

Chillstream  
Applicazioni e Servizi Web

Gianmaria Casamenti - 0001136953 {gianmaria.casamenti@unibo.it}  
Luca Pasini - 0001140963 {luca.pasini8@unibo.it}

Maggio 2025

# Contents

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Requisiti</b>	<b>5</b>
2.1	Requisiti di business . . . . .	5
2.2	Requisiti utente . . . . .	5
2.3	Requisiti funzionali . . . . .	6
2.4	Requisiti non funzionali . . . . .	8
2.5	Requisiti di implementazione . . . . .	9
<b>3</b>	<b>Design</b>	<b>10</b>
3.1	Design dell'architettura del sistema . . . . .	10
3.1.1	Componenti funzionali (Vue Composition API) . . . . .	11
3.1.2	Vue Template e Single File Components . . . . .	11
3.2	Metodologie di sviluppo . . . . .	11
3.2.1	Metodologia di design adottata . . . . .	11
3.2.2	Modalità di collaborazione e gestione del progetto . . . . .	12
3.3	Mockup iniziale . . . . .	13
3.3.1	Fase iniziali del sistema . . . . .	13
3.3.2	Mockup delle funzionalità principali . . . . .	16
3.4	Target User Analysis . . . . .	20
3.5	StoryBoard . . . . .	22
3.5.1	StoryBoard della registrazione . . . . .	22
3.5.2	StoryBoard delle funzionalità principali . . . . .	23
3.5.3	Storyboard delle funzionalità accessibili dalla Navbar . . . . .	27
3.5.4	Storyboard: Sezione Admin . . . . .	29
<b>4</b>	<b>Tecnologie</b>	<b>32</b>
4.1	Docker . . . . .	32
4.1.1	Shared Docker Network . . . . .	33
4.2	REST API . . . . .	34
4.3	Two-Way Binding . . . . .	35
4.4	Axios . . . . .	36
4.5	CORS . . . . .	37
4.6	SocketIO . . . . .	37

4.6.1	Comunicazione in tempo reale per stanza . . . . .	38
<b>5</b>	<b>Codice</b>	<b>41</b>
5.1	Utilizzo della sintassi <script setup> in Vue.js . . . . .	41
5.1.1	Confronto con la sintassi tradizionale . . . . .	41
5.2	Components di Vue.js . . . . .	42
<b>6</b>	<b>Test</b>	<b>44</b>
<b>7</b>	<b>Deployment</b>	<b>47</b>
<b>8</b>	<b>Conclusione e sviluppi futuri</b>	<b>48</b>
8.0.1	Sviluppi futuri . . . . .	48

# Chapter 1

## Introduzione

Il progetto **Chillstream** nasce dall'esigenza di un Cinema che desidera ampliare la propria offerta, sviluppando una Web App che permetta agli utenti di registrarsi, guardare film e interagire tra loro in tempo reale, condividendo pensieri ed emozioni.

Chillstream è un'applicazione web che consente di accedere a un'ampia libreria di contenuti multimediali, selezionare un film e guardarla in compagnia di altri utenti online. La sua funzionalità principale è la riproduzione condivisa dei film presenti sulla piattaforma, ma l'app offre anche diverse funzioni aggiuntive, come la possibilità di chattare in tempo reale con gli altri spettatori dello stesso film.

Oltre alla riproduzione dei contenuti e alla chat interattiva, Chillstream offre agli utenti registrati la possibilità di personalizzare la propria esperienza mediante la creazione di più profili, ciascuno identificato da un *nickname* e un'immagine del profilo.

Ogni profilo dispone delle seguenti sezioni:

- **Continua a guardare:** una lista di film iniziati ma non ancora terminati.
- **MyList:** una raccolta personalizzata di contenuti che l'utente desidera guardare in un secondo momento.
- **Notifiche:** un sistema che segnala l'uscita di nuovi contenuti disponibili sulla piattaforma.

Il sistema nella sua intezza invece include le seguenti funzionalità:

- Sistema di registrazione attraverso dati personali
- Sistema per la gestione sicura delle credenziali d'accesso: solo gli utenti autenticati possono accedere ai servizi offerti dalla piattaforma.
- Sistema per la gestione di informazioni e servizi in tempo reale.
- Sistema per la gestione dei profili personali associati ad ogni utente.

- Sistema di notifiche per segnalare le nuove uscite e aggiornamenti rilevanti.
- Sistema con accesso unico all'admin del sistema per gestire le uscite di nuovi film, casting e messaggi di notifica.

# **Chapter 2**

# **Requisiti**

Questo capitolo ha come scopo quello descrivere dettagliatamente tutti i requisiti del software implementato.  
E' 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. Possibilità di visionare contenuti multimediali presenti nella raccolta del sito insieme ad altri utenti connessi.
2. Possibilità di comunicare tramite chat in tempo reale durante una sessione di visione.
3. Accesso da parte dell'amministratore a un pannello di controllo per la gestione della piattaforma.

## **2.2 Requisiti utente**

In particolare, l'utente può usufruire dei seguenti aspetti:

### **2.2.1 Autenticazione al sito**

- 2.2.1.1 Registrazione di un nuovo utente che non dispone delle credenziali;
- 2.2.1.2 Login di un utente che dispone di credenziali di accesso;
- 2.2.1.3 Logout dell'utente dalla sessione corrente.

### **2.2.2 Gestione dei profili**

- 2.2.2.1 Creazione di un nuovo profilo;

- 2.2.2.2 Modifica dell'immagine del profilo;
- 2.2.2.3 Modifica del nickname del profilo;
- 2.2.2.4 Rimozione di un profilo;
- 2.2.2.5 Logout di un profilo.

#### **2.2.3 Interazione con i contenuti multimediali**

- 2.2.3.1 Selezione di un film dal catalogo;
- 2.2.3.2 Accesso rapido ai contenuti in "Continua a guardare";
- 2.2.3.3 Aggiunta e gestione di film nella sezione "MyList";
- 2.2.3.4 Interazione nella live-chat durante la visione condivisa;
- 2.2.3.5 Visualizzazione del cast del film;
  - 2.2.3.5.1 Visione delle informazioni relative agli attori;
  - 2.2.3.5.2 Visualizzazione di tutti i film a cui l'attore ha partecipato.
- 2.2.3.6 Aggiunta di recensioni per ogni film.

#### **2.2.4 Gestione notifiche**

- 2.2.4.1 Visualizzazione delle notifiche ricevute;
- 2.2.4.2 Notifiche da parte dell'admin;
- 2.2.4.3 Notifiche sulle nuove uscite;
- 2.2.4.4 Possibilità di segnare una notifica come "visualizzata".

#### **2.2.5 Accesso alla sezione Help**

- 2.2.5.1 Ricerca e consultazione degli errori comuni.

### **2.3 Requisiti funzionali**

Il sistema prodotto dovrà:

#### **2.3.1 Gestire l'autenticazione degli utenti al sito:**

- 2.3.1.1 Registrazione di utenti che non possiedono credenziali:
  - 2.3.1.1.1 Richiesta obbligatoria dei seguenti campi: nome, cognome, email, password;
  - 2.3.1.1.2 Verifica dell'unicità dell'email;
  - 2.3.1.1.3 Verifica semantica dell'email;
  - 2.3.1.1.4 Hashing della password per la memorizzazione nel database.
- 2.3.1.2 Login di utenti che possiedono credenziali:
  - 2.3.1.2.1 Richiesta di email e password per l'autenticazione;
  - 2.3.1.2.2 Verifica dell'esistenza dell'utente tramite email;

2.3.1.2.3 Verifica della correttezza della password inserita.

2.3.1.3 Logout degli utenti autenticati al sistema.

2.3.2 Gestire i profili utente:

2.3.2.1 Creazione di un nuovo profilo:

2.3.2.1.1 Obbligatoria in caso di prima iscrizione;

2.3.2.1.2 Richiesta obbligatoria di un nickname e di un'immagine, altri-  
menti assegnati di default.

2.3.2.2 Modifica di un profilo esistente;

2.3.2.3 Eliminazione di un profilo, con successiva richiesta di selezione di un  
altro profilo attivo;

2.3.2.4 Logout del profilo dalla sessione corrente.

2.3.3 Gestire le notifiche associate ad eventi:

2.3.3.1 Notifica di nuove uscite (film);

2.3.3.2 Notifica di messaggi da parte dell'amministratore.

2.3.4 Gestire le azioni connesse ai contenuti multimediali:

2.3.4.1 Avvio della riproduzione e accesso alla pagina contenente la live-chat;

2.3.4.2 Controlli sulla riproduzione:

2.3.4.2.1 Mettere in pausa il film;

2.3.4.2.2 Avanzare nella riproduzione;

2.3.4.2.3 Attivare la modalità muto;

2.3.4.2.4 Attivare i sottotitoli in più lingue.

2.3.4.3 Aggiunta del film alla lista "Keep Watching" con salvataggio del min-  
utaggio corrente;

2.3.4.4 Aggiunta del film alla lista personale "MyList";

2.3.4.5 Gestione della live-chat:

2.3.4.5.1 Visualizzazione del messaggio di accesso degli utenti;

2.3.4.5.2 Invio di messaggi da parte degli utenti;

2.3.4.5.3 Ricezione dei messaggi da parte di tutti i partecipanti.

2.3.4.6 Gestione delle recensioni:

2.3.4.6.1 Aggiunta delle recensioni tramite data binding;

2.3.4.6.2 Visualizzazione immediata sul client senza necessità di notificare  
il server.

2.3.5 Gestire le informazioni relative agli attori:

2.3.5.1 Visualizzazione del cast associato ad ogni film;

- 2.3.5.2 Visualizzazione delle informazioni personali (nome, cognome, data di nascita) di ciascun attore;
  - 2.3.5.3 Visualizzazione dei film a cui l'attore ha partecipato (dal database).
- 2.3.6 Gestire il pannello di controllo dell'amministratore:
- 2.3.6.1 Login sicuro da parte dell'admin con verifica delle credenziali e hashing della password.
  - 2.3.6.2 Gestione della sezione dei contenuti:
    - 2.3.6.2.1 Aggiunta di una nuova entità *film*, richiedendo la compilazione dei campi: titolo, anno, valutazione, immagine, video, descrizione, e casting;
    - 2.3.6.2.2 Modifica di un film esistente con inserimento dei campi da aggiornare;
    - 2.3.6.2.3 Eliminazione di un film specificando il titolo.
  - 2.3.6.3 Gestione della sezione degli attori:
    - 2.3.6.3.1 Aggiunta di una nuova entità *attore*, richiedendo i campi: nome, cognome e data di nascita;
    - 2.3.6.3.2 Modifica di un attore esistente (es. aggiornamento della data di nascita);
    - 2.3.6.3.3 Eliminazione di un attore tramite nome e cognome.
  - 2.3.6.4 Gestione della sezione notifiche:
    - 2.3.6.4.1 Invio di un messaggio a tutti gli utenti registrati.

## 2.4 Requisiti non funzionali

I seguenti requisiti non funzionali descrivono le caratteristiche qualitative che il sistema dovrà garantire per assicurare affidabilità, usabilità e prestazioni adeguate.

### 2.4.1 Sicurezza:

- 2.4.1.1 Le password degli utenti devono essere criptate mediante un algoritmo sicuro e memorizzate in forma non reversibile.

