

Let's Stream It

Applicazioni e Servizi Web 22/23

Simone Ceredi, Luca Fabri

Alma Mater Studiorum - Universitá di Bologna

15 gennaio 2025

Indice

1 Introduzione

2 Design

- Mockup
- Architettura
- Core

3 Implementazione

4 Validazione e Testing

5 Deployment

6 GUI

- Desktop
- Mobile

Introduzione

Let's Stream It è un'applicazione a microservizi che consente ad un gruppo di utenti di guardare video YouTube in maniera condivisa e sincronizzata.

Fornisce un'interfaccia web user friendly e responsive per permettere all'utente di creare e effettuare il join nelle sessioni in maniera facile ed intuitiva. Inoltre, contiene una chat tramite cui è possibile scambiarsi messaggi durante la riproduzione del video.

Indice

1 Introduzione

2 Design

- Mockup
- Architettura
- Core

3 Implementazione

4 Validazione e Testing

5 Deployment

6 GUI

- Desktop
- Mobile

Mockup - Home



Figura: Mockup Home page

Mockup - Creazione Sessione

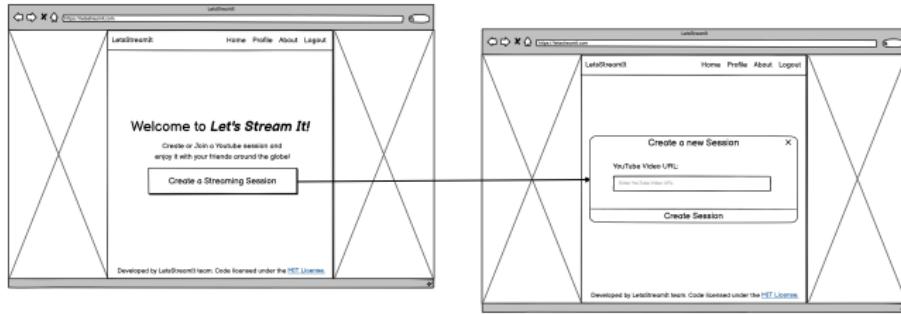


Figura: Mockup Creazione Sessione

Mockup - Login e Registrazione

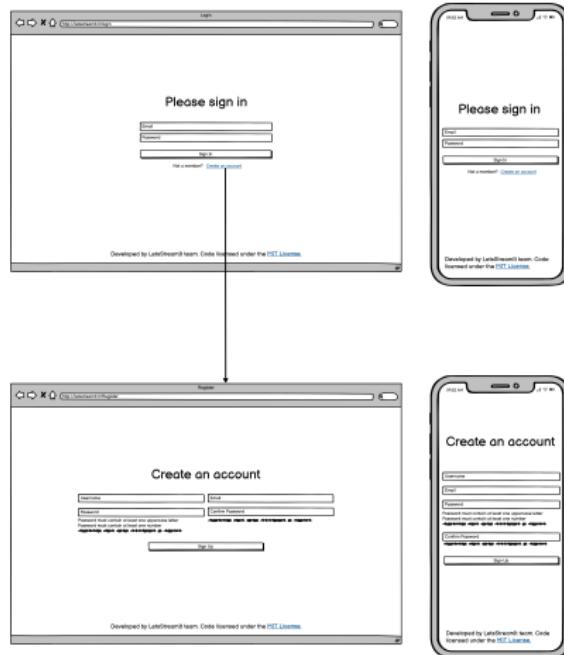
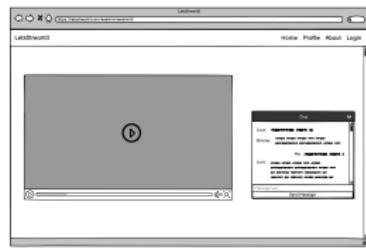
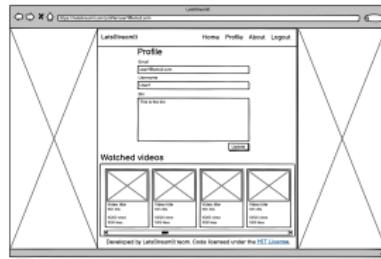


Figura: Mockups Login e Registrazione

Mockup - Sessione e Profilo



(a) Mockup Sessione



(b) Mockup Profilo



Architettura

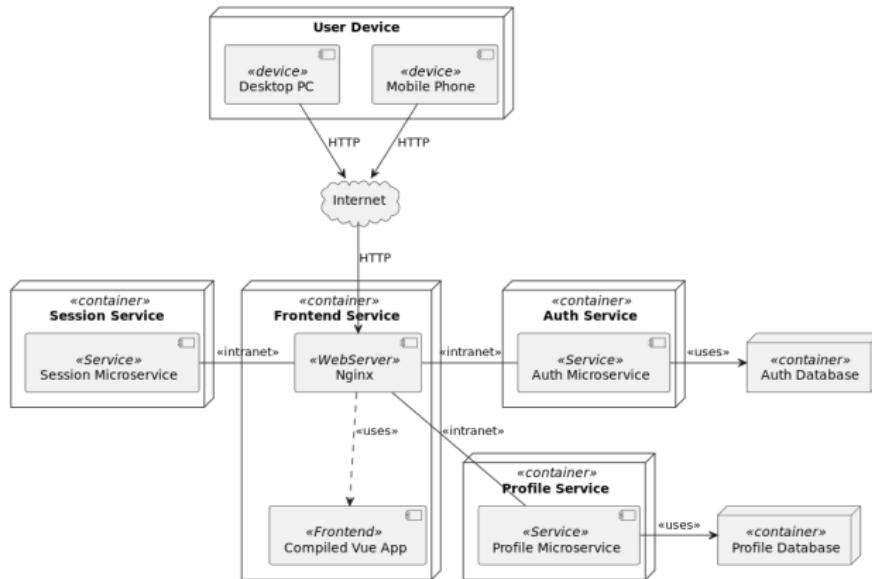


Figura: Architettura del sistema

Core

Il core di *Let's Stream It* è la **Sessione YouTube**, gestita da:

