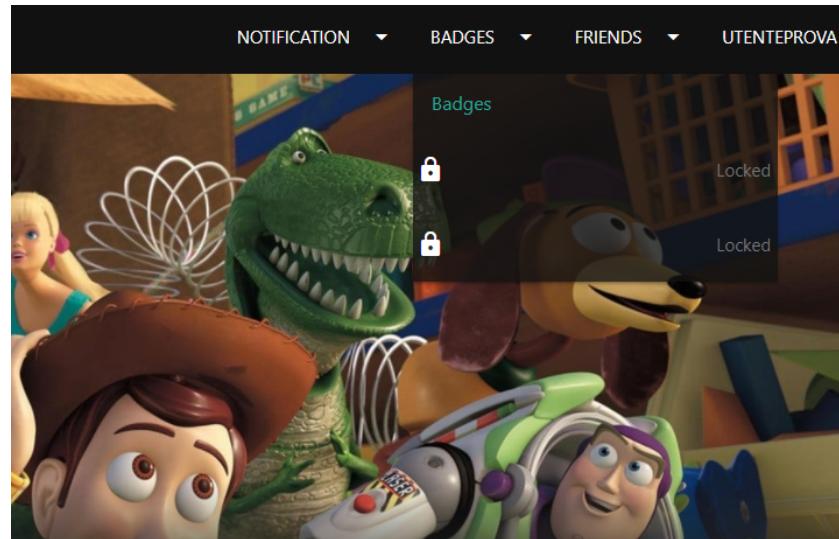


Figura 3.28: locked badges

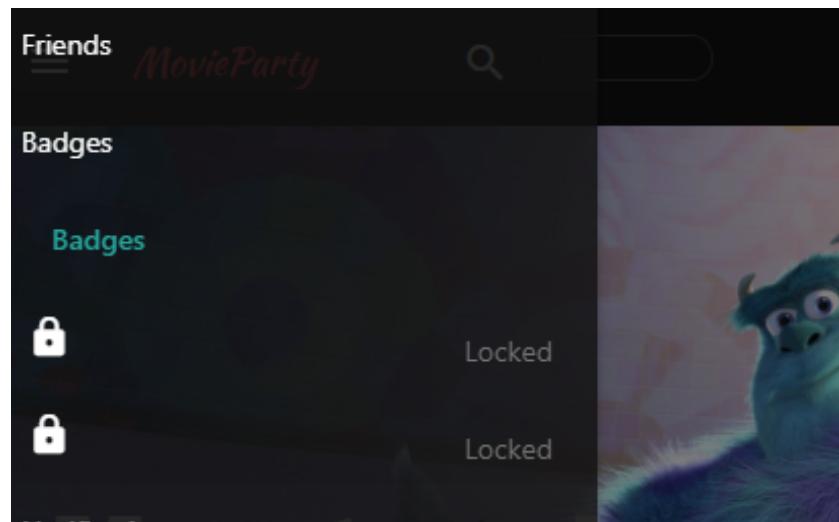


Figura 3.29: locked badges mobile view

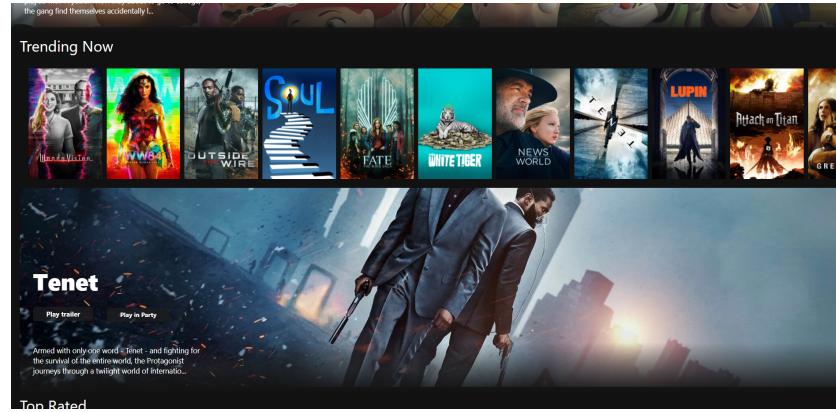


Figura 3.30: banner poster in dashboard

Il banner presenta il titolo del film, un suo poster a ricoprire il background la descrizione della trama e la possibilità di riprodurre il trailer o di visionare il film in modalità party. Selezionando il trailer si aprirà un player interno adibito alla riproduzione di quest'ultimo, come rappresentato in figura 3.31