### 2.4.2 Prestazioni:

- 2.4.2.1 Il tempo di risposta del server non deve superare i 2 secondi nel 95% delle richieste.
- 2.4.2.2 Lo streaming video deve essere fluido su connessioni con velocità di almeno 10 Mbps.

### 2.4.3 Usabilità:

2.4.3.1 L'interfaccia utente deve essere responsive e compatibile con dispositivi desktop, tablet e mobile.

2.4.3.2 Il sistema deve risultare facile da utilizzare anche da utenti senza competenze tecniche.

#### 2.4.4 **Manutenibilità:**

2.4.4.1 Il codice deve essere modulare e documentato per facilitare interventi di aggiornamento o estensione.

#### 2.4.5 **Compatibilità:**

2.4.5.1 Il sistema deve essere compatibile con le ultime due versioni dei principali browser: Chrome, Firefox, Edge e Safari.

2.4.5.2 L'applicazione non deve richiedere l'installazione di plugin esterni per funzionare correttamente.

## 2.5 Requisiti di implementazione

2.5.1 Sviluppo dell'applicazione con stack MEVN.

2.5.2 Utilizzo della libreria Socket.IO per la realizzazione di comunicazione in real-time.

2.5.3 Utilizzo di API esterna per la visione dei contenuti.

# Chapter 3

## Design

### 3.1 Design dell'architettura del sistema

L'applicativo è stato costruito seguendo un'architettura a microservizi, adottando principalmente lo stack **MEVN**, ovvero:

- **MongoDB** – database NoSQL per la memorizzazione dei dati;
- **Express.js** – framework backend su Node.js per la gestione delle API;
- **Vue.js** – framework frontend reactive per la costruzione dell'interfaccia utente;
- **Node.js** – runtime JavaScript per l'esecuzione lato server.

Lo stack MEVN è una variante del ben noto stack MEAN, dove Angular viene sostituito da Vue.js, mantenendo però l'intero flusso di sviluppo basato su JavaScript e JSON. Questo approccio ci ha permesso di costruire un sistema a 3 livelli (frontend, backend, database) con una separazione chiara delle responsabilità. Nel caso specifico del progetto, l'architettura è stata ulteriormente estesa in ottica **microservizi**, introducendo anche componenti sviluppati in linguaggi diversi:

- **Frontend:** interamente sviluppato con **Vue.js**, in combinazione con JavaScript e il client HTTP **Axios** per la comunicazione con i servizi backend.
- **Backend:** composto da più microservizi:
  - Un **microservizio per l'autenticazione** sviluppato con **Node.js** ed **Express.js**, che mantiene l'allineamento con lo stack MEVN per le funzionalità principali;
  - Due **microservizi aggiuntivi sviluppati con Python e Flask**, che non verranno approfonditi in questa relazione, ma che completano la logica di business e supportano l'architettura modulare.

Questa scelta ha permesso di sfruttare i vantaggi di una **microservice architecture eterogenea**, dove ogni servizio è stato realizzato con la tecnologia più adatta al suo ambito specifico, mantenendo una comunicazione coerente grazie a interfacce RESTful.

Infine, la presenza di **MongoDB**, grazie alla sua struttura basata su documenti JSON, ha semplificato l'integrazione dei dati tra i vari livelli dell'applicazione e facilitato le operazioni di persistenza e scambio di informazioni tra microservizi.

### 3.1.1 Componenti funzionali (Vue Composition API)

Nel presente progetto è stata adottata la **Composition API** di Vue.js per la realizzazione dei componenti funzionali. Questo approccio, introdotto a partire da Vue 3, consente una maggiore flessibilità rispetto alla tradizionale Options API, favorendo il riutilizzo della logica di business tra più componenti.

La Composition API permette di definire lo stato reattivo e le funzioni di un componente all'interno della funzione `setup()`, con un miglior supporto alla Type Safety e una maggiore organizzazione del codice.

Tra le funzionalità più utilizzate:

- `reactive()` e `ref()`: per definire variabili reattive;
- `watch()`: per monitorare cambiamenti su valori specifici;
- `onMounted()`: per eseguire operazioni al montaggio del componente;
- `computed()`: per definire proprietà derivate in modo efficiente.

Tali strumenti hanno permesso di strutturare i componenti in modo modulare e manutenibile, separando chiaramente le responsabilità di ciascun blocco.

### 3.1.2 Vue Template e Single File Components

L'interfaccia utente è stata costruita utilizzando la sintassi **Vue Template**, racchiusa all'interno dei **Single File Components (.vue)**. Questo approccio consente di mantenere fortemente coeso il markup (HTML), lo script (JavaScript) e lo stile (CSS) all'interno dello stesso file, favorendo la leggibilità e l'organizzazione del progetto.

L'utilizzo della sintassi dichiarativa offerta dai template ha permesso un rapido sviluppo delle viste e una facile integrazione con lo stile CSS modulare.

## 3.2 Metodologie di sviluppo

### 3.2.1 Metodologia di design adottata

Per la progettazione dell'applicazione è stata adottata la metodologia **User Centered Design (UCD)**, incentrata sull'utente come elemento chiave di ogni fase del processo di sviluppo. Tale scelta è stata dettata dalla volontà del team

di focalizzarsi, sin dalle fasi iniziali del progetto, sulla **Human-Computer Interaction (HCI)**, con l'obiettivo di ottenere un'interfaccia altamente usabile e migliorare l'esperienza utente (*User Experience*).

L'intero processo progettuale si è concentrato sull'applicazione finale, cercando di rispondere concretamente ai bisogni e alle aspettative degli utenti target, definiti attraverso la costruzione di *Personas* — profili utente fintizi ma realistici, rappresentativi del pubblico di riferimento.

Per ottimizzare ulteriormente la qualità dell'esperienza utente, sono stati applicati i seguenti principi fondamentali:

- **Responsive Design:** tutte le pagine web sono state realizzate in modo da adattarsi automaticamente alle dimensioni e risoluzioni dei dispositivi utilizzati per la visualizzazione;
- **Mobile First:** l'interfaccia è stata progettata inizialmente per dispositivi a risorse limitate (smartphone e tablet), per poi essere estesa e adattata ai dispositivi desktop;
- **KISS (Keep It Simple, Stupid):** le interfacce sono state concepite con un design semplice ed essenziale, contenente solo gli elementi funzionali più rilevanti e facilmente accessibili per l'utente.

### 3.2.2 Modalità di collaborazione e gestione del progetto

Essendo il team composto da due persone, è stata adottata una modalità di lavoro flessibile, ispirata a principi snelli e collaborativi simili a quelli del modello “*two-pizza team*”, in cui un piccolo gruppo è in grado di lavorare in autonomia con comunicazione costante ed efficace.

Non sono stati pianificati sprint formali o attività di Sprint Review/Planning, ma il team ha comunque mantenuto una forte organizzazione interna.

I membri si sono incontrati frequentemente, sia in presenza che online, sfruttando la possibilità di collaborare direttamente su codice e decisioni progettuali. Questa modalità ha permesso di ridurre i tempi di coordinamento e favorire una maggiore reattività alle esigenze progettuali emergenti.

Durante tutto lo sviluppo, si è mantenuta una comunicazione continua, basata su confronto immediato, condivisione di obiettivi a breve termine e rapida risoluzione dei problemi. Ogni nuova funzionalità è stata progettata, sviluppata e testata in modo incrementale, seguendo una logica iterativa, seppur priva di formalismi tipici dell'Agile canonico.

Questo approccio informale ma efficace ha permesso al team di mantenere un buon ritmo di sviluppo, garantendo coerenza tra le parti del sistema e favorendo una forte intesa tra i membri, elemento fondamentale in progetti condotti da gruppi ristretti.

### 3.3 Mockup iniziale

Per lo sviluppo dell’interfaccia utente dell’applicazione, è stata adottata una filosofia di design semplice e intuitiva, volta a garantire un’esperienza d’uso chiara e accessibile. Nella fase preliminare di progettazione, si è fatto uso di mockup per delineare visivamente la struttura dell’applicativo e anticiparne le funzionalità principali.

A tale scopo, è stato utilizzato il tool **Balsamiq**, scelto per la sua efficienza nella realizzazione di *wireframe* rapidi e orientati alla fase concettuale. Questi prototipi iniziali hanno rappresentato un valido supporto per definire l’organizzazione delle viste, la disposizione degli elementi e le principali interazioni previste.

L’applicazione è stata progettata principalmente per un utilizzo su dispositivi **desktop**, in quanto tale piattaforma è risultata la più adatta alla fruizione di contenuti audiovisivi accompagnati da funzionalità di chat in tempo reale. Tuttavia, è stata prestata attenzione anche all’adattabilità dell’interfaccia per **tablet** e **smartphone**, al fine di offrire un’esperienza d’uso soddisfacente anche su schermi di dimensioni ridotte.

In seguito alla realizzazione dei primi sketch e alla raccolta di feedback qualitativi da parte di utenti test, l’interfaccia è stata progressivamente affinata, giungendo a una soluzione in grado di conciliare chiarezza visiva, coerenza stilistica e funzionalità.

Nella fase iniziale di progettazione dell’interfaccia utente sono stati realizzati dei mockup per illustrare il flusso di autenticazione e la selezione del profilo utente. Di seguito vengono descritte le principali schermate in base alla fase funzionale e al tipo di dispositivo.

#### 3.3.1 Fase iniziali del sistema

##### Fase di autenticazione (desktop)

La fase di autenticazione si articola in tre viste principali:

- **Homepage (Get Started):** l’utente visualizza una pagina introduttiva con un breve messaggio descrittivo dell’applicazione e la possibilità di iniziare la registrazione o accedere tramite il pulsante **Sign In**.
- **Login:** viene richiesto all’utente l’inserimento delle credenziali (**username** e **password**) per accedere alla piattaforma.
- **Registrazione:** in alternativa, l’utente può creare un nuovo account compilando un modulo che include campi come **nome**, **cognome**, **email**, **password**, **data di nascita** e **metodo di pagamento**.

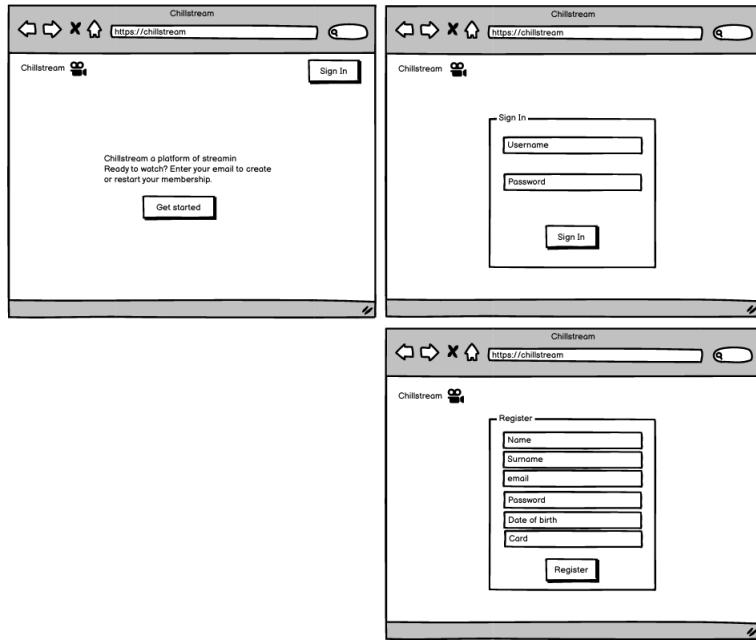


Figure 3.1: Mockup pagine di autenticazione

### Fase di selezione del profilo (desktop)

Dopo una corretta autenticazione, l’utente accede a una schermata di selezione del profilo. In questa fase è possibile:

- Selezionare un profilo esistente tra quelli associati all’account;
- Creare un nuovo profilo, scegliendo un **nickname** e un’icona avatar tra quelle disponibili.