- **Frontend Service.** Durante lo streaming, scarica localmente la porzione video da YouTube e comunica con il Session Service a fronte di iterazioni dell'utente locale / utente interno alla sessione per sincronizzare il video; Inoltre, invia e riceve messaggi dalla chat;
- **Session Service.** Responsabile della gestione delle sessioni YouTube. Sincronizza i video degli utenti di una sessione a fronte di interazioni di play/pause/seek; Mantiene aggiornate le chat delle sessioni.

Indice

1 Introduzione

2 Design

- Mockup
- Architettura
- Core

3 Implementazione

4 Validazione e Testing

5 Deployment

6 GUI

- Desktop
- Mobile

Implementazione

- **Frontend Service.** Realizzato in Typescript mediante l'utilizzo del framework VueJs. Sfrutta alcune delle feature che offre VueJs, tra cui i Components, Composables ed il Routing.
Comunica con il Profile Service e Auth Service attraverso la libreria [Axios](#), con il Session Service tramite l'API client di [Socket IO](#).
Altre librerie utilizzate sono: [Player API](#) di Youtube, Pinia.

Implementazione

- **Session Service.** Implementato in NodeJs, comunica con i client tramite WebSocket. A tale scopo è stata sfruttata la l'API [Socket IO](#);
- **Auth Service.** Servizio realizzato in NodeJs con il framework ExpressJs. Gestisce l'autenticazione tramite i Json Web Tokens ([JWT](#)). Altre librerie utilizzate: [Mongoose](#) e [Bcrypt](#);
- **Profile Service.** Implementato in Scala sfruttando la libreria [Akka HTTP](#) per la fruizione dell'API ReST e [MongoDB scala driver](#) per la gestione del DBMS.

Indice

1 Introduzione

2 Design

- Mockup
- Architettura
- Core

3 Implementazione

4 Validazione e Testing

5 Deployment

6 GUI

- Desktop
- Mobile

Validazione e Testing

- Unit ed integration (mocha, chai, scalatest)
- End to end (playwright)
- Performance, accessibilità, SEO, best practices (lighthouse)
- ci-cd pipeline (github actions)

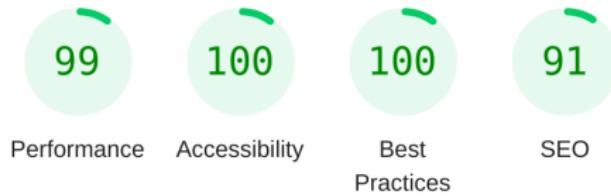


Figura: Home page lighthouse report

Indice

1 Introduzione

2 Design

- Mockup
- Architettura
- Core

3 Implementazione

4 Validazione e Testing

5 Deployment

6 GUI

- Desktop
- Mobile

Deployment

- Docker Compose
- Healthchecks

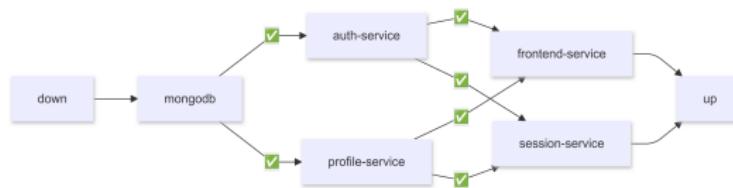


Figura: Diagramma di deploy

Indice

1 Introduzione

2 Design

- Mockup
- Architettura
- Core

3 Implementazione

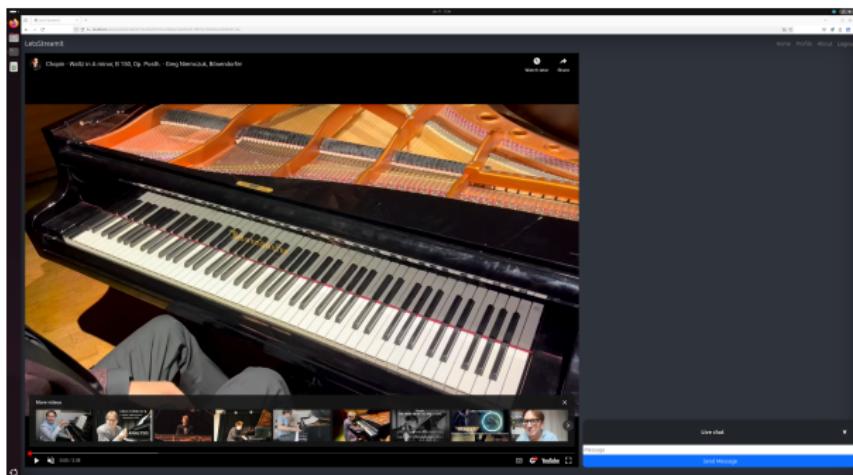
4 Validazione e Testing

5 Deployment