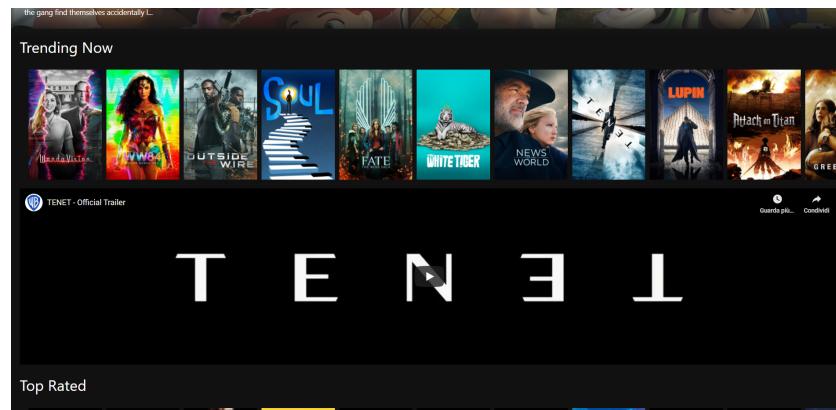


Figura 3.31: banner poster in dashboard

Nella versione mobile, selezionando un poster si avrà la schermata della Fig 3.32

Riproducendo il trailer si avrà la schermata della Fig 3.33

È possibile vedere il trailer direttamente nella schermata del player oppure è anche disponibile la modalità in schermo intero.

Se invece si è intenzionati a vedere il film in party, con qualche amico si verrà portati verso una nuova schermata di lobby per agevolare le azioni da intraprendere prima della riproduzione del film (Fig 3.34)