Questa fase fa parte integrante del processo di accesso, in quanto consente la personalizzazione dell’esperienza utente.

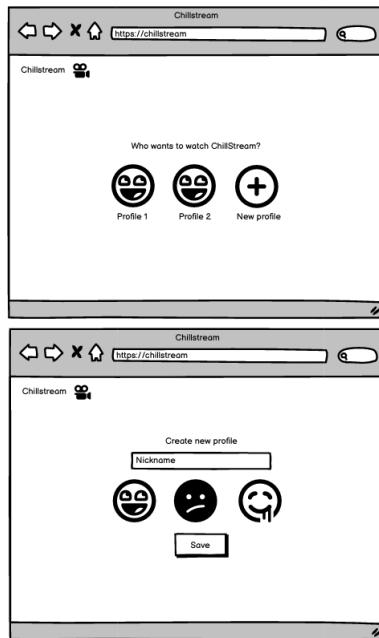


Figure 3.2: Mockup pagine selezione/creazione dei profili

### Mockup mobile

Tutte le fasi precedenti sono state replicate anche in versione mobile, in linea con i principi di responsive design e mobile first.

- Le schermate di **homepage**, **login** e **selezione profilo** sono state adattate al formato smartphone mantenendo la stessa struttura funzionale ma ottimizzata in termini di layout, proporzioni e dimensionamento dei componenti.
- L'interfaccia è stata semplificata visivamente per garantire una fruizione efficace anche su schermi ridotti, mantenendo coerenza con la versione desktop.

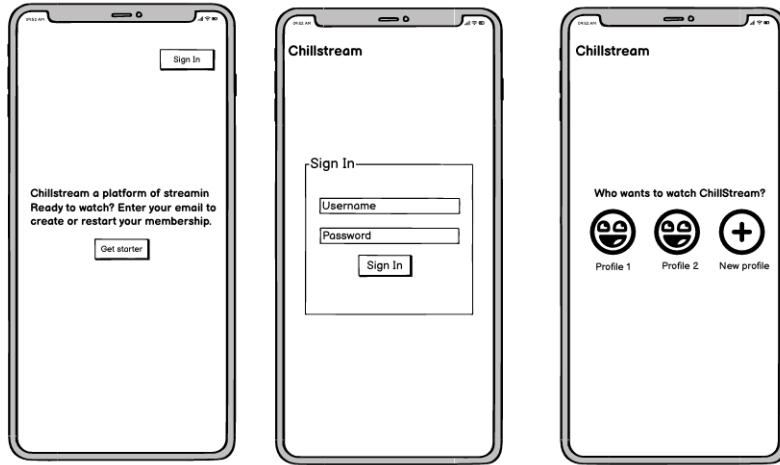


Figure 3.3: Mockup pagine autenticazione e selezione profilo in vista mobile

### 3.3.2 Mockup delle funzionalità principali

Di seguito vengono illustrate le schermate principali dell'applicazione progettata, a partire dalla homepage post-login, fino alla visione del contenuto e alle funzionalità interattive associate.

#### Homepage

Dopo aver effettuato l'accesso e selezionato il profilo utente, l'utente viene reindirizzato alla homepage, dove sono visibili tre sezioni principali:

- **Movie:** contiene una selezione di contenuti suggeriti o in evidenza;
- **Continue to watching:** mostra i film iniziati ma non ancora completati;
- **MyList:** lista personalizzata dei contenuti salvati dall'utente.

L'interfaccia è progettata con caroselli orizzontali navigabili, sia in versione desktop che mobile.

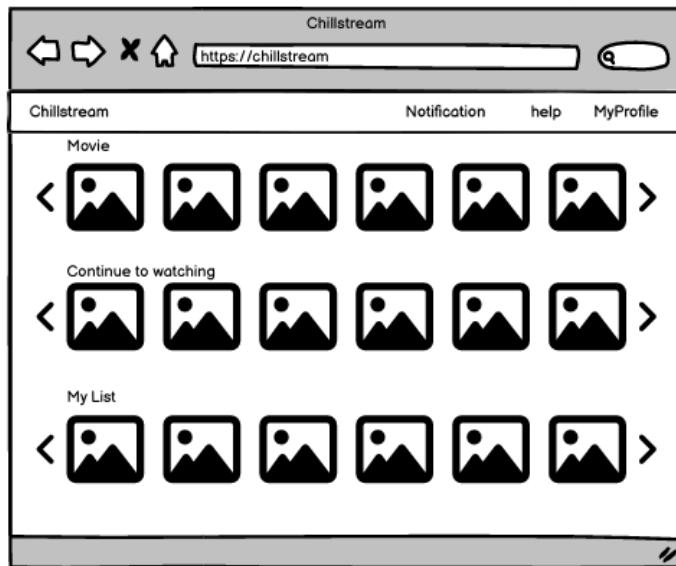


Figure 3.4: Mockup homepage

### Dettagli del film

Selezionando un film dalla homepage, si accede alla schermata di dettaglio. Essa presenta:

- Una miniatura del film, titolo, anno, valutazione e genere;
- Un breve riepilogo descrittivo;
- Due pulsanti: **Play** per avviare la visione, e **MyList** per salvare il contenuto;
- Due tab interattivi:
  - **Cast**: elenca gli attori partecipanti con nome e cognome;
  - **Review**: visualizza e consente l'inserimento di recensioni da parte degli utenti.

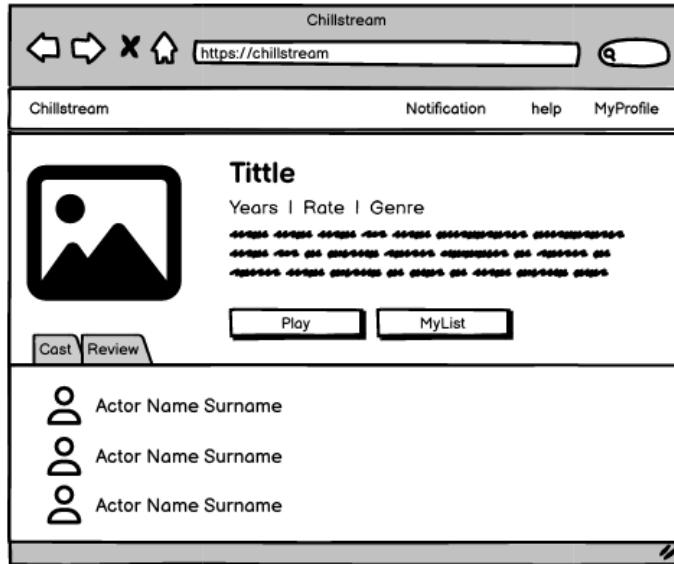


Figure 3.5: Mockup pagina dettagli di un film

### Visione del film e chat (desktop e mobile)

Una volta avviata la visione del film, l'utente accede alla modalità player. L'interfaccia comprende:

- Un lettore video con controlli standard (play, volume, sottotitoli);
- Una **live chat** posizionata lateralmente (desktop) o sotto il player (mobile), che consente l'invio e la ricezione di messaggi tra tutti gli utenti che stanno guardando lo stesso contenuto in tempo reale;
- Un campo di input e pulsante per inserire nuovi messaggi.

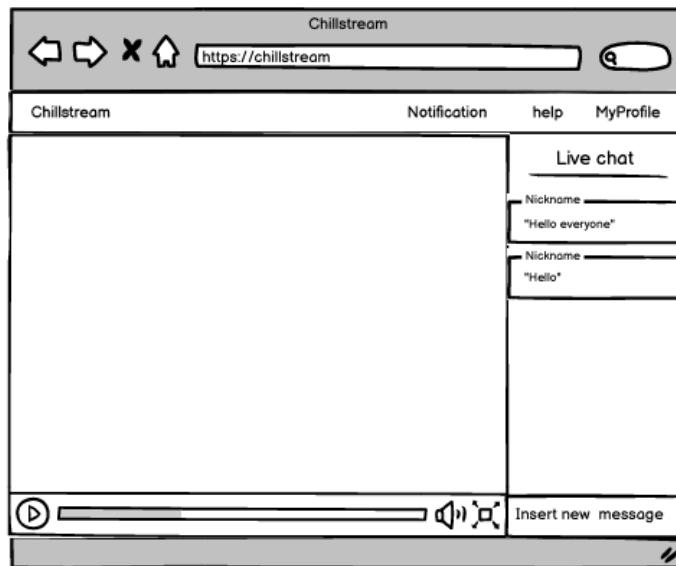


Figure 3.6: Mockup pagina con la visione di un film

### Responsive design e layout mobile

Tutte le funzionalità sopra descritte sono state progettate in ottica **mobile first**, come dimostrato nei mockup per smartphone:

- La homepage mobile mantiene lo stesso schema di contenuti della versione desktop, adattando gli elementi a uno scorrimento verticale;
- Le schermate di dettaglio e visione sono state semplificate per adattarsi a schermi ridotti, mantenendo accessibilità e facilità d'uso;
- La live chat è collocata nella parte inferiore per favorire un uso simultaneo durante la visione.

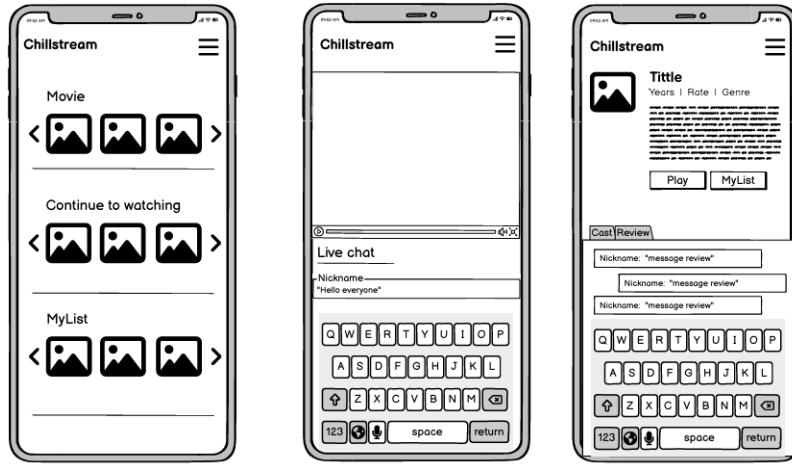


Figure 3.7: Mockup delle pagine principali in mobile

### 3.4 Target User Analysis

Lo sviluppo dell'applicazione è stato guidato da un'attenta analisi del target di riferimento, fondamentale per allineare le funzionalità del sistema alle reali esigenze degli utenti. In questo caso, l'applicazione è stata commissionata da un **cinema locale** con l'obiettivo di ampliare il proprio business, estendendo l'esperienza cinematografica anche all'ambiente digitale.

Il pubblico target è rappresentato da utenti interessati alla visione di contenuti multimediali in compagnia, con un forte orientamento alla condivisione dell'esperienza. A differenza di piattaforme che obbligano a coordinarsi tramite strumenti esterni (come Discord o Skype), Chillstream integra nativamente la possibilità di commentare e interagire tramite chat in tempo reale. L'accesso alla chat è libero e continuo: ogni utente può entrare o uscire dalla conversazione in qualsiasi momento, senza vincoli di sincronizzazione rispetto alla visione del contenuto.

L'applicazione offre così un ambiente unico dove visione e interazione convivono, semplificando l'esperienza dell'utente e centralizzando tutte le funzionalità in un'unica piattaforma.

L'analisi del pubblico è stata condotta secondo i seguenti criteri:

- **Contenuti specifici:** conoscere il proprio target consente di definire contenuti e messaggi più efficaci. Il tono comunicativo e le funzionalità offerte sono state pensate per un pubblico giovane e tecnologicamente attivo.
- **Ottimizzazione dell'interfaccia:** è stato considerato il comportamento degli utenti su diversi dispositivi (desktop, tablet e mobile), garantendo un'esperienza coerente e usabile in ogni contesto.

- **Focalizzazione sul potenziale:** si è deciso di indirizzare l'applicazione a un pubblico realmente interessato al servizio, piuttosto che disperdere risorse cercando di attrarre utenti non in target.
- **Strategia mirata:** la definizione di un pubblico ben delineato ha permesso di stabilire linee guida precise anche per future strategie di marketing, puntando su media e tecniche comunicative coerenti con il profilo dell'utente tipo.

A partire da queste considerazioni, sono stati definiti alcuni *personas*, ovvero personaggi fintizi creati per rappresentare in modo realistico gli utenti finali. Queste figure hanno aiutato il team a progettare soluzioni orientate ai bisogni reali e a mantenere il focus sull'esperienza utente durante tutte le fasi dello sviluppo.

#### **Persona 1: Marta – “La Social Streamer”**

- **Età:** 24 anni
- **Professione:** Studentessa magistrale in Comunicazione Digitale
- **Dispositivi preferiti:** Smartphone e laptop
- **Abitudini di visione:** Guarda serie TV e film in streaming ogni sera, spesso in compagnia virtuale di amici
- **Obiettivi:**
  - Condividere l'esperienza di visione con amici, anche a distanza
  - Scoprire nuovi contenuti tramite consigli della community
- **Frustrazioni:**
  - Difficoltà nel sincronizzare la visione con amici utilizzando piattaforme diverse
  - Mancanza di un luogo unico per chat e visione simultanea
- **Citazione:** “Vorrei poter guardare un film con i miei amici, commentarlo in tempo reale, senza dover usare mille app diverse.”

#### **Persona 2: Luca – “Il Cinefilo Locale”**

- **Età:** 38 anni
- **Professione:** Impiegato amministrativo
- **Dispositivi preferiti:** Smart TV e tablet
- **Abitudini di visione:** Appassionato di cinema d'autore e film indipendenti, frequenta il cinema locale settimanalmente

- **Obiettivi:**

- Accedere a contenuti esclusivi del cinema anche da casa
- Partecipare a discussioni e recensioni con altri appassionati

- **Frustrazioni:**

- Limitata disponibilità di film d'autore sulle piattaforme mainstream
  - Difficoltà nel trovare una community locale di cinefili online
- **Citazione:** “Mi piacerebbe poter vedere i film del mio cinema preferito anche da casa e discuterne con altri appassionati.”

## 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.

### 3.5.1 StoryBoard della registrazione

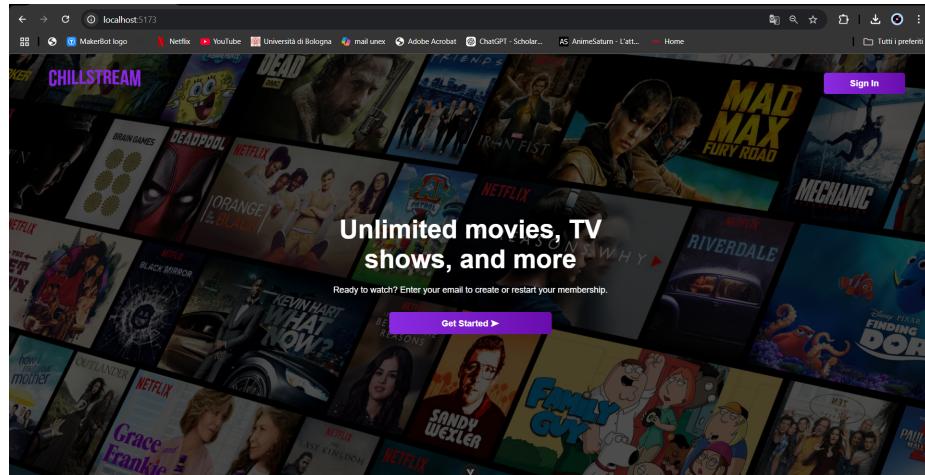


Figure 3.8: Pagina iniziale di Chillstream

Da questa schermata l'utente potrà accedere alla schermata dove registrarsi nel caso in cui non abbia ancora le credenziali per l'accesso al sito. Se l'utente durante la registrazione al sito, commette errori esso viene notificato visivamente, nello specifico si ha un warning visivo in alto a destra e di colore rosso per gli errori riguardanti l'inserimento di una mail già presente nel database oppure se il form non è stato compilato nella maniera corretta.

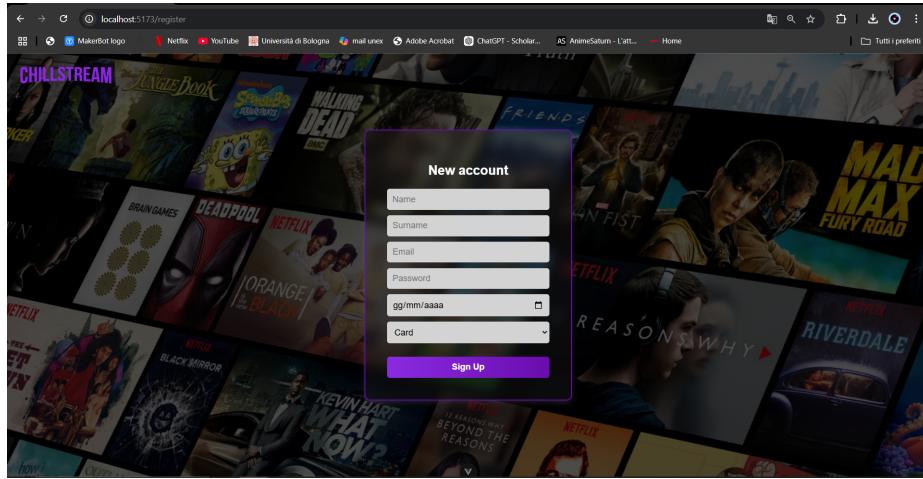


Figure 3.9: Pagina di registrazione di un nuovo utente

Nel caso in cui le credenziali siano già in possesso dalla schermata principale 3.8 è possibile accedere alla pagina di sign-in, anche in questo caso viene fatta una verifica che la password e la mail siano corrette e presenti nel database.

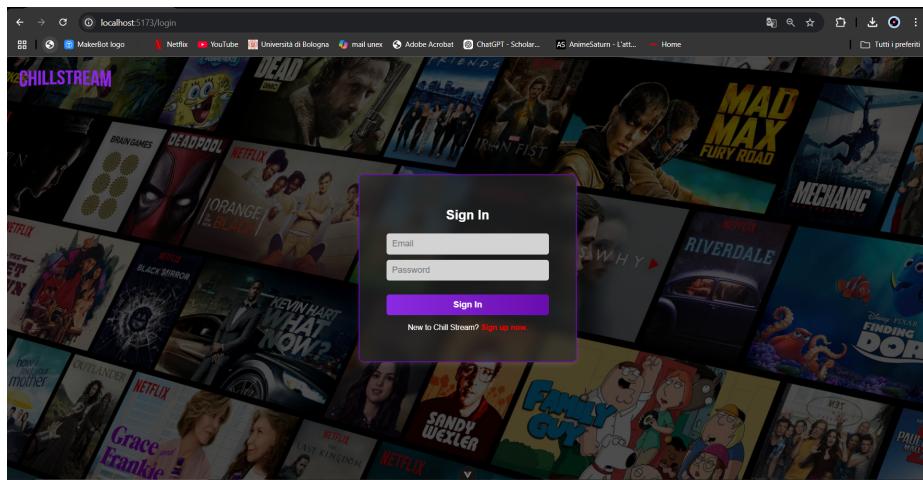


Figure 3.10: Pagina di sign-in di un utente già registrato

### 3.5.2 StoryBoard delle funzionalità principali

Lo storyboard seguente descrive il flusso delle principali funzionalità dell'applicazione Chillstream, illustrando l'esperienza utente dalla homepage alla visione del contenuto multimediale con chat attiva.

## Homepage utente

Dopo aver effettuato l'accesso e selezionato il proprio profilo, l'utente atterra sulla homepage dell'applicazione. Qui può visualizzare tre sezioni principali:

- **Trending Movies** – film popolari del momento;
- **Continue to Watching** – contenuti iniziati e da completare;
- **My List** – contenuti salvati manualmente dall'utente.

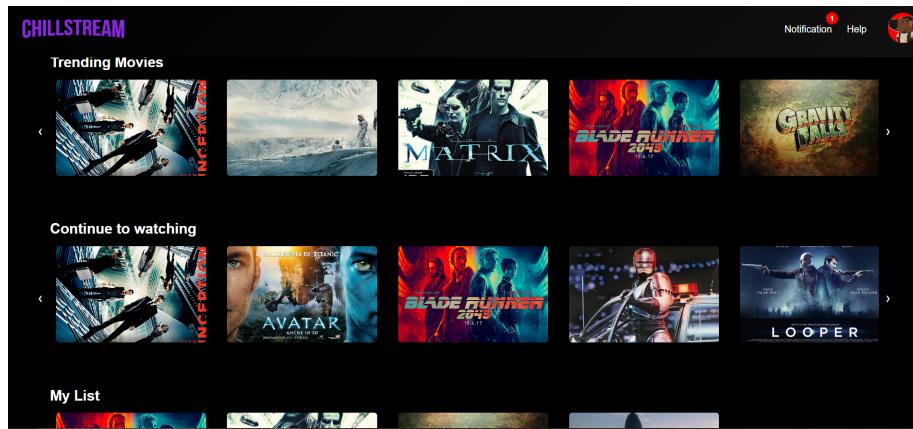


Figure 3.11: Pagina principale del sistema

## Dettagli del film selezionato

Se l'utente seleziona un film dalla homepage, accede alla pagina dei dettagli. Qui può:

- Leggere la sinossi del film;
- Visualizzare l'anno, il voto medio e il genere;
- Aggiungere il contenuto alla propria lista personale con **In My List**;
- Riprendere la visione se già iniziata, tramite il pulsante **Keep Watching**;
- Accedere ai tab **Cast** e **Reviews** per conoscere gli attori o leggere/lasciare recensioni.

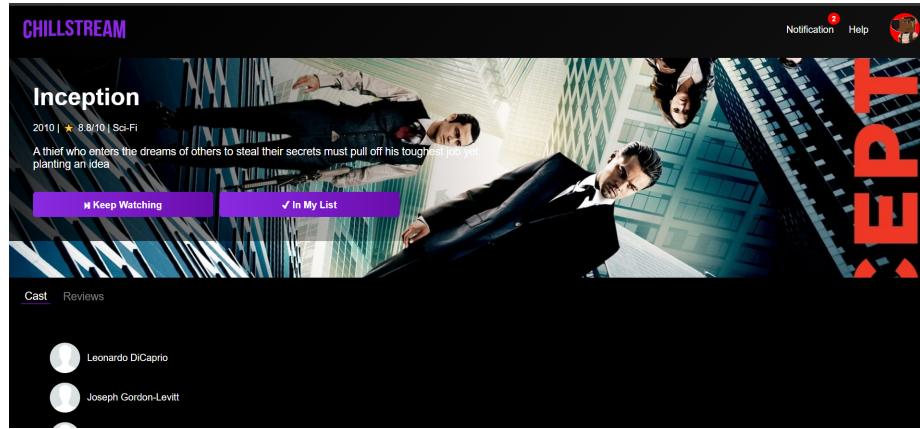


Figure 3.12: Pagina di interazione con i film

### Pagina di visione e chat real-time

Cliccando su Play oppure Keep watching in caso il sito sia già iniziato , viene avviato il player video. A fianco del film è presente la sezione **Live Chat**, dove l'utente può:

- Leggere i messaggi degli altri spettatori in tempo reale;
- Inviare messaggi di testo;
- Visualizzare eventi di sistema come l'ingresso di un nuovo utente nella chat.

Tutti i contenuti e le interazioni sono integrati in un'unica interfaccia.

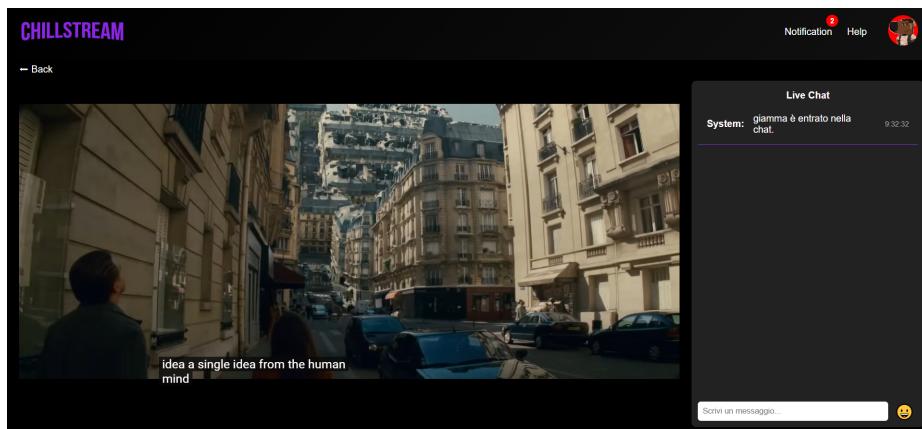


Figure 3.13: Pagina di visione di un contenuto

## Dettagli attore e filmografia

Cliccando sul nome di un attore nella sezione **Cast** della pagina film, l'utente accede a una pagina dedicata che presenta:

- **Informazioni personali** dell'attore (nome, cognome, data di nascita);
- **Filmografia**, ovvero l'elenco dei film presenti nella piattaforma a cui l'attore ha partecipato, cliccabili per accedere alla rispettiva scheda.

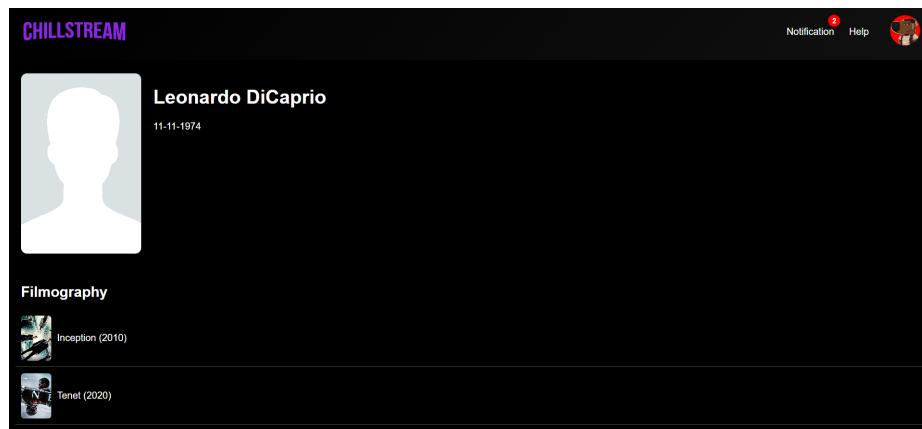


Figure 3.14: Pagina di informazioni relative a un attore

## Sezione recensioni utente

All'interno della stessa pagina del film, nella sezione **Reviews**, gli utenti possono:

- Scrivere una recensione personalizzata;
- Inviare il commento tramite il pulsante **Submit**;
- Leggere le recensioni pubblicate da altri utenti, ciascuna associata al nickname di chi l'ha scritta.

Questa sezione stimola l'interazione tra gli utenti e favorisce la scoperta di nuovi contenuti grazie ai giudizi della community.



Figure 3.15: Sezione delle recensioni all'interno della pagine singola di ogni film

### 3.5.3 Storyboard delle funzionalità accessibili dalla Navbar

Dalla barra di navigazione in alto a destra, l'utente può accedere rapidamente a tre funzionalità fondamentali per la gestione dell'esperienza d'uso: il centro assistenza, la sezione notifiche e le impostazioni del profilo.

#### Help Center

Cliccando sulla voce **Help**, si accede a una sezione dedicata all'assistenza utente. L'interfaccia è composta da:

- Una barra di ricerca per trovare articoli di supporto;
- Categorie tematiche come:
  - **Manage My Account**: gestione dei profili, email e password;
  - **Can't Watch**: problemi di riproduzione o sottotitoli;
  - **Quick Links**: azioni rapide come cancellazione account o aggiornamento email.

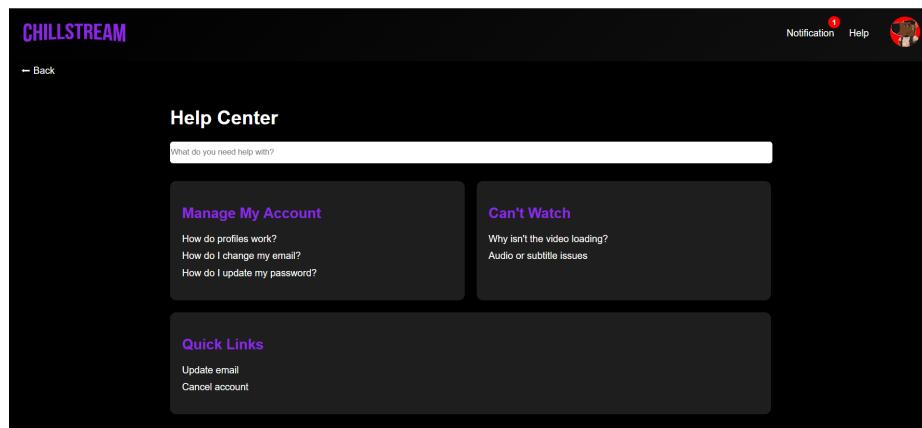


Figure 3.16: Pagina di aiuto in caso di problemi nel sistema

#### Sezione notifiche

L'icona **Notification**, visibile sempre nella navbar, conduce l'utente a un elenco ordinato di messaggi informativi provenienti dal sistema. Le notifiche possono includere:

- Messaggi di benvenuto o sistema;
- Comunicazioni relative a nuovi contenuti disponibili;
- Annunci da parte dell'amministrazione.

Ogni notifica è marcata da data e orario per facilitare la consultazione e le notifiche non ancora visualizzate sono contraddistinte dal fatto che rimangono evidenziate in bianco.



Figure 3.17: Pagina delle notifiche

### Impostazioni profilo

Cliccando sull'icona avatar in alto a destra, si accede alla pagina **Profile Settings**, in cui l'utente può:

- Modificare il nickname del profilo attivo;
- Cambiare l'immagine associata al profilo scegliendo tra avatar disponibili;
- Salvare le modifiche;
- Eliminare il profilo corrente;
- Effettuare il logout o cambiare profilo.

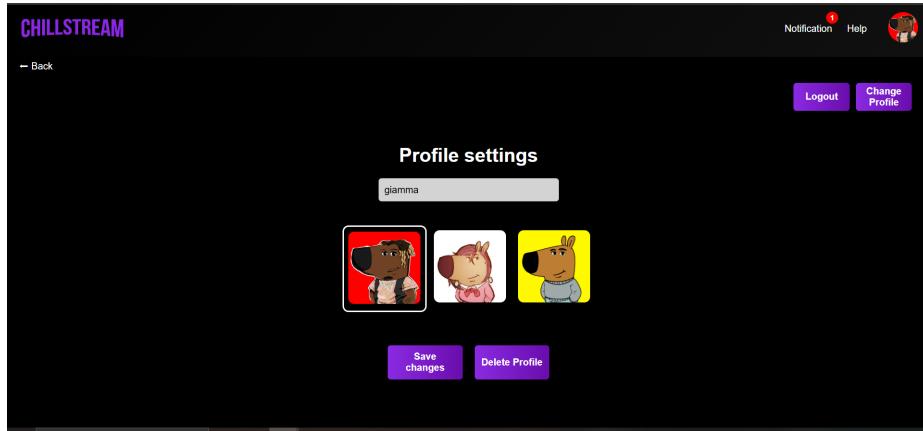


Figure 3.18: Pagina di impostazioni del profilo

### 3.5.4 Storyboard: Sezione Admin

L'applicazione Chillstream prevede un pannello dedicato agli amministratori di sistema, accessibile attraverso una login riservata. Dopo l'autenticazione, l'amministratore può gestire contenuti, attori e notifiche in modo centralizzato tramite tre sezioni principali nella navbar superiore: **Contents Management**, **Actors Management**, e **Send Notifications**.

#### Admin Login

L'accesso alla dashboard avviene tramite credenziali specifiche. Le password vengono trattate in modo sicuro (hashing lato backend).

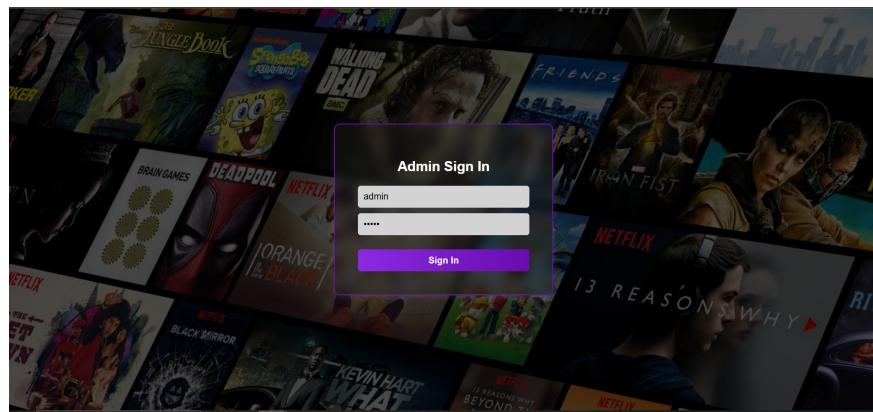


Figure 3.19: Login Admin – accesso riservato all'area gestionale

## Contents Management

All'interno della sezione **Contents Management**, l'amministratore ha la possibilità di:

- Inserire nuovi contenuti (film) indicando: titolo, anno, genere, rating, descrizione, file immagine, trailer e attori associati;
- Aggiornare contenuti esistenti tramite i medesimi campi;
- Cancellare contenuti esistenti fornendo il titolo.

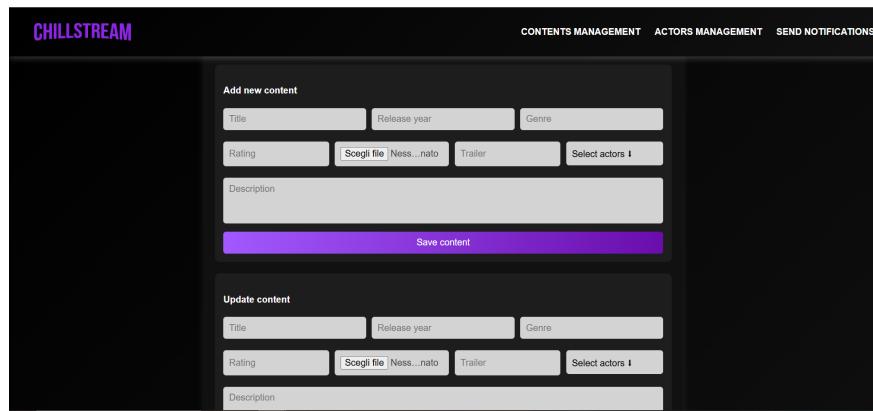


Figure 3.20: Gestione contenuti multimediali da parte dell'amministratore

## Actors Management

La sezione **Actors Management** consente la gestione degli attori coinvolti nei film. Le funzionalità includono:

- Aggiunta di un attore (nome, cognome, data di nascita);
- Modifica dei dati di un attore esistente;
- Eliminazione di un attore tramite nome e cognome.

The screenshot shows the 'ACTORS MANAGEMENT' section of the CHILLSTREAM application. At the top, there are navigation links: 'CONTENTS MANAGEMENT', 'ACTORS MANAGEMENT', and 'SEND NOTIFICATIONS'. Below these are three separate input forms:

- Add new actor:** Fields for Name, Surname, and date of birth, followed by a 'Save actors' button.
- Update actor:** Fields for Name, Surname, and date of birth, followed by an 'Update actor' button.
- Delete actor:** A field for Surname, followed by a 'Delete' button.

Figure 3.21: Gestione anagrafica attori associati ai film

### Invio Notifiche

Attraverso la sezione **Send Notifications**, l'amministratore può inviare comunicazioni globali visibili a tutti gli utenti. Queste vengono visualizzate all'interno dell'applicazione nella sezione notifiche dell'utente.

- Inserimento testo notifica;
- Invio immediato al database e visibilità lato client.

The screenshot shows the 'SEND NOTIFICATIONS' section of the CHILLSTREAM application. At the top, there are navigation links: 'CONTENTS MANAGEMENT', 'ACTORS MANAGEMENT', and 'SEND NOTIFICATIONS'. Below these is a single input form:

- Add new notifications:** A field for 'Notification text' and a 'Send' button.

Figure 3.22: Sezione per l'invio di notifiche globali agli utenti

# Chapter 4

## Tecnologie

All'interno di questa sezione troviamo spiegate le tecnologie più rilevanti utilizzate all'interno del progetto.

Patendo da tecnologie più generali per la gestione del progetto come docker fino ad arrivare a tecnologie molto specifiche utilizzate e pensate per una singola feature del progetto.

### 4.1 Docker

Nel progetto è stato utilizzato Docker come strumento fondamentale per garantire la portabilità e la coerenza dell'ambiente di esecuzione tra sviluppo e test. Attraverso l'uso dei container, l'intera applicazione è stata suddivisa in componenti indipendenti, ciascuna eseguita in un ambiente isolato ma comunicante, facilitando sia il deploy che la gestione dei servizi. In particolare, sono stati containerizzati il frontend (realizzato in Vue), il backend(Node.js e flask) e il database (MongoDB), permettendo a ciascuno di essere eseguito in maniera autonoma ma coordinata tramite Docker Compose.

Questa scelta ha consentito di evitare problemi legati alle diverse configurazioni delle macchine degli sviluppatori o ai conflitti tra librerie e dipendenze. L'impiego di Docker ha anche migliorato la stabilità del sistema nel tempo, assicurando che eventuali modifiche o aggiornamenti a uno dei componenti non compromettessero il funzionamento dell'intero ambiente.

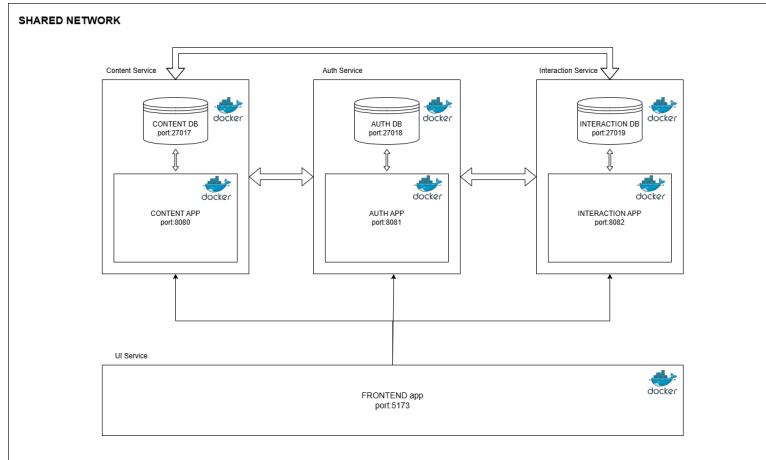


Figure 4.1: Docker architecture diagram

Come possiamo notare dalla figura 4.1 il sistema è composto da quattro microservizi:

- **Content Service**: gestisce i contenuti, con app su porta 8080 e database MongoDB su 27017.
- **Auth Service**: responsabile dell'autenticazione e dei profili, espone la porta 8081 e usa un DB su 27018.
- **Interaction Service**: gestisce interazioni utente (view, reviews, favorite), porta 8082, DB su 27019.
- **Frontend**: UI che comunica con i servizi backend, espone la porta 5173.

Tutti i servizi sono containerizzati con Docker e sono connessi tra loro tramite una rete condivisa (SHARED NETWORK). Ogni servizio ha il proprio database e comunica con gli altri tramite API interne.

#### 4.1.1 Shared Docker Network

All'interno del progetto Chillstream, è stata creata una rete Docker condivisa denominata **shared\_network**, alla quale sono stati collegati tutti i container dei diversi microservizi. L'utilizzo di una rete condivisa tra i container ha rappresentato un elemento fondamentale per consentire la comunicazione tra i diversi microservizi in maniera semplice ed efficace.

Quando si eseguono più container su Docker, ogni container di default è isolato e non è in grado di comunicare direttamente con gli altri. Creare una rete condivisa permette invece ai container collegati di riconoscersi per nome (hostname)

e scambiare dati tra loro utilizzando questi nomi come endpoint di rete. Questo consente, ad esempio, al microservizio `auth` di contattare `content` semplicemente usando l'indirizzo `http://content:porta`, senza dover conoscere l'IP esatto del container, che potrebbe cambiare ad ogni riavvio.

Questa configurazione migliora la manutenibilità, poiché ogni servizio può essere aggiornato o riavviato senza rompere la comunicazione con gli altri. Inoltre, l'uso di una rete Docker garantisce anche un maggiore livello di sicurezza, poiché i container comunicano solo all'interno di un ambiente isolato che non è visibile all'esterno della rete definita, riducendo i rischi legati all'esposizione accidentale di porte o servizi. Questo meccanismo ci ha permesso ad esempio di avere tutti i dati del relativo film quando viene aggiunto tra i visualizzati oppure tra i preferiti, in modo da poter provare che questo film (il suo id) sia effettivamente tra la collezione di film.

## 4.2 REST API

Nel progetto Chillstream, l'architettura REST API è stata adottata come paradigma fondamentale per la comunicazione tra i microservizi. Le REST API (Representational State Transfer Application Programming Interface) rappresentano uno stile architettonico per la progettazione di servizi web che consente ai sistemi distribuiti di comunicare tra loro in modo semplice, scalabile e stateless attraverso il protocollo HTTP.

Ogni microservizio del sistema — Auth, Content, Interaction — espone un insieme di endpoint RESTful che permettono al client (il frontend) di eseguire operazioni CRUD (Create, Read, Update, Delete) sulle rispettive risorse. Questi endpoint sono stati scelti in anticipo per apparire chiari: ad esempio, per ricevere la lista dei si utilizza una richiesta GET verso `/films`, per modificare un attore si utilizza un PUT su `/actors/:id`.

L'utilizzo delle REST API ha permesso di mantenere una forte separazione

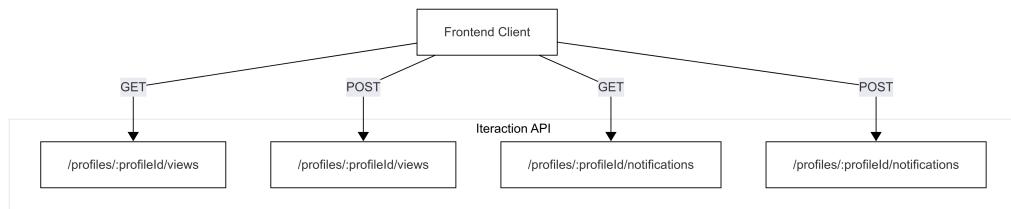


Figure 4.2: Frontend using interaction API

tra la logica di presentazione (frontend) e la logica di business e persistenza (backend), facilitando lo sviluppo parallelo delle componenti e migliorando la manutenibilità dell'intero sistema. Inoltre, grazie al fatto che le REST API sono basate su HTTP, è stato possibile testarle facilmente tramite strumenti come Postman e integrarle senza la necessità di protocolli o librerie complesse.

Nel nostro progetto, ogni chiamata proveniente dal frontend verso i microservizi è stata implementata utilizzando la libreria Axios, che semplifica l'invio di richieste HTTP e la gestione delle risposte. Le risorse restituite dai vari microservizi, in formato JSON, vengono quindi interpretate dal client e utilizzate per aggiornare dinamicamente l'interfaccia utente. L'approccio REST ha offerto numerosi vantaggi tra cui la possibilità di sfruttare appieno le potenzialità del protocollo HTTP, la semplicità nella gestione delle rotte e delle risorse, e soprattutto una grande interoperabilità, dato che le REST API possono essere consumate da qualsiasi client capace di effettuare richieste HTTP, rendendo il sistema facilmente estendibile anche verso applicazioni mobile o altri front-end futuri.

### 4.3 Two-Way Binding

Nel progetto ChillStream, una delle funzionalità fondamentali offerte dal framework Vue.js è il *two-way data binding*, ovvero il **dual binding**. Questa caratteristica permette di sincronizzare automaticamente il modello dati (JavaScript) e la vista (HTML) in entrambe le direzioni. In altre parole, ogni modifica apportata all'interfaccia utente aggiorna automaticamente il dato nel modello e viceversa.

Prendiamo per esempio l'utilizzo del **v-model**, la direttiva fornita da Vue.js per implementare il two-way binding, è stato fondamentale in diverse componenti del progetto, in particolare nei form per la registrazione, login, creazione profili e inserimento contenuti da parte dell'admin. Grazie a **v-model**, è stato possibile legare direttamente i campi di input ai dati del componente, semplificando enormemente la gestione dello stato locale e la validazione dei form.

Un esempio tipico può essere osservato nella gestione di un form per la creazione di un nuovo profilo utente:

```

1 <template>
2   <div>
3     <input v-model="profile.nickname" placeholder="Nickname" />
4     <input v-model="profile.profileImage" placeholder="Profile
      Image URL" />
5     <button @click="createProfile">Crea Profilo</button>
6   </div>
7 </template>
8
9 <script setup>
10 import { ref } from 'vue';
11
12 const profile = ref({
13   nickname: '',
14   profileImage: ''
15 });
16
17 const createProfile = () => {
18   // invio dei dati al backend tramite axios
19 };

```

20 </script>

In questo esempio, grazie all'utilizzo di **v-model** ogni modifica fatta nei campi di input aggiorna automaticamente l'oggetto **profile**, che può essere successivamente inviato al backend attraverso una chiamata HTTP con axios.

Il dual binding è stato particolarmente utile anche per mantenere coerente lo stato dei dati nei form di login, dove è importante legare i campi di input come **email** e **password** a variabili che verranno poi utilizzate per l'autenticazione.

Grazie a questa tecnologia, lo stato interno del componente viene costantemente aggiornato in tempo reale ogni volta che l'utente interagisce con un elemento del DOM, come un campo di input, una checkbox o una select. Questo aggiornamento immediato del modello dati lato client permette di lavorare con una copia locale delle informazioni, che può essere validata, modificata o anche visualizzata in tempo reale senza dover inviare continuamente richieste HTTP al backend.

### Vantaggi del dual binding in ChillStream

- **Maggiore reattività dell'interfaccia utente:** l'utente percepisce un'applicazione fluida, veloce e interattiva.
- **Minore carico sul server:** le chiamate HTTP vengono limitate ai soli momenti significativi (invio dati, salvataggio, ecc.).
- **Semplificazione del codice:** elimina la necessità di scrivere funzioni di aggiornamento manuali per i campi di input.
- **Maggiore leggibilità e mantenibilità:** i dati sono sempre allineati con la vista.
- **Efficienza nello sviluppo:** riduce la possibilità di errori nella sincronizzazione tra interfaccia utente e logica applicativa.

## 4.4 Axios

Axios è una libreria JavaScript che consente di effettuare richieste HTTP in modo semplice e intuitivo, basandosi sul concetto di Promises. All'interno del progetto, è stata utilizzata per gestire tutta la comunicazione asincrona tra il frontend, sviluppato in Vue, e i vari microservizi backend esposti tramite API REST.

Il vantaggio principale di Axios risiede nella sua capacità di semplificare l'invio di richieste HTTP come GET, POST, PUT e DELETE, permettendo di gestire facilmente la serializzazione e deserializzazione dei dati in formato JSON.

Nel contesto dell'applicazione Chillstream, Axios è stato utilizzato per inviare richieste verso i microservizi responsabili dell'autenticazione, della gestione dei profili utente, del recupero delle informazioni relative ai film e agli attori, nonché

per la gestione delle liste personalizzate come “continua a guardare” e “MyList”. Inoltre, Axios consente di gestire facilmente le risposte asincrone grazie alla struttura delle Promises, garantendo un flusso chiaro delle operazioni in caso di successo o fallimento. Questo ha migliorato significativamente l'affidabilità del frontend, permettendo di fornire un feedback immediato all'utente in caso di errori o problemi di connessione.

## 4.5 CORS

Nel progetto Chillstream, l'utilizzo della libreria CORS (*Cross-Origin Resource Sharing*) è stato fondamentale per permettere al frontend, ospitato su un dominio o una porta diversa rispetto ai vari microservizi backend, di effettuare chiamate HTTP come GET, POST, PUT e DELETE senza incorrere in restrizioni imposte dal browser.

CORS è un meccanismo di sicurezza implementato dai browser per impedire che script eseguiti da un'origine possano accedere a risorse su un'altra origine senza esplicita autorizzazione. Questo comportamento è essenziale per proteggere l'utente da possibili attacchi cross-site, ma nei progetti basati su architetture a microservizi o su separazione tra client e server come il nostro, è necessario configurare il backend per permettere tali interazioni.

All'interno dei microservizi sviluppati in Node.js (con Express) e Flask, è stata integrata la libreria CORS per dichiarare esplicitamente le origini autorizzate a comunicare con ciascun servizio. Ad esempio, nel backend Express è stata utilizzata la libreria `cors` tramite il middleware `app.use(cors())`, mentre nei servizi Flask è stato utilizzato il modulo `flask_cors` tramite `CORS(app)`. In entrambi i casi, la configurazione ha permesso di autorizzare l'origine del frontend (ad esempio `http://localhost:5173`) potendo in questa maniera svolgere richieste API senza problemi.

Questa configurazione ha garantito che tutte le richieste HTTP provenienti dal client fossero accettate dai server senza blocchi dovuti alle politiche di sicurezza dei browser. In assenza di tale configurazione, qualsiasi tentativo del frontend di interagire con le API sarebbe stato respinto con un errore di tipo CORS, impedendo il corretto funzionamento dell'applicazione.

L'integrazione di CORS ha quindi rappresentato un passaggio essenziale per l'interoperabilità tra i componenti dell'architettura distribuita, rendendo possibile una comunicazione fluida e sicura tra frontend e backend anche in ambienti di sviluppo e produzione differenti.

## 4.6 SocketIO

All'interno del progetto, Socket.IO è stato utilizzato per gestire la comunicazione in tempo reale, in particolare per la funzionalità di chat live presente nella sezione di streaming video. L'obiettivo era permettere agli utenti connessi alla stessa stanza di interagire istantaneamente, senza la necessità di ricaricare la

pagina o interrogare continuamente il server tramite richieste HTTP (polling). Socket.IO si basa su WebSocket, ma offre un'interfaccia più semplice e robusta, con gestione automatica della riconnessione, fallback in caso di incompatibilità e un sistema di eventi personalizzati. La scelta di questo strumento ha permesso di creare un'esperienza fluida e interattiva per l'utente.

#### 4.6.1 Comunicazione in tempo reale per stanza

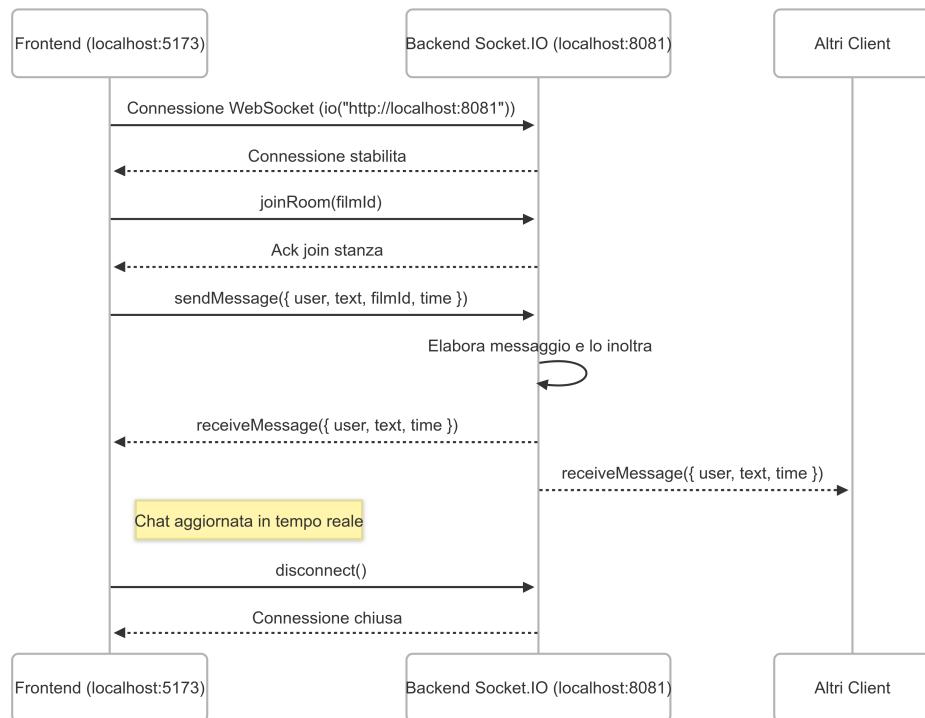


Figure 4.3: SockerIO Schema

Come si può notare dalla figura , il client (eseguito su `localhost:5173`) si connette al server backend (in ascolto su `localhost:8081`) e interagisce secondo una sequenza di eventi strutturati. La seguente descrizione schematizza la sequenza tipica delle comunicazioni:

- 1. Connessione WebSocket:** appena il componente Vue.js della chat viene montato, viene inizializzata una connessione tramite Socket.IO verso il server(in questo caso authAPI):

```
1 const socket = io("http://localhost:8081");
```

2. **Join Room:** subito dopo, il client emette un evento per entrare in una stanza corrispondente all'ID del film:

```
socket.emit("joinRoom", filmId);
```

3. **Messaggio di ingresso:** viene generato un messaggio automatico che notifica l'ingresso dell'utente nella chat:

```
1 socket.emit("sendMessage", {
2   user: "System",
3   text: `${nickname}    entrato nella chat.`,
4   filmId,
5   time
6});
```

4. **Invio messaggi:** ogni volta che l'utente preme invio, viene creato un nuovo oggetto messaggio e inviato al server:

```
1 socket.emit("sendMessage", {
2   user: nickname,
3   text: newMessage.value,
4   filmId,
5   time
6});
```

5. **Ricezione dei messaggi:** lato client, viene ascoltato l'evento `receiveMessage` per visualizzare i messaggi ricevuti:

```
1
2 socket.on("receiveMessage", (message) => {
3   messages.value.push(message);
4});
```

6. **Disconnessione:** quando il componente viene smontato, la connessione viene chiusa:

```
socket.disconnect();
```

## Funzionamento lato server

Il server Node.js riceve gli eventi dai vari client e li redistribuisce nella stanza corrispondente:

```
1
2 // Evento join room
3 socket.on("joinRoom", (roomId) => {
4   socket.join(roomId);
5});
6
```

```
7 // Evento messaggio ricevuto
8 socket.on("sendMessage", (message) => {
9     io.to(message.filmId).emit("receiveMessage", message);
10});
```

### Vantaggi di questo approccio

- Comunicazione asincrona e in tempo reale.
- Architettura a stanze per separare le chat per film.
- Minor latenza rispetto a polling HTTP.
- Interfaccia reattiva e fluida per l'utente.

# Chapter 5

## Codice

### 5.1 Utilizzo della sintassi `<script setup>` in Vue.js

Nel progetto ChillStream, è stata adottata la sintassi `<script setup>` introdotta con Vue 3 per la definizione dei componenti. Questa nuova modalità semplifica e rende più leggibile la struttura dei componenti, migliorando l'esperienza di sviluppo. La sintassi `setup` è una forma più concisa per scrivere il blocco `setup()` che normalmente si userebbe nella compositional API di Vue, eliminando la necessità di un ritorno esplicito dei dati e dei metodi dal blocco.

Di seguito un esempio di codice del progetto:

```
1 <script setup>
2 const menuOpen = ref(false);
3 const unreadCount = ref(0);
4 const toggleMenu = () => {
5   menuOpen.value = !menuOpen.value;
6 };
```

Questa versione è estremamente compatta e leggibile, poiché non richiede l'esplicito ritorno di variabili o funzioni, e permette di definire in modo diretto tutti gli elementi del componente.

#### 5.1.1 Confronto con la sintassi tradizionale

La stessa logica, scritta con la sintassi tradizionale, apparirebbe così:

```
1 <script>
2 export default {
3   components: {
4     Logo,
5     NotificationButton
6   },
7   setup() {
8     const menuOpen = ref(false);
9     const unreadCount = ref(0);
10    const toggleMenu = () => {
```

```

11     menuOpen.value = !menuOpen.value;
12   };
13 }
14 </script>

```

Come si può osservare, la versione tradizionale richiede più boilerplate, come la dichiarazione esplicita del blocco `setup()` e il ritorno manuale di tutte le variabili e funzioni utilizzate nel template.

### Vantaggi di `<script setup>`

L'utilizzo di `<script setup>` porta con sé numerosi vantaggi:

- Riduce la quantità di codice necessario per definire un componente.
- Migliora la leggibilità grazie a una struttura più diretta e lineare.
- Ottimizza le performance a livello di compilazione, grazie all'integrazione diretta con il compilatore di Vue.
- Rende più naturale l'importazione e l'utilizzo di componenti figli.

## 5.2 Components di Vue.js

Vue.js fornisce un'architettura fortemente orientata ai componenti. Questa caratteristica si è rivelata particolarmente vantaggiosa nello sviluppo del progetto ChillStream, in quanto ha permesso la creazione di interfacce modulari, riutilizzabili e facilmente manutenibili.

Un componente rappresenta una sezione autonoma dell'interfaccia utente, con una propria logica, stile e template HTML. Questo approccio consente di mantenere il codice organizzato, ridurre la duplicazione e rispetta quindi i pensieri DRY e KISS.

Ad esempio, la seguente struttura è stata utilizzata per creare la homepage dell'utente, definita come un insieme di vari componenti:

```

1 <template>
2   <div class="homepage">
3     <UserNavbar/>
4
5     <div class="content">
6       <ContentRow title="Trending Movies" :contents="allMovies"/>
7       <ContentRow title="Continue to watching" :contents="
8         viewedMovies"/>
9       <ContentRow title="My List" :contents="MyListMovie"/>
10    </div>
11  </div>

```

Qui il componente `homepage` incorpora una barra di navigazione (`UserNavbar`) e una sezione centrale con tre righe di contenuti visuali, ciascuna rappresentata da un’istanza del componente `ContentRow`. Questo componente accetta due proprietà (`props`): il titolo della riga e un array di contenuti da visualizzare. Grazie a questa struttura, è possibile riutilizzare `ContentRow` per mostrare diverse categorie di film con lo stesso layout ma contenuti diversi.

I componenti a parte a essere riutilizzabili aumentano anche di molto la pagina la leggibilità del nostro codice, prediamo per esempio la pagina di gestione dei contenuti, dedicata all’amministratore:

```
1 <template>
2   <div class="content-management">
3     <h3>Contents Management</h3>
4     <save-content-form></save-content-form>
5     <update-content-form></update-content-form>
6     <delete-component-form></delete-component-form>
7   </div>
8 </template>
```

In questo frammento di codice, il componente `content-management` funge da contenitore per tre sotto-componenti: `save-content-form`, `update-content-form` e `delete-component-form`. Ciascuno di questi rappresenta un modulo distinto per aggiungere, modificare o eliminare contenuti. In questo modo, ogni funzionalità amministrativa è incapsulata in un blocco autonomo e riutilizzabile che definisce perfettamente la il suo scopo.

L’approccio a componenti ha permesso quindi di costruire l’interfaccia di Chill-Stream in maniera scalabile, mantenendo ogni sezione del sito separata e facilmente estendibile. Inoltre, grazie alla possibilità di passare dati tra componenti tramite `props` e `emit`, la comunicazione interna tra le parti dell’applicazione è risultata semplice ed efficace.

# Chapter 6

## Test

I test sono una parte cruciale del processo di sviluppo, in quanto garantiscono che l'applicazione si comporti come previsto e soddisfi i requisiti funzionali e non funzionali. Per questo progetto, i test sono stati condotti a vari livelli.

Le funzionalità del sistema ChillStream sono state testate dal team su diversi browser di riferimento, in particolare Google Chrome, Microsoft Edge e Mozilla Firefox, con l'obiettivo di verificarne la correttezza, la stabilità e la portabilità. L'attenzione si è concentrata su funzionalità complesse, come la chat in tempo reale basata su `Socket.IO`.

Anche le API REST sviluppate per i microservizi (Auth, Content, Interaction) sono state testate tramite `Postman`, così da garantire l'integrità delle comunicazioni client-server e la corretta gestione delle richieste HTTP. In aggiunta ai test tecnici, è stata condotta un'analisi euristica secondo i principi di usabilità di Nielsen, da cui sono emerse le seguenti considerazioni:

- **Controllo e Libertà:** l'interfaccia è stata progettata per ridurre al minimo le operazioni necessarie, facilitando l'interazione e migliorando la fluidità d'uso.  
notif
- **Aiuto all'utente:** sono presenti messaggi informativi e label di errore che supportano l'utente in caso di input errati, come l'inserimento di un'e-mail non valida o un accesso fallito.
- **Prevenzione degli errori:** l'esperienza d'uso è guidata e semplificata, cercando di evitare scenari ambigui che potrebbero confondere gli utenti.
- **Riconoscimento più che ricordo:** le icone e i bottoni sono progettati per essere autoesplicativi. Lo stile uniforme tra le pagine favorisce l'intuitività e la navigazione.

- **Visibilità dello stato del sistema:** le operazioni asincrone, come il caricamento dei contenuti o l'invio di notifiche, sono accompagnate da feedback visivi che informano l'utente sull'esito delle azioni.
- **Corrispondenza tra sistema e mondo reale:** terminologie comuni e icone comprensibili aiutano a mantenere coerenza con l'esperienza degli utenti.
- **Consistenza e standard:** l'interfaccia presenta una gamma coerente di colori e componenti grafici, facilitando il riconoscimento delle funzionalità più importanti.
- **Estetica e design minimalista:** si è cercato durante lo sviluppo del sito di non aggiungere a schermo elementi che non siano importanti per l'usabilità del prodotto.
- **Flessibilità ed efficienza d'uso:** i menu e le sezioni sono semplici da consultare e da utilizzare anche per utenti inesperti, grazie all'utilizzo di pattern convenzionali.
- **Documentazione:** grazie al design essenziale e all'elevata usabilità, si è deciso di fornire come documentazione all'utente il minimo indispensabile cioè una pagina help.

## Approccio iterativo e feedback utente

I test di usabilità sono stati effettuati in modo iterativo lungo tutto il processo di sviluppo. Fin dalla fase di prototipazione, gli utenti hanno potuto fornire feedback che sono stati utilizzati per migliorare l'esperienza d'uso e correggere problematiche funzionali, portando alla realizzazione di un sistema solido, intuitivo e godibile.

## Verifica dell'accessibilità (WCAG 2.0)

Durante lo sviluppo dell'applicazione è stata riservata particolare attenzione all'**accessibilità**, prendendo come riferimento le linee guida WCAG 2.0 (*Web Content Accessibility Guidelines*) per migliorare l'usabilità anche per utenti con disabilità visive o motorie.

Pur non avendo eseguito test automatici o formali di validazione, sono stati adottati alcuni accorgimenti utili per favorire l'inclusività:

- **Contrasto visivo:** la combinazione di colori tra sfondi scuri e testo chiaro è stata scelta per garantire una buona leggibilità, in linea con il criterio WCAG 1.4.3.
- **Layout coerente e ordinato:** la struttura visiva semplice e gerarchica delle pagine facilita la navigazione e la comprensione dei contenuti.

- **Chiarezza testuale e icone esplicative:** bottoni e funzioni principali sono accompagnati da etichette intuitive o icone riconoscibili, che aiutano anche utenti con ridotta familiarità con le interfacce digitali.

Sebbene l'applicazione non raggiunga una piena conformità con il livello **WCAG 2.0 – AA**, si può affermare che implementa diversi principi fondamentali di accessibilità.

# Chapter 7

# Deployment

## Step 1: Clonare i progetti

Copiare l' indirizzo HTTPS dei seguenti repository e poi copiarlo dopo il comando `git clone`

- Frontend
- Auth service
- Content service
- Interaction service

**Step 2: Creazione di una rete Docker condivisa** Prima di procedere al deployment del sistema, è stata creata una rete condivisa tra i container. Questo ha permesso ai vari microservizi (Auth, Content, Interaction) di comunicare tra loro in modo efficiente e isolato.

```
docker network create shared_network
```

**Step 3: Build e avvio dei servizi** Per costruire le immagini Docker e avviare i container, è necessario utilizzare il comando all'interno della directory dove si trova il file `docker-compose.yml`:

```
docker-compose up --build
```

## Step 4: Verifica del deployment

Una volta avviati i container, lo stato dei servizi può essere verificato con:

```
docker ps
```

## Step 5: Accesso all'applicazione

Dopo l'avvio dei container, il frontend è accessibile nel browser all'indirizzo:

```
http://localhost:5173
```

## Chapter 8

# Conclusione e sviluppi futuri

Il progetto *ChillStream* rappresenta un'applicazione concreta dei principi di architettura software moderna, con particolare attenzione alla modularità, alla flessibilità e alla manutenibilità. Grazie all'adozione di un'architettura a microservizi, il sistema risulta scalabile, robusto e facilmente estendibile nel tempo.

### Principali risultati ottenuti:

- **Architettura a microservizi:** ogni servizio ha responsabilità ben definite ed è sviluppato in modo indipendente, consentendo aggiornamenti e deploy mirati senza impattare sull'intero sistema.
- **Containerizzazione:** tutti i componenti (backend e frontend) sono stati containerizzati tramite Docker e orchestrati con Docker Compose, facilitando il processo di deploy e garantendo una comunicazione stabile tra i servizi.
- **Integrazione Frontend:** il frontend sviluppato con Vue.js interagisce efficacemente con i microservizi, offrendo un'interfaccia utente reattiva e moderna.

### 8.0.1 Sviluppi futuri

Di seguito alcune migliorie che si potranno applicare in futuro:

- **Miglioramenti in termini di scalabilità:**

In caso di un'incremento di utilizzo Implementazione di un sistema di bilanciamento del carico (*load balancing*) per gestire un maggior numero di utenti simultanei. Migrazione da Docker Compose a Kubernetes per un'orchestrazione più avanzata e scalabile dei container.

- **Potenziali aggiornamenti in sicurezza:** Introduzione di un sistema di autenticazione basato su token (JWT) per rendere più sicura la comunicazione tra client e server, anche attraverso l'utilizzo di `axios`. Abilitazione del protocollo HTTPS per garantire la cifratura del traffico tra servizi.
- **Ottimizzazioni prestazionali:** Integrazione di meccanismi di caching per migliorare i tempi di risposta nelle richieste più frequenti. Ottimizzazione delle query al database per ridurre la latenza e migliorare le prestazioni complessive del sistema.
- **Barra di ricerca dei film:** pianificata l'introduzione di una barra di ricerca che permetta agli utenti di trovare facilmente i film all'interno della piattaforma.
- **Supporto multilingua:** aggiunta della funzionalità per il cambio lingua dell'interfaccia utente, tramite un selettori accessibile e persistente. Questo consente una maggiore inclusività e adattabilità dell'applicazione per un pubblico internazionale. Il sistema potrà supportare più lingue tramite file di localizzazione (es. JSON) e garantire la traduzione dinamica di contenuti, messaggi di errore e interfacce.