Let's Stream It Report
Applicazioni e Servizi Web 22/23

Luca Fabri
luca.fabri@studio.unibo.it
Simone Ceredi
simone.ceredi@studio.unibo.it

7 gennaio 2025

Indice

1 Introduzione **3**

2 Requisiti **4**
2.1 Requisiti di business 4
2.2 Requisiti utente 4
2.3 Requisiti funzionali 5
2.4 Requisiti non funzionali 7
2.5 Requisiti di implementazione 7

3 Design **8**
3.1 Target User Analysis 8
3.2 Mockup iniziali 10
3.3 Architettura del sistema 15
3.4 Design dei microservizi 15

4 Implementazione **17**
4.1 Frontend Service 17
4.1.1 VueJs features 18
4.1.2 Tecnologie 19
4.2 Session Service 20
4.2.1 Tecnologie 22
4.3 Auth Service 22
4.3.1 Tecnologie 22
4.4 Profile Service 23
4.4.1 Tecnologie 23

5 Test **25**
5.1 Unit tests 25
5.2 Test end-to-end 25
5.3 Qualità pagine web 26

6 Deployment **28**

7 Conclusioni **29**

Capitolo 1

Introduzione

La piattaforma Let's Stream It nasce per dare la possibilità ad un gruppo di amici di riprodurre video di YouTube insieme ed in modo sincronizzato. La principale caratteristica dell'implementazione è la sessione, che un utente può creare e condividere con i propri amici, in modo da poter guardare un video tutti insieme. Trattandosi di una riproduzione sincronizzata tutti gli utenti potranno fermare, riprodurre, avanzare o arretrare la riproduzione. Durante la visione del video è possibile utilizzare la chat per comunicare con i propri amici e condividere opinioni, giudizi, pensieri o altro con tutti i partecipanti.

Il progetto si basa su altri servizi simili quali Watch2Gether e TwoSeven. Le principali caratteristiche della piattaforma sono:

- Sistema per la gestione sicura delle credenziali di accesso. Solamente gli utenti autenticati potranno accedere alla piattaforma;
- Sistema per la gestione del profilo dell'utente;
- Sistema per la gestione della sessione di streaming e della chat.

Capitolo 2

Requisiti

In questo capitolo vengono descritti i requisiti del software implementato. Questi sono stati fissati durante le prime fasi del progetto e sono rimasti invariati durante tutte le fasi di sviluppo della piattaforma. L'utente ha rappresentato il punto centrale attorno al quale si è svolta tutta la definizione dei requisiti, la progettazione e anche il design delle interfacce andando a valorizzare l'esperienza utente.

2.1 Requisiti di business

L'applicazione dovrà disporre delle seguenti caratteristiche:

1. Business:
    1.1 Possibilità di visionare video di YouTube in maniera sincronizzata assieme ad altri utenti;
    1.2 Possibilità di comunicare in tempo reale durante la visione del video attraverso la chat.

2.2 Requisiti utente

L'utente potrà usufruire dei seguenti aspetti:

2. Utente:
    2.1 Autenticazione al sito;

2.1.1 Registrazione di un nuovo utente;
2.1.2 Login di un utente già registrato;
2.1.3 Logout di un utente.
2.2 Gestione della sessione;
2.2.1 Creazione di una sessione;
2.2.2 Accesso ad una sessione;
2.2.3 Uscita dalla sessione.
2.3 Riproduzione sincronizzata del video;
2.3.1 Fermare la riproduzione del video;
2.3.2 Riprendere la riproduzione del video;
2.3.3 Avanzare la riproduzione del video;
2.3.4 Arretrare la riproduzione del video.
2.4 Comunicazione tramite chat;
2.4.1 Invio di un nuovo messaggio;
2.4.2 Ricezione messaggi inviati da altri utenti nella sessione;
2.4.3 Ricezione di notifiche riguardanti la connessione/disconnessione di un utente.
2.5 Gestione del profilo utente;
2.5.1 Visualizzazione del profilo di un utente;
2.5.1.1 Visualizzazione dei dati personali;
2.5.1.2 Visualizzazione dei video guardati;
2.5.2 Modifica dei dati personali.

2.3 Requisiti funzionali

La piattaforma sviluppata rispetterà i seguenti requisiti:

3. Funzionali:
3.1 Gestione dell'autenticazione degli utenti al sito;
3.1.1 Registrazione di utenti che non dispongono di credenziali;
3.1.1.1 Verifica semantica dell'email;
3.1.1.2 Verifica unicità dell'email;

3.1.1.3 Verifica sicurezza della password;

3.1.1.4 Verifica che password e conferma coincidano;

3.1.1.5 Hashing della password per la memorizzazione su database.

3.1.2 Login di utenti in possesso di credenziali;

3.1.2.1 Richiesta di email e password per l'autenticazione alla piattaforma;

3.1.2.2 Verifica dell'esistenza di un utente con email uguale a quella inserita;

3.1.2.3 Verifica la password inserita coincida con quella specificata al momento della registrazione.

3.1.3 Gestione dell'autenticazione tramite apposito token;

3.1.3.1 Verifica della validità del token, per garantire che l'utente sia ancora autenticato;

3.1.3.2 Refresh del token allo scadere dello stesso;

3.1.3.3 Recupero di email e username dell'utente autenticato utilizzando i dati salvati nel token.

3.1.4 Logout di utenti autenticati alla piattaforma;

3.2 Gestione della sessione

3.2.1 Creazione di una sessione;

3.2.1.1 Il creatore della sessione inserisce url del video YouTube che vuole vedere;

3.2.1.2 Il creatore entra automaticamente nella sessione appena creata;

3.2.1.3 Il creatore condivide l'url attuale della piattaforma con gli amici.

3.2.2 Accesso ad una sessione;

3.2.2.1 L'utente apre l'url ricevuto dal creatore della sessione entrando così nella sessione.

3.2.3 Riproduzione sincronizzata del video;

3.2.3.1 Possibilità per chiunque partecipi alla sessione di effettuare azioni sul video in riproduzione che verranno sincronizzate con tutti gli utenti;