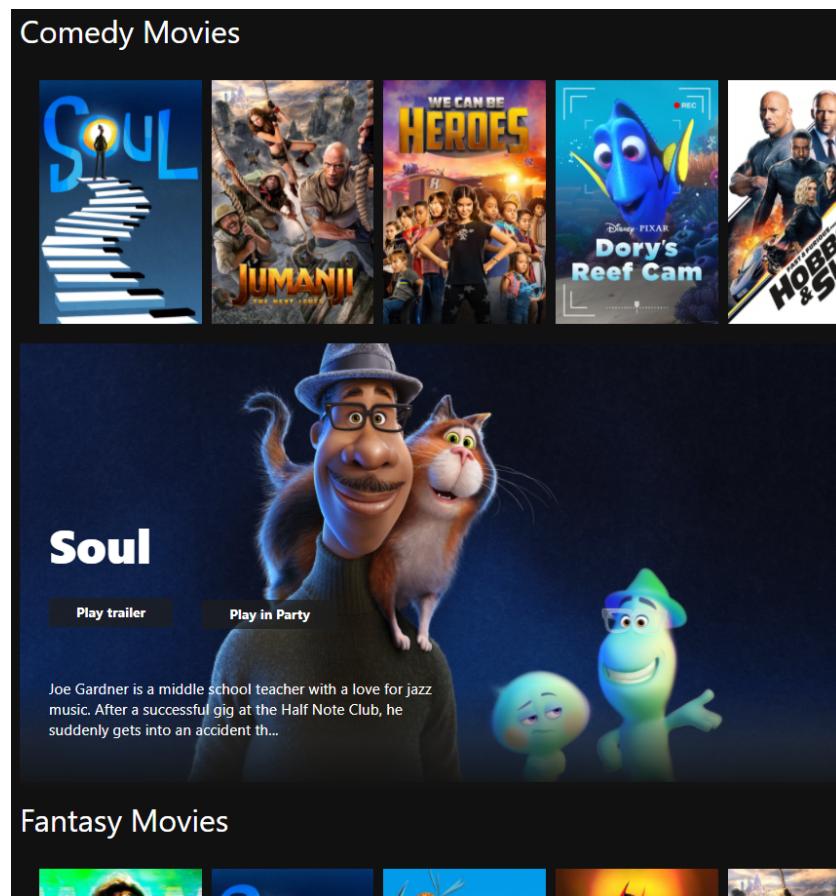


Figura 3.32: trailer from banner poster in mobile view

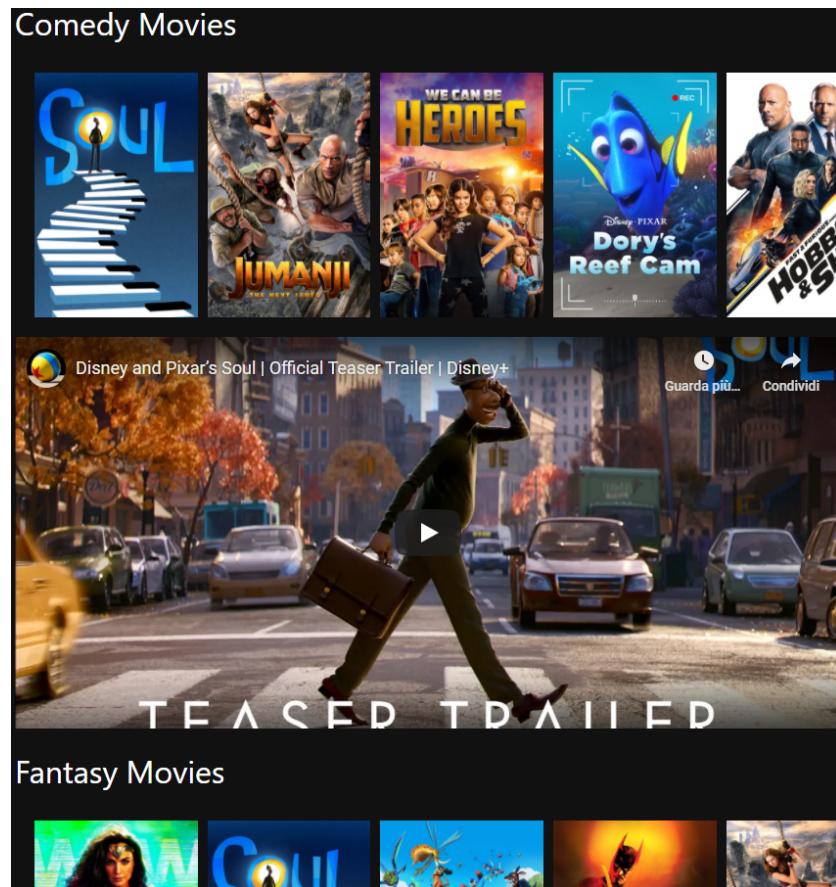


Figura 3.33: trailer player in mobile view

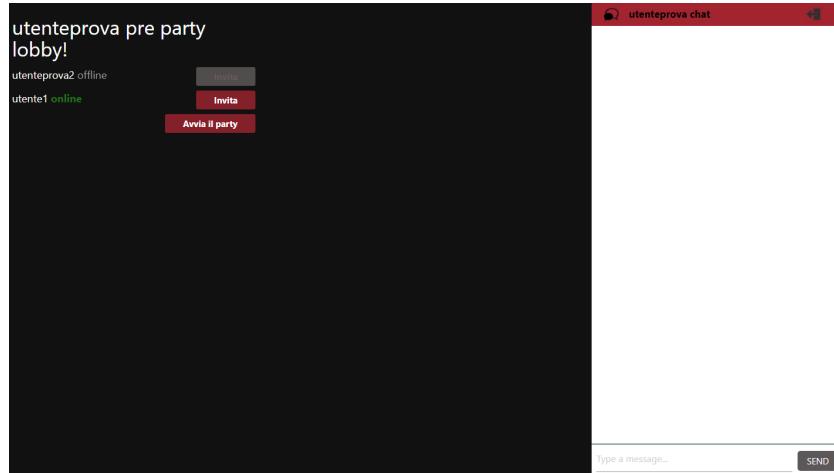


Figura 3.34: lobby page before the start of the party

Da questa schermata il leader potrà visualizzare gli utenti online e quelli offline, avendo quindi a disposibilità la possibilità di inviarne uno o più prima di avviare il party. A destra è disponibile la chat, dove gli utenti potranno comunicare inviando messaggi e organizzarsi, prima dell'inizio del film. Una volta che un utente viene invitato si attende la sua risposta (Fig 3.35)

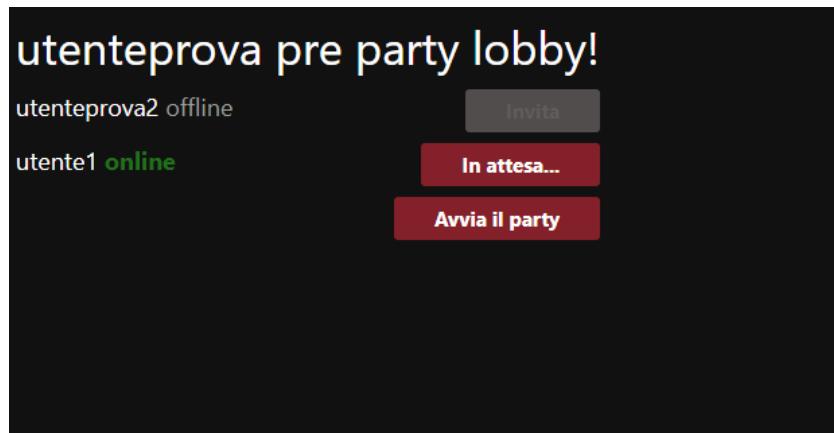


Figura 3.35: waiting response from the friend

Una volta che l'utente accetta viene fatto partecipare al party, come mostrato in figura 3.36.

La versione mobile è concettualmente simile, differenzia solamente dalla posizione della chat, posta nel modo più congeniale ad un dispositivo mobile (Fig 3.37).

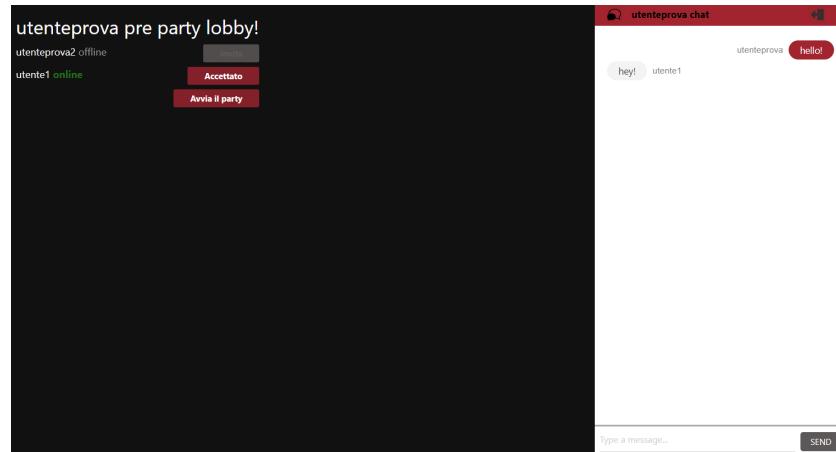


Figura 3.36: chat interaction in lobby

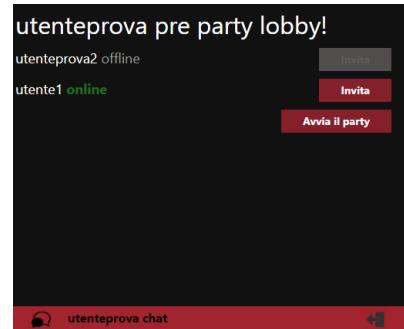


Figura 3.37: lobby page in mobile view

Una volta che il party è pronto il leader fa partire il film, portando la schermata a modificarsi come mostrato in Fig 3.38.



Figura 3.38: movie started in party with chat interaction

Una sezione sostanziale è ovviamente presa dal player che permette la riproduzione del film, permettendo ai dispositivi desktop di sfruttare i loro schermi, mentre a destra è visibile la chat, dove gli utenti partecipanti al party potranno commentare in tempo reale. La visione è sincronizzata tra tutti gli utenti del party, questo significa che saranno allo stesso minutaggio, inoltre è possibile da parte di un utente intraprendere le seguenti azioni che avranno un riscontro anche nella riproduzione degli altri utenti del party, essendo una visione sincronizzata:

- pause: quest'azione mette in pausa il video a tutti gli utenti del party
- seek forward: quest'azione è intrapresa quando un utente avanza nella riproduzione del film, e tutti gli utenti del party vengono sincronizzati allo stesso minutaggio
- seek backward: quest'azione è intrapresa quando un utente indietreggia nella riproduzione del film, e similmente a prima tutti gli utenti del party vengono sincronizzati allo stesso minutaggio.

La versione mobile è stata strutturata in modo da sfruttare la verticalità del dispositivo, e la visualizzazione risulta come riportato in Fig 3.39.

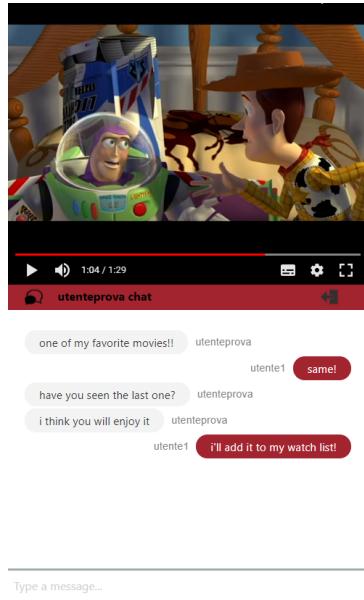


Figura 3.39: movie started in party with chat interaction mobile view

### 3.6 Gamification

Attraverso la gamification è possibile rendere piacevoli, coinvolgenti e divertenti azioni noiose e ripetitive. Viene usato per intrattenere, coinvolgere e ingaggiare gli utenti tramite il gioco, ossia un meccanismo che mette insieme elementi come punti, livelli, ricompense... Attraverso il gioco si vogliono attivare gli utenti a partecipare e svolgere determinate azioni, avvicinando e fidelizzando gli utenti alla piattaforma. Inoltre, grazie ai meccanismi del gioco sarebbe anche possibile monitorare e misurare i comportamenti e immagazzinare dati riguardanti le azioni degli utenti. Attraverso la tecnica di gamification si possono quindi ottenere diverse cose come: ottimizzare il rapporto coi clienti, accrescere la dedizione al brand, rendere l'esperienza d'uso il più piacevole possibile. Il funzionamento della gamification prevede l'utilizzo di alcuni elementi utili a suscitare dei bisogni nei giocatori. In tal modo, l'utente sarà invogliato a continuare il gioco e quindi rimanere nella nostra applicazione per soddisfare tali necessità. Il risultato finale sarà l'engagement, la fidelizzazione e l'intrattenimento degli utenti finali. Nel nostro caso vengono proposte sfide e obiettivi il quale compimento porterà a ricompense virtuali agli utenti e il compimento di tali sfide e obiettivi serviranno per sbloccare i badge, ovvero delle ricompense virtuali. Inizialmente i badge risultano bloccati (Fig 3.40).

Una volta completati gli obiettivi e le sfide proposte, il badge risulterà sbloccato come è possibile notare in Fig 3.41.

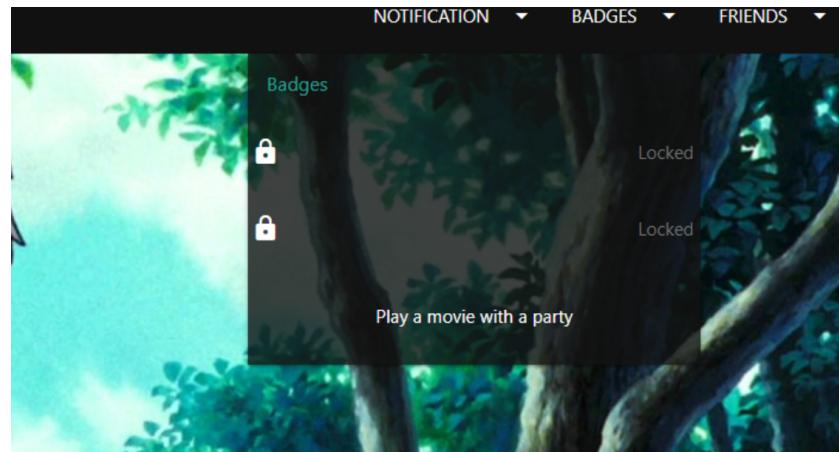


Figura 3.40: locked badges

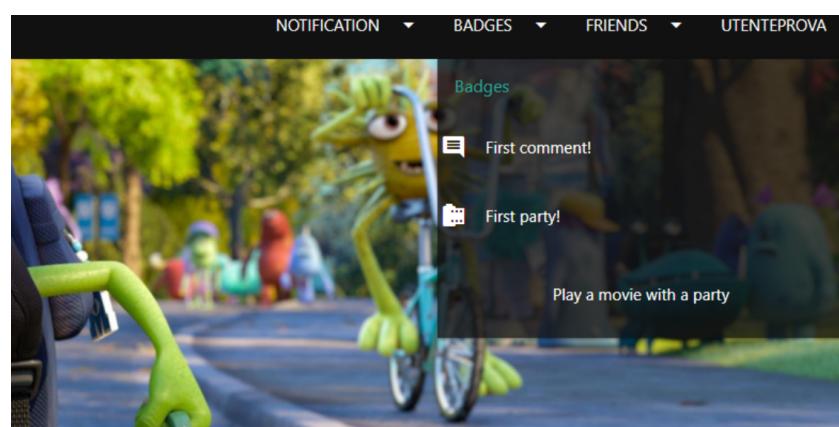


Figura 3.41: unclocked badges

# Capitolo 4

## Tecnologie

### 4.1 Redux

Redux è un contenitore di stati prevedibili ovvero deterministico per le applicazioni JavaScript. La gestione dello stato nelle applicazioni può essere causa di errori e complessità dovuto principalmente a modificabilità e asincronicità. Con Redux si mira a ridurre queste due criticità in quanto esiste una singola fonte di stato, essendo memorizzato in un unico oggetto. Inoltre lo stato è immutabile, sola lettura, non è infatti possibile modificare direttamente lo stato se non tramite funzioni esplicite, dove lo stato corrente viene sostituito con funzioni pure sulla base di una azione e dallo stato precedente. L'architettura generale di Redux è la seguente:

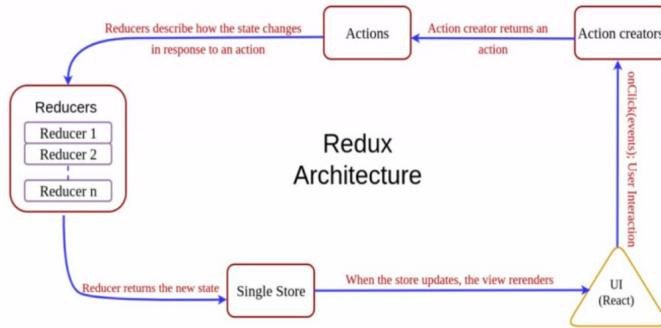


Figura 4.1: Redux architecture

I principali componenti sono i seguenti:

- State: si intende lo stato dell'applicazione ed è costituito da un oggetto con struttura arbitraria, dipendente dalle nostre esigenze applicative.

- Store: si intende il contenitore di Redux che contiene lo stato dell'applicazione consente l'accesso ed è l'unico componente che ne consente la manipolazione dall'esterno.
- Reducers: si intendono funzioni JavaScript che prendono lo stato corrente e un'azione, e ne restituiscono lo stato successivo. È il componente responsabile della transizione tra uno stato e l'altro nel flusso operativo dell'applicazione ed è una funzione pura, ovvero una funzione senza effetti collaterali.
- Azione: si intende un oggetto JavaScript che rappresenta un'azione in grado di sostituire l'attuale stato con un nuovo stato. Per convenzione ha almeno una proprietà type che identifica il tipo di azione e una proprietà payload che contiene eventuali parametri che caratterizzano una specifica azione.
- Dispatcher: si intende il componente che ha il compito di inviare una Action allo Store il quale, tramite un opportuno Reducer, effettua la transizione di stato prevista dal tipo di Action.

In particolare, le possibili transizioni di stato applicabili sono determinate dalle Action previste, il flusso generale per modificare lo stato corrente di un'applicazione segue i seguenti passi: viene individuata una Action che viene inviata allo Store tramite un Dispatcher. All'interno dello Store, un Reducer sostituisce allo stato corrente l'eventuale nuovo stato individuato in base alla Action.

## 4.2 Socket.io

È una libreria JavaScript per le real-time web application, che abilita la comunicazione bidirezionale in tempo reale tra client e server. È composta da due componenti aventi la stessa API, una gira client-side mentre l'altra server-side come libreria per Node.js. Viene utilizzata per gestire la chat e quindi instant messaging e per eseguire push notification. L'idea di base è quella che quando si verifica un evento il server lo pusha direttamente ai client connessi di interesse. Il funzionamento dell'architettura high-level si può riassumere come mostrato in Fig 4.2.

Server e Client sfruttano le le websocket per comunicare, tramite invio di messaggi. I messaggi, o eventi vengono quindi intercettati dai listener registrati.

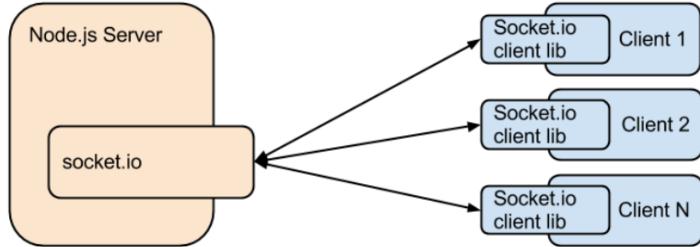


Figura 4.2: Socketio simple client-server interaction

### 4.3 Axios

Axios è usata come libreria promise-based HTTP per JavaScript. Può essere utilizzata dalla parte front-end dell'applicazione e dalla parte back-end in Node.js.

Gli oggetti Promise sono usati per computazioni in differita e asincrone. Una Promise rappresenta un'operazione che non è ancora completata, ma lo sarà in futuro. Utilizzando Axios risulta facile inviare richieste HTTP asincrone agli endpoint REST ed eseguire operazioni CRUD.

Con il concetto di asincronicità viene introdotto quello della promise ovvero un proxy per un valore non necessariamente noto quando la promise è stata creata. Consente di associare degli handlers con il successo o il fallimento di un'azione asincrona e il valore di ritorno in caso di successo, o intercettare l'errore in caso di fallimento. Questo consente di utilizzare dei metodi asincroni di fatto come se fossero sincroni: la funzione che compie del lavoro asincrono non ritorna il valore di completamento ma ritorna una promise, tramite la quale si potrà ottenere il valore di completamento una volta che la promise sarà terminata. Una Promise ha tre diversi stati:

- pending (attesa): stato iniziale, né soddisfatto né respinto.
- fulfilled (soddisfatto): significa che l'operazione si è conclusa con successo.
- rejected (respinto): significa che l'operazione è fallita.

Una promise in pending può evolvere sia in fulfilled con un valore, sia in rejected con un'errore. Quando da pending lo stato diventa fulfilled o rejected, vengono chiamati gli handler associati che sono stati accodati dal metodo `then` della promise. Utilizzando lo stack MERN l'architettura con l'aggiunta di Axios diventa come in Fig 4.3.

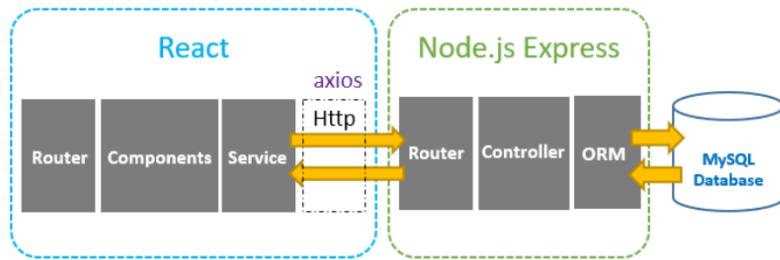


Figura 4.3: Axios interaction in MERN stack

#### 4.4 Materialize

Libreria usata per migliorare lo stile grafico dell'applicazione e quindi la User Experience degli utenti, utilizzando elementi del Material Design ovvero uno stile, un linguaggio di design. È lo stile con cui Google ha deciso di rinnovare tutti i suoi prodotti e sotto il quale gestirli tutti quanti allo stesso modo, con gli stessi principi di esteriorità grafica. Materialize è composta da componenti che incorporano animazioni e forniscono un feedback soddisfacente agli utenti durante l'uso dell'app.

# Capitolo 5

## Codice

Di seguito verranno riportate le porzioni di codice ritenute importanti per la comprensione del progetto.

### 5.1 Gestione stato con Redux

Per l'interazione con lo store Redux è necessario l'utilizzo di due componenti fondamentali: il dispatcher e il reducer. Uno si interessa della spedizione di un dato, potenzialmente strutturato, allo store mentre l'altro esegue una specifica azione in base alle informazioni che ha ricevuto dal dispatcher. Per spedire informazioni allo store è sufficiente:

```
1 store.dispatch({  
2   type: MOVIEPARTY_IS_STARTED,  
3   payload: true  
4 })
```

Nel dispatch si può notare che l'informazione può avere una struttura simile a quella di un oggetto JSON, dove per convenzione ha almeno una proprietà `type` che identifica il tipo di azione e una proprietà `payload` che contiene eventuali parametri che caratterizzano una specifica azione.

I Reducer richiedono lo stato corrente e un'azione, e restituiscono lo stato successivo. È il componente responsabile della transizione da uno stato all'altro del flusso operativo dell'applicazione (Fig 5.2).

```
1 export default function movieparty(state = initialState, action) {  
2   switch (action.type) {  
3     case MOVIEPARTY_IS_STARTED:  
4       {  
5         if(state.movieURL !== "" || state.movieURL ===  
6           undefined){  
7           return {  
8             ...state,  
9             movieparty_isStarted: action.payload,  
10            };  
11        }  
12      }  
13    }  
14  }
```

```

11         return {
12             ...
13             ...state,
14             movieparty_isStarted: false ,
15         };
16     case PARTY_INVITATION:
17     {
18         return {
19             ...
20             ...state,
21             leader: action.payload.sender,
22             room: action.payload.room,
23             movieURL: action.payload.movieURL,
24             inLobby: false
25         }
26     }
27 }

```

## 5.2 Gestione comunicazione real-time con Socket.io

Per la comunicazione in real time si è deciso di utilizzare la libreria Socket.io sia lato client che lato server. La libreria lavora tramite canali di comunicazione su cui vengono inviati dei messaggi. I messaggi (o eventi), vengono intercettati dal destinatario solamente se nel canale del ricevente vi è presente un listener agganciato allo specifico evento descritto nel messaggio. Nell'evento a cui è agganciato il listener sia uguale all'evento specificato nel messaggio, verrà eseguito l'handler associato.

Per l'invio di messaggi si utilizza la funzione di emit, che consente anche di associare un corpo all'evento del messaggio, consentendo di inserirne così al suo interno le informazioni di interesse.

```

1     socket.emit("chatMessage", {username: myUsername , roomName:
2                     roomName , message: message})

```

Per la ricezione di messaggi, è necessario associare alla socket un listener che intercetta uno specifico evento:

```

1     socket.on("chatMessage", (data) => {
2         socket.broadcast.to(data.roomName).emit("receiveChatMessage"
3             " , {username: data.username , text: data.message})
4     })

```

In questo caso la socket lato server rimane in ascolto dell'evento “chatMessage” e una volta ricevuto l'evento decide di propagare in broadcast ad una specifica room l'informazione ricevuta. Con room si intende un canale arbitrario nella quale la socket può eseguire operazioni di join e leave. Nel caso mostrato sopra il dato viene propagato a tutte le socket facenti parte della room eccezion fatta per il mittente dell'evento.

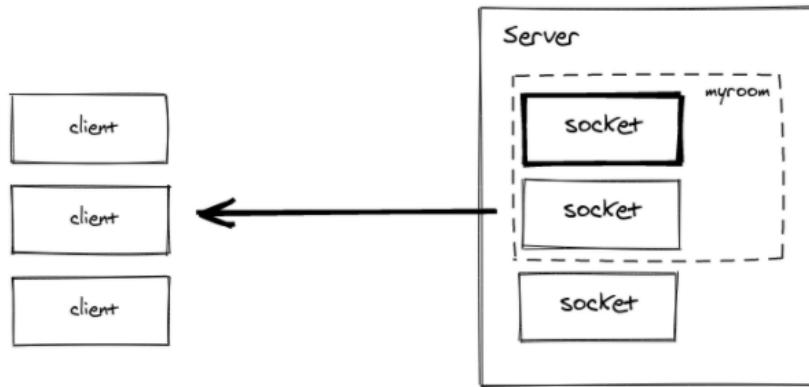


Figura 5.1: socket.io broadcast in room

Inoltre non è escluso data una socket e un suo handler, di poter sfruttare il dispatcher al suo interno, come ad esempio viene mostrato nella Fig 5.1.

```

1   socket.on("friendRequest", (data) => {
2     store.dispatch({
3       type: SET_FRIEND_USERNAME,
4       payload: data
5     })
6   });

```

# Capitolo 6

## Test

Le funzionalità del sistema sono state testate dal team su vari Browser di riferimento (principalmente Chrome, ma anche Edge e Firefox) con l'obiettivo di verificare e garantire la correttezza dell'applicazione e la sua portabilità. In particolare, i test sono stati atti a verificare che funzionalità non semplici come ad esempio l'utilizzo della chat real-time e la sincronizzazione dei player. Anche le API sviluppate lato server sono state testate in modo da garantire il loro corretto funzionamento. Si è inoltre sottoposto il sistema all'euristica di Nielsen, per avere un controllo qualitativo sull'usabilità dell'interfaccia, venendo a capo delle seguenti considerazioni:

- Controllo e Libertà: sono state ridotte al minimo le operazioni necessarie a svolgere un determinato compito in modo da ridurre la complessità totale dell'utilizzo dell'applicazione.
- Aiuto Utente: vi sono presenti messaggi informativi per gli utenti che commettono errori nell'utilizzo dell'app, come ad esempio le notifiche o le label di errore quando si immette un'indirizzo mail non valido.
- Prevenzione Errori: il sistema cerca il più possibile di evitare situazioni che portano l'utente a commettere errori. L'utente è guidato da procedure facilmente intuibili, e come detto in precedenza vengono mostrati messaggi di errore nel caso in cui l'utente non esegua correttamente certi task.
- Riconoscimento più che ricordo: le immagini e i bottoni presenti sulla piattaforma cercano di essere il più autoesplicative possibili, inoltre si è reso lo stile di ogni pagina uniforme in tutto il sistema.
- Visibilità stato del sistema: si è cercato di gestire le latenze presenti nel sistema in modo che non risultino dannose per il feedback visivo dell'utente.
- Corrispondenza sistema e mondo reale: i termini e le icone utilizzate nel sistema risultano di facile comprensione.

- Consistenza e Standard: i pulsanti presentano caratteristiche visive simili e la piattaforma presenta una gamma di colori ben distinta e utilizzata in modo uniforme all'interno del sistema in modo da rivolgere l'attenzione dell'utente verso gli elementi più importanti delle varie schermate.
- Estetica e progettazione minimalistica: viene utilizzato il principio chiave KISS, con poche funzionalità disponibili per l'utente in ogni pagina, ma ben distinte e distinguibili.
- Flessibilità ed efficienza: i menù sono resi estremamente facili da consultare ed utilizzare, il sistema offre funzionalità semplici per cui non si è ritenuto necessario introdurre scorciatoie o modalità di uso alternative.
- Documentazione: vista la semplicità delle azioni e dell'utilizzo generale del sistema dovuto a scelte di design ben precise (minimalist design) si è deciso che la fornitura di una documentazione non risultasse utile all'utente.

I test di usabilità sono stati effettuati durante tutto lo sviluppo del sistema. Gli utenti hanno testato il sistema dalla prima prototipazione fino alla terminazione dello stesso, fornendo feedback utili e necessari per la correzione o la modifica di funzionalità in modo da rendere l'esperienza d'uso il più godibile possibile.

# Capitolo 7

# Deployment

## 7.1 Installazione

Per installare l'applicativo è necessario seguire le istruzioni sottoelencate:

- 1 **Clonazione repository:** git clone <https://github.com/andreavaienti/MovieParty.git>.
- 2 **Tramite una Shell Bash spostarsi all'interno della cartella dell'e-laborato**
- 3 **Eseguire i seguenti comandi:**
  - 3.1 cd ./movieparty-app/
  - 3.1 install
  - 3.1 cd ..
  - 3.1 cd ./movieparty-server/
  - 3.1 npm install

Una volta che le seguenti istruzioni sono state effettuate, il software è correttamente installato e pronto per la messa in funzione.

## 7.2 Messa in funzione

Per eseguire l'applicativo è necessario avviare il frontend ed il backend:

- **Esecuzione frontend:**
  - 1 Recarsi all'interno della cartella *movieparty-app*

**2** Eseguire il comando `npm start`

- **Esecuzione backend:**

**1** Recarsi all'interno della cartella *movieparty-server*

**2** Eseguire il comando `npm run server`

Ora è possibile collegarsi alla piattaforma tramite un browser: `http://localhost:3000/`

### **7.3 Conclusioni**

La realizzazione di questo progetto ha contribuito ad arricchire il bagaglio culturale delle tecniche acquisite durante il corso, dandoci la possibilità di porci nelle condizioni di mettere alla prova le nostre conoscenze in campo pratico nella realizzazione di funzionalità real-time e coinvolgendo tecnologie diversificate nella stessa soluzione. Nella realizzazione della piattaforma ha giocato un ruolo importante l'utilizzo di React, tecnologia inizialmente sconosciuta, che dopo una prima fase di studio, ha contribuito nel migliorare l'organizzazione del codice e ci ha permesso di conoscere un nuovo tool di supporto alla programmazione lato Client. Oltre a React anche altre tecnologie sono state scoperte dal team, prima poco utilizzate o addirittura sconosciute come ad esempio Socket.io e Redux. Nel complesso, all'interno del team non sono state riscontrate grosse problematiche relative alla pianificazione degli Sprint e alla metodologia di sviluppo. Anzi, l'intero team considera che tale progetto sia stata un'esperienza altamente formativa, permettendoci di imbattersi in aspetti di design, progettazione e sviluppo di un applicazione web complessa. Riteniamo la conoscenza dello stack MEAN (e delle sue varianti) una nozione fondamentale per lo sviluppo di applicazioni WEB, perché come da Lei detto a lezione, difficilmente riusciremo a tornare indietro ai vecchi approcci.