3.2.3.2 Mettere in pausa;

3.2.3.3 Riprendere un video precedentemente in pausa;

3.2.3.4 Avanzare la riproduzione del video;

3.2.3.5 Indietreggiare la riproduzione del video.

3.2.4 Invio e ricezione messaggi nella chat della sessione;

3.2.4.1 Invio di messaggi;

3.2.4.2 Ricezione di messaggi;

3.2.4.3 Ricezione di notifiche riguardanti accesso e uscita dalla sessione.

3.3 Gestione del profilo utente;

3.3.1 Gestione delle informazioni personali;

3.3.1.1 Modifica dell'username;

3.3.1.2 Modifica della bio.

3.3.2 Visualizzazione degli ultimi video visti;

3.3.3 Visualizzazione del profilo di un altro utente.

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 MEVN;

5.2 Utilizzo della libreria Socket.IO per la realizzazione delle comunicazioni real-time;

5.3 Utilizzo di un'altra tecnologia oltre allo stack MEVN, ricaduta su Scala.

Capitolo 3

Design

Il design dell'applicazione é stato realizzato in modo tale da rendere l'esperienza utente il piú gradevole possibile. La metodologia di design utilizzata é Human Centered Design, nella quale, iterativamente, l'utente specifica i requisiti del sistema e verifica che l'implementazione sia adeguata alle sue necessitá, una volta che la feature viene terminata dal team.

Gli utenti a cui si sta riferendo sono utenti "virtualizzati", ovvero delle persone eventuali che usufruiscono dell'applicativo, anche chiamati Personas. I vari scenari identificati sono serviti al team ad indirizzare le scelte progettuali.

L'applicazione inoltre:

1. Segue i principi di Responsive Design, la quale permette la fruizione dei contenuti sia in modalitá Desktop che Mobile;
2. É Desktop first. Le interfacce sono state progettate per i dispositivi Desktop in prima fase. Successivamente, il supporto é stato esteso per i dispositivi mobili;
3. Segue il principio KISS. Si é cercato di di sviluppare l'applicazione in modo tale che le interfacce avessero un design minimale per rendere il sito user-friendly.

3.1 Target User Analysis

Il target della piattaforma comprende diverse tipologie di utenti che vogliono guardare video YouTube con i propri amici; a seguito vengono individuate di-

verse *personas* che si differenziano tra loro sulla base di ciò che si aspettano dalla piattaforma.

Personas: Simone

Simone é uno studente a cui piace nel tempo libero guardare video su YouTube. Ha trovato un video interessante che vuole guardare insieme a dei suoi amici, tuttavia non si trovano nel suo stesso luogo fisico.
Senario d'uso: Guardare un video con i propri amici a distanza

1. Simone accede alla piattaforma *Let's Stream It*;
2. Si iscrive registrandosi con i suoi dati ed effettua il login;
3. Copia l'URL del video di YouTube che vuole visionare;
4. Crea una sessione di streaming incollando l'URL;
5. Copia l'URL della sessione a cui viene indirizzato e lo condivide con i suoi amici;
6. Simone, come i suoi amici all'interno della sessione, puó riprodurre il video, metterlo in pausa o spostare la timeline;
7. Simone, come i suoi amici all'interno della sessione, puó inviare messaggi via chat per reagire al video.

Personas: Andrea e Fabrizio

Andrea e Fabrizio sono due studenti di ingegneria biomedica presso UniBo, che studiano per passare l'esame di algebra e geometria. Durante il periodo natalizio, non possono andare a ricevimento dal professore a farsi spiegare l'integrazione per parti, concetto a loro non chiaro, decidono quindi che un video YouTube sull'argomento sia la loro miglior risorsa.
Scenario d'uso: Guardare un video educativo per comprendere meglio dei concetti di algebra

1. Andrea accede alla piattaforma *Let's Stream It*;
2. Si iscrive registrandosi con i suoi dati ed effettua il login;
3. Copia l'URL del video di YouTube che vuole visionare;
4. Crea una sessione di streaming incollando l'URL;
5. Copia l'URL della sessione a cui viene indirizzato e lo condivide con Fabrizio;

6. Fabrizio accede alla sessione in quanto già registrato alla piattaforma;
7. Andrea e Fabrizio riproducono il video, possono metterlo in pausa e spostare la timeline;
8. Andrea chiede un chiarimento tramite chat a Fabrizio su una parte che non gli è chiara;
9. Andrea e Fabrizio terminata la visione del video escono dalla piattaforma;

Personas: Alice

Alice è una veterinaria curiosa che è sempre stata appassionata all'architettura romana e che ha recentemente visto in televisione un documentario sull'ordine corinzio.
Scenario d'uso: Alice vuole condividere la sua passione con gli amici e guardare assieme a loro un documentario

1. Alice accede alla piattaforma *Let's Stream It*;
2. Si iscrive registrandosi con i suoi dati ed effettua il login;
3. Copia l'URL del documentario di YouTube che vuole visionare;
4. Crea una sessione di streaming incollando l'URL;
5. Copia l'URL della sessione a cui viene indirizzata e lo condivide con gli amici;
6. Alice e gli amici guardano il documentario, mettendolo in pausa e spostando la timeline;
7. Alice e gli amici reagiscono al video e comunicano utilizzando la chat della sessione.

3.2 Mockup iniziali

Per la realizzazione della user interface, come giá menzionato in precedenza, si é scelto un design minimale in modo che ogni tipologia di utente possa muoversi facilmente all'interno dell'applicazione. Per minimale si intende un'interfaccia contenente un'insieme di pagine, ciascuna focalizzandosi su un insieme ristretto di funzionalitá (una o poco piú) e con un numero di componenti al suo interno limitati al necessario.

Nella prima fase di design, sono stati creati dei mockup per avere un'idea di massima del risultato finale dell'applicazione. Questi sono anche risultati utili per i membri del team poiché hanno permesso una migliore suddivisione del lavoro.

Per la loro realizzazione é stato utilizzato lo strumento Balsamiq il quale permette la creazione di wireframes.

A seguire vengono mostrati i mockup delle pagine che compongono l'applicazione: Home, Login, Registration, Session e Profile e il popup Create Session per creare la sessione.

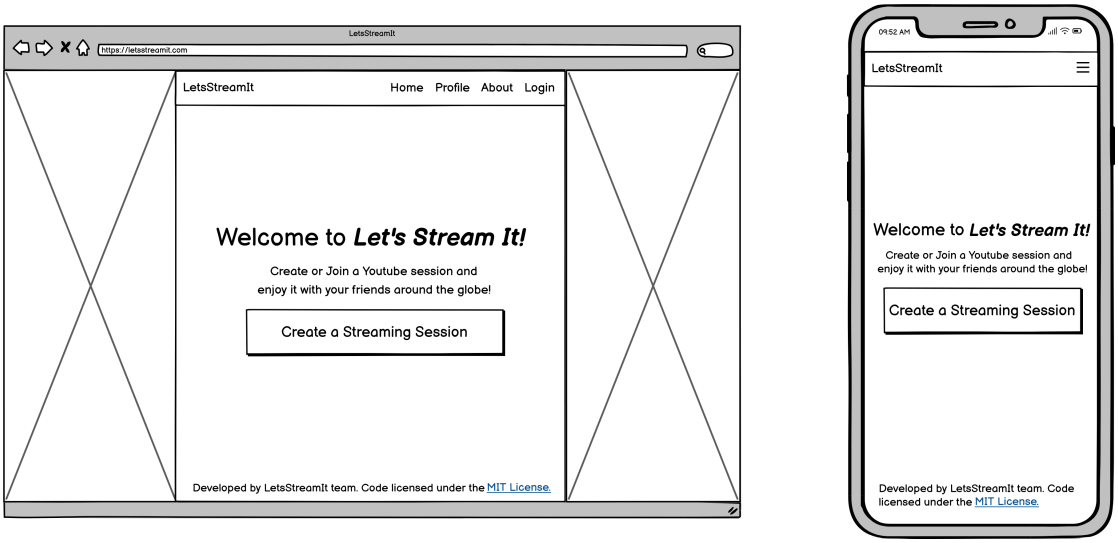


Figura 3.1: Home page

La Home page contiene un menú attraverso cui é possibile navigare all'interno dell'applicazione e un pulsante "Create a Streaming Session". L'utente in questa fase non ha effettuato l'accesso, pertanto un click su quest'ultimo reindirizza l'utente alla pagina di Login. Stessa cosa se l'utente vuole visualizzare la sua pagina del Profilo.

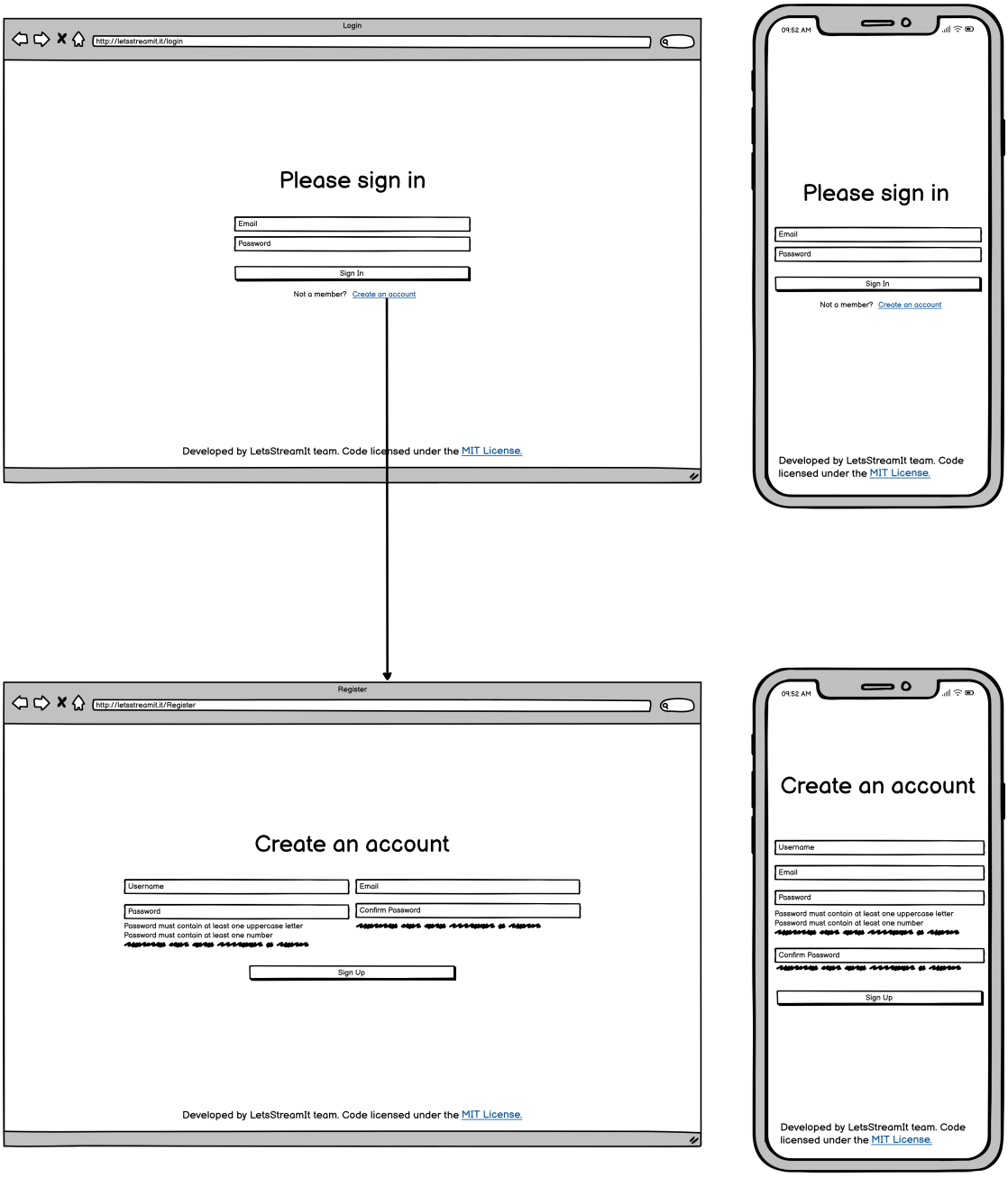


Figura 3.2: Login e Registration page

Una volta effettuata la registrazione e login, sul menú della Home page deve comparire la scritta Logout al posto di Login e il un click sul pulsante “Create a Streaming Session” deve far comparire la finestra popup per creare la sessione.

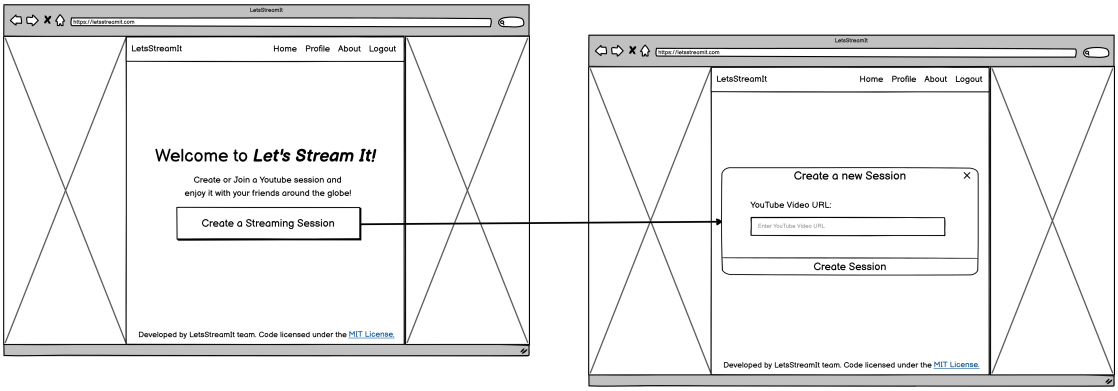


Figura 3.3: Create Session Popup

L'utente in questa fase puó incollare l'URL del video di YouTube e creare la sessione. Se l'URL é valido l'applicativo deve effettuare un reindirizzamento automatico alla pagina della sessione dedicata.

Per condividere la sessione con i suoi amici, l'utente puó copiare l'URL della sessione, contenente l'ID.

Ogni utente puó ora interagire con video e chat della sessione.

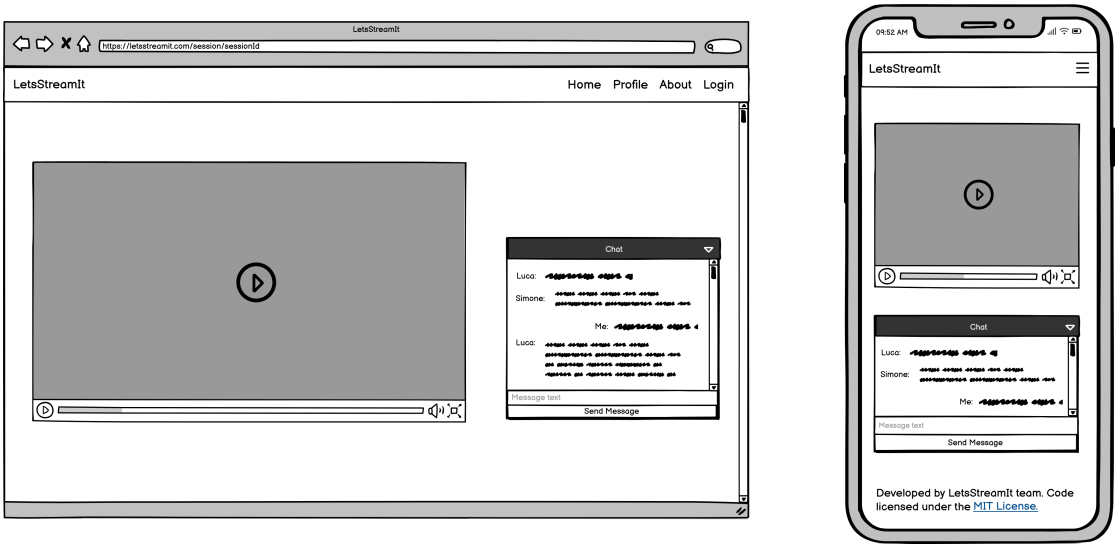


Figura 3.4: Session Page

L'utente loggato puó inoltre navigare la pagina del profilo personale per vedere o modificare le sue informazioni, o visualizzare il profilo di un altro utente. In que-

st'ultimo caso, non deve comparire il pulsante "Update", ma il restante contenuto deve rimanere invariato.

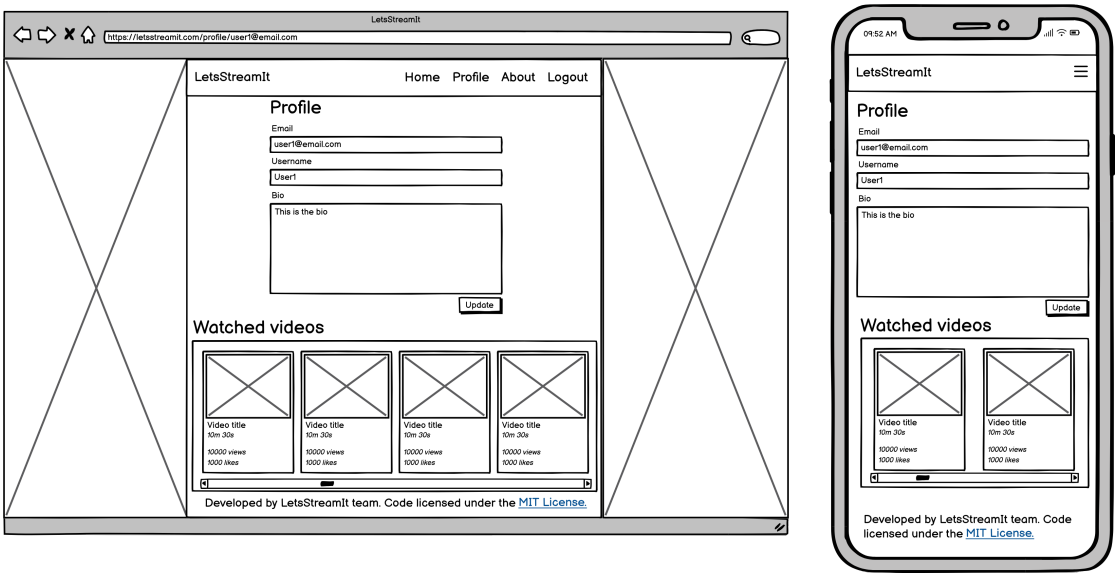


Figura 3.5: Profile Page

Infine, l'About page mostra le informazioni sul progetto e sui membri del team.

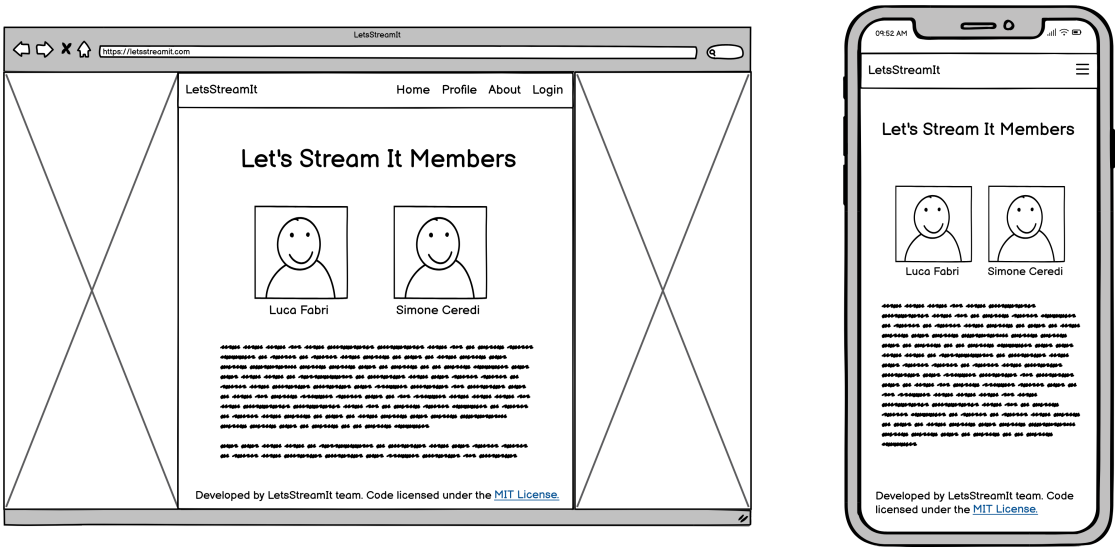


Figura 3.6: About Page

3.3 Architettura del sistema

Il sistema é stato progettato mediante l'architettura a microservizi. Questo tipo di architettura si adatta bene per questo sistema poiché permette di sviluppare e effettuare il deploy ogni componente in maniera indipendente e ottenere una netta separazione delle responsabilitá. Questo approccio inoltre permette al codice di essere facilmente mantenibile: ogni microservizio ha dipendenze minimali con gli altri che compongono il sistema, di conseguenza le modifiche saranno apportate solamente al microservizio di competenza, evitando di cambiare l'intera applicazione monolitica.

Il sistema é stato scomposto nelle seguenti componenti, identificate in fase di analisi:

- **Frontend service**. Questo componente é responsabile della fruizione dell'applicazione frontend all'utente. Gestisce le interazioni dell'utente e mostra il contenuto grafico;
- **Authentication service**. Questo componente ha la responsabilitá di gestire l'autenticazione dell'utente. Fornisce gli endpoint per permettere la facoltá di registrarsi, autenticarsi e disconnettersi;
- **Profile service**. Gestice le informazioni dell'utente e fornisce la possibilitá di visualizzare e/o modificare le informazioni personali;
- **Session service**. Componente core dell'applicazione, gestisce la sessione di streaming YouTube. Prevede la possibilitá di creare/entrare all'interno di una sessione e gestisce la sincronizzazione del video a fronte di interazioni di un utente. Inoltre, permette la fruizione di una chat attraverso cui gli utenti possono comunicare durante la riproduzione del video.

3.4 Design dei microservizi

Ogni microservizio che compone l'applicazione, eccetto il microservizio Frontend, adotta l'architettura Clean Architecture. Questa architettura permette di separare il dominio e la logica applicativa dai dettagli piú tecnici quali le tecnologie utilizzate per la comunicazione. L'architettura é composta da piú layer ed é basata sulla *Dependency Rule*: ogni layer puó dipendere solo dai layer al suo interno. I layer utilizzati sono i seguenti:

1. Domain. É il layer piú interno e contiene le definizioni del dominio applicativo. Non puó dipendere da nessun altro layer;
2. Application. Dipende dal Domain layer e funge da intermediario tra Infrastructure e Domain;
3. Infrastructure. Gestisce la comunicazione con gli altri microservizi;
4. Presentation. Contiene gli oggetti per serializzare/deserializzare messaggi verso/da altri microservizi.

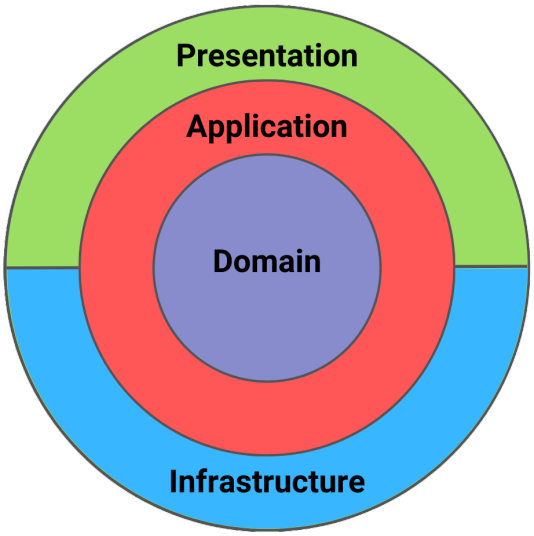


Figura 3.7: Clean Architecture

Capitolo 4

Implementazione

4.1 Frontend Service

Il microservizio di frontend é implementato utilizzando il framework VueJs in linguaggio TypeScript. Tra i numerosi strumenti che offre VueJs, sono stati sfruttati i Components e i Composables, i quali permettono una maggiore decomposizione e riusabilitá del codice.

A seguire viene mostrato il diagramma dei packages del microservizio.

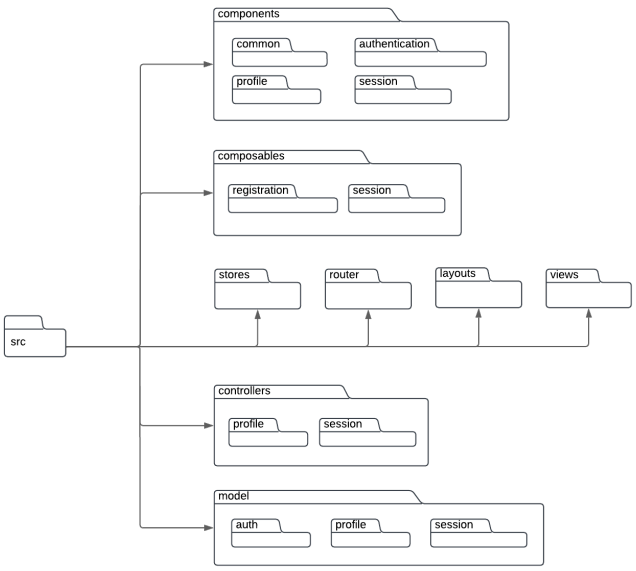


Figura 4.1: Diagramma packages - Frontend Service

Oltre ai due packages contenenti i Components e Composables, il microservizio comprende il package `model` il quale contiene le definizioni degli oggetti inviati/-ricevuti da altri microservizi e il package `controllers`, il quale é responsabile nel gestire la comunicazione con essi.

4.1.1 VueJs features

A seguire vengono esposti gli strumenti di VueJs che sono stati adottati per lo sviluppo del microservizio.

Components

I VueJs Components permettono di creare parti di UI indipendenti, in modo da poter essere riutilizzati in piú pagine dell'applicazione. Sono stati inseriti in quattro packages distinti:

1. `common`: contiene i VueJs Components utilizzati in diverse pagine dell'applicazione. In particolare é composto da Components che gestiscono la Navbar e il Footer;
2. `authentication`: contiene i Components per gestire le pagine di login e registrazione, come ad esempio i form `LoginForm` e `RegistrationForm`;
3. `profile`: per i componenti della pagina del profilo;
4. `session`: per le parti di UI che compongono la pagina della sessione, come ad esempio `SessionChat` che gestisce la chat e `SessionFrame` che gestisce il video YouTube.

Composables

I VueJs Composables sono funzioni riutilizzabili che incapsulano una logica stateful. In quanto tali, non sono semplici funzioni che accettano un input e producono un output ma dipendono anche dallo stato di un componente in un determinato istante.

Nel microservizio di frontend sono stati utilizzati dei Composables per validare l'input della password e dell'email, `useEmailValidator` e `usePasswordValidator` rispettivamente, e altri due per la gestione della sessione: `connectionErrors` e `connectToSession`. Questi ultimi sono risultati utili sia all'interno del Component

che contiene il Popup per creare la sessione (`CreateSessionPopup`) che dentro la view della sessione (`SessionView`).

Vue Router

Il Vue Router é stato utilizzato per gestire il routing client-side. Questo permette di passare dei valori in input a una pagina attraverso i Props o aggiungere permessi speciali. Inoltre fornisce la possibilitá di gestire rotte dinamiche. Nella pagina della sessione, ad esempio, é stata aggiunta una rotta dinamica per accedere alla sessione specifica e un permesso che costringe l'utente ad essere autenticato per accedere alla pagina.

Listing 4.1: Session route

```
{
      path: '/session/:sessionName',
      component: SessionLayout,
      meta: { requiresAuth: true },
      children: [
        {
          path: '',
          name: 'session',
          component: SessionView
        }
      ]
    },
```

4.1.2 Tecnologie

Pinia

Pinia é una libreria per la gestione dello stato. Vengono sfruttate le sue funzionalitá per gestire l'autenticazione dell'utente. Infatti, viene utilizzata per eseguire il refresh del token quando sta per scadere e, tramite la creazione di piú store, fornisce alle diverse pagine che compongono il frontend di reperire informazioni riguardo l'utente.

Attraverso questo é anche possibile implementare un blocco di accesso dalle pagine che richiedono l'autenticazione utente.

Socket IO Client

L'API Socket IO Client é stata utilizzata per la comunicazione con il Session Service. Questa é basata sul protocollo WebSocket per lo scambio di messaggi.

Il controller che gestisce la comunicazione con questo servizio é `SessionController`.

Axios

Axios é un client HTTP per NodeJs e i Browser.

É stato utilizzato per la comunicazione con Auth e Profile Service, i quali espongono un'API ReST.

Youtube Player API

L'embedding del video YouTube all'interno della pagina della sessione é realizzato tramite l'API Youtube Player.

Il Player permette di fare Play/Pause del video programmaticamente, e spostare la timeline attraverso un'operazione di Seek. Queste funzionalitá sono necessarie affinché il video possa essere sincronizzato in maniera esterna da Session Service.

Inoltre, l'API fornice la possibilitá di "ascoltare" i cambiamenti di stato del Player attraverso un listener, fondamentale per reagire ad un'azione effettuata dall'utente. I possibili stati sono:

- -1: unstarted
- 0: ended
- 1: playing
- 2: paused
- 3: buffering
- 4: video cued

4.2 Session Service

Questo microservizio si occupa di gestire le sessioni di streaming della piattaforma.

É scritta in linguaggio TypeScript e contiene un'API basata su WebSocket per interagire con essa, fornendo la possibilitá ai client di creare, accedere ed interagire con la sessione.

Prima che un client possa interagire con una sessione, la comunicazione con questo servizio deve passare attraverso una serie di stati, le cui transizioni avvengono alla ricezione di un messaggio specifico e a seguito di una validazione da parte del core logico.

L'insieme di stati che descrivono la comunicazione possono essere modellati attraverso un automa a stati finiti, mostrato qui di seguito.

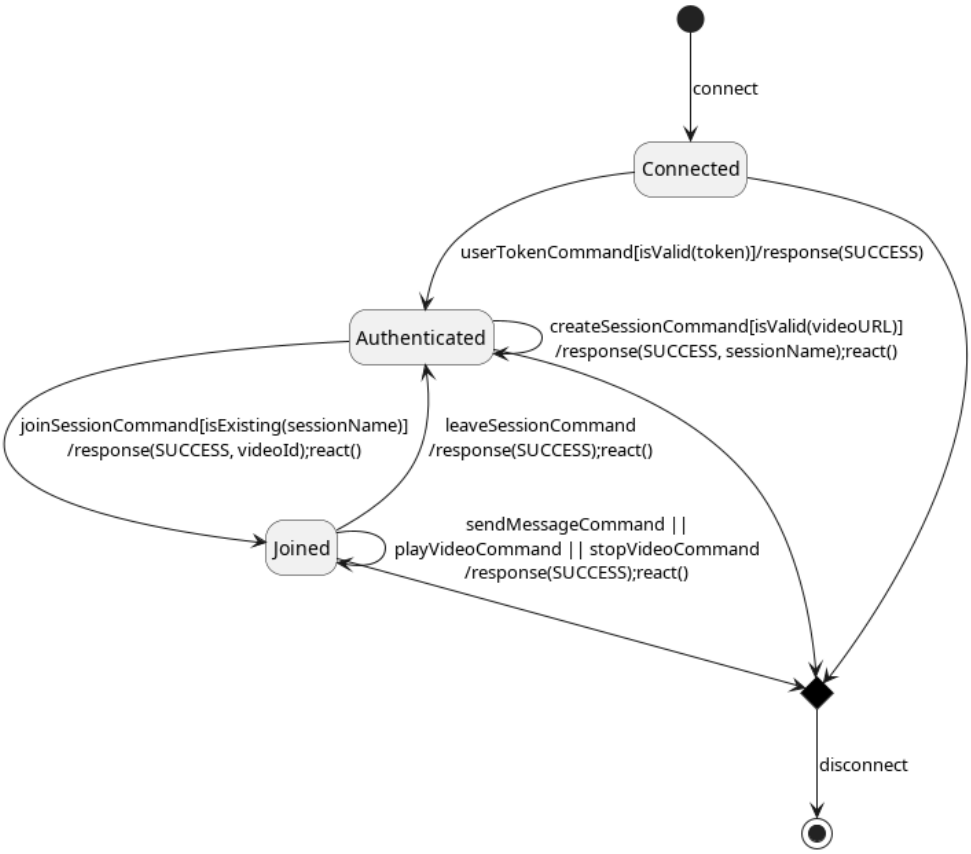


Figura 4.2: FSM Connessione

Notare come il client, prima di poter inviare messaggi nella chat o interagire con il video, debba passare prima nello stato Authenticated e poi Joined.

La specifica dei messaggi che vengono accettati dal microservizio é disponibile al seguente link: AsyncAPI specification.

4.2.1 Tecnologie

Socket IO

La tecnologia utilizzata per la comunicazione WebSocket é SocketIO.

Questa é basata ad eventi e permette di raggiungere uno scambio di messaggi real time.

É stato inoltre utilizzato il meccanismo di Room, tramite il quale é possibile spedire messaggi in modo semplice a tutti i client connessi a una determinata sessione.

4.3 Auth Service

Il microservizio di autenticazione gestisce la registrazione e l'accesso alla piattaforma da parte degli utenti. Gestisce inoltre la validazione e il refresh dei token di accesso a *Let's Stream It*. Implementato in Typescript seguendo la Clean Architecture 3.7. Espone un'API REST che viene utilizzata da tutti gli altri microservizi.

4.3.1 Tecnologie

JSON Web Tokens

Il servizio di autenticazione genera e valida i JSON Web Tokens (JWT) degli utenti. L'utilizzo di questa tecnologia permette di firmare digitalmente il token, rendendolo sicuro, verificabile ed affidabile. Per firmare digitalmente il token è necessaria una chiave che viene salvata nelle variabili d'ambiente del microservizio. La chiave non viene condivisa con gli altri microservizi, ma sono questi ultimi a contattare il servizio di autenticazione per verificare la correttezza di un dato token, migliorando la sicurezza dell'intero sistema.

Express

Il microservizio di autenticazione è realizzato utilizzando il framework Express, standard de facto per la realizzazione di server web con Node.js è la scelta ovvia per questo semplice servizio.

Bcrypt

Per salvare le password degli utenti in modo sicuro viene utilizzata la libreria Bcrypt. L'utilizzo di questa libreria permette di effettuare l'hashing delle password prima di salvarle sul database. In questo modo non è possibile a partire dall'hash ottenere la password originale; è però possibile confrontare una password testuale con l'hash per verificare che queste combacino.

Mongoose

Questa implementazione del servizio di autenticazione sfrutta un database MongoDB per salvare le informazioni sugli utenti. Per la comunicazione con il database è stata utilizzata la libreria Mongoose.

4.4 Profile Service

Questo microservizio si occupa della gestione del profilo degli utenti di *Let's Stream It*. È realizzato in Scala con Akka ed espone la seguente API REST.

4.4.1 Tecnologie

Akka HTTP

Il servizio si basa sul framework Akka HTTP. Si tratta di un framework costruito su Akka, che lo rende, grazie al sistema di attori, un'ottima soluzione per la realizzazione di applicazioni distribuite che necessitano di performance elevate.

MongoDB scala driver

Questa versione del profile service utilizza per il salvataggio di informazioni un database MongoDB. Per la comunicazione con il database viene utilizzata la libreria

MongoDB scala driver, che offre API scala idiomatiche, permettendo interazioni reattive, asincrone e non bloccanti con MongoDB.

Capitolo 5

Test

Per mantenere elevata la qualità del codice ed assicurare il corretto funzionamento delle API esposte e della logica di business sono stati realizzate diverse tipologie di test.

5.1 Unit tests

In ogni microservizio sono stati implementati unit tests per andare a testare la correttezza della business logic. `Session` e `auth` service sono testati utilizzando Mocha e Chai, mentre `profile` service è stato testato utilizzando ScalaTest e Akka Route TestKit.

5.2 Test end-to-end

I test end-to-end verificano che le componenti di un sistema funzionino correttamente in situazioni reali, andando a simulare l'utilizzo che un utente finale farebbe della piattaforma. Presenti nel repository bootstrap, nella cartella `tests`, sono scritti in Typescript ed utilizzano la libreria Playwright andando a simulare le interazioni più comuni che un utente ha all'interno della piattaforma.

I test controllano il funzionamento dei seguenti scenari:

- L'utente può registrarsi ed accedere alla piattaforma;
- L'utente può creare ed entrare in una sessione;
- L'utente può visualizzare ed aggiornare il suo profilo.

I test vengono eseguiti su 3 diversi browser:

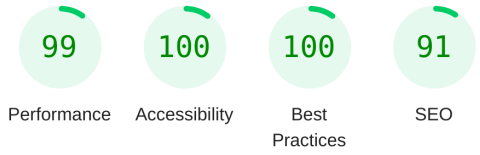
- Chromium
- Firefox
- Webkit

Vengono eseguiti ogni volta che uno dei microservizi della piattaforma rilascia una nuova versione, in modo da assicurare il corretto funzionamento di *Let's Stream It*.

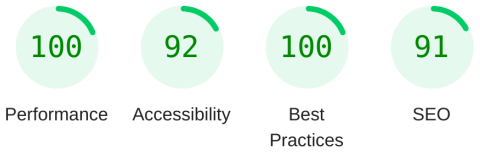
5.3 Qualità pagine web

Per testare la qualità delle pagine che compongono l'applicazione, é stato utilizzato Lighthouse, un tool automatico per Google Chrome che permette di testare le Performance, l'Accessibilitá, l'utilizzo delle Best practices, e delle SEO (Search Engine Optimizations).

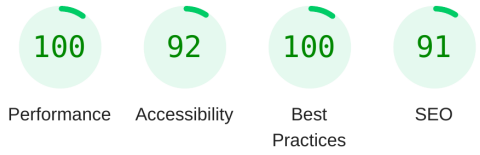
A seguito di risultati di qualitá medi, sono state effettuate delle modifiche, andando sopratutto a puntare ad una migliore Accessibilitá. La versione finale dell'applicazione risulta ottima in termini di qualitá. A seguire vengono mostrati i report di Lighthouse eseguiti su dispositivo Desktop per ciascuna pagina del frontend.



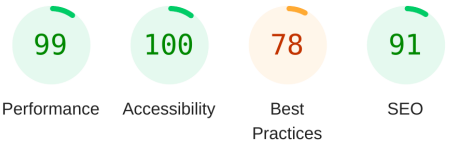
(a) Home page



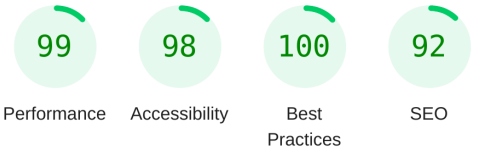
(b) Sign up page



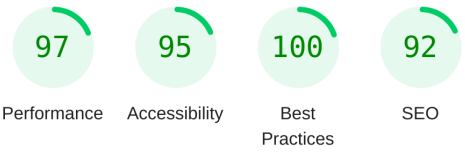
(c) Sign in page



(d) Session page



(e) Profile page



(f) About page

Figura 5.1: Lighthouse Report

Capitolo 6

Deployment

L'esecuzione della piattaforma *Let's Stream It* è possibile in due diverse modalità:

- Eseguendo ogni microservizio localmente, seguendo le istruzioni riportate nel `README` che si trova all'interno di ogni repository (Session Service, Profile Service, Auth Service e Frontend Service). Inoltre eseguendo un container MongoDB sulla porta 27017;
- Effettuando il deploy di tutti i servizi assieme tramite docker compose, per fare questo è sufficiente navigare al repository di bootstrap e seguire le indicazioni riportate nel `README`.

In entrambi i casi i microservizi hanno alcune intra-dipendenze, come mostrato sotto.

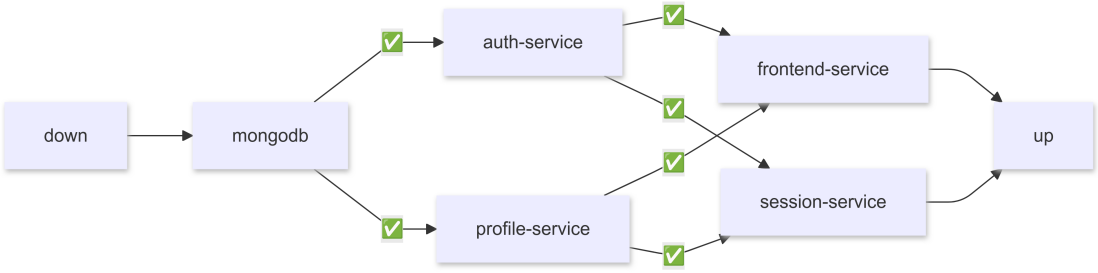


Figura 6.1: Diagramma di deployment

Utilizzando il file `docker-compose` si garantisce l'ordine di avvio dei servizi, grazie all'utilizzo degli attributi `healthchecks` e `depends_on`.

Capitolo 7

Conclusioni

Questo progetto ci ha permesso di approfondire diversi aspetti della progettazione e realizzazione di piattaforme web. L'utilizzo di un framework moderno e potente come Vue.js, assieme ad una pipeline di ci/cd robusta, ci ha permesso di iterare rapidamente e trovare soluzioni eleganti ai problemi incontrati.
La piattaforma *Let's Stream It* può essere migliorata su diversi aspetti:

- La possibilità di diversificare le fonti video disponibili, aggiungendone altre oltre a YouTube;
- La possibilità di scalare i microservizi, in particolare il `session service` permettendo così alla piattaforma di gestire meglio un numero maggiore di utenti. Si puó valutare questa opzione andando anche a verificare le feature offerte da SocketIO, come ad esempio le funzionalitá di load balancing;
- Nel `profile service` si possono aggiungere alcune informazioni al profilo utente, oppure la possibilità di filtrare i video visualizzati per categorie, tag, autori, ecc.;
- Effettuare una valutazione di sicurezza della piattaforma ed eventualmente proporre miglioramenti per i punti più critici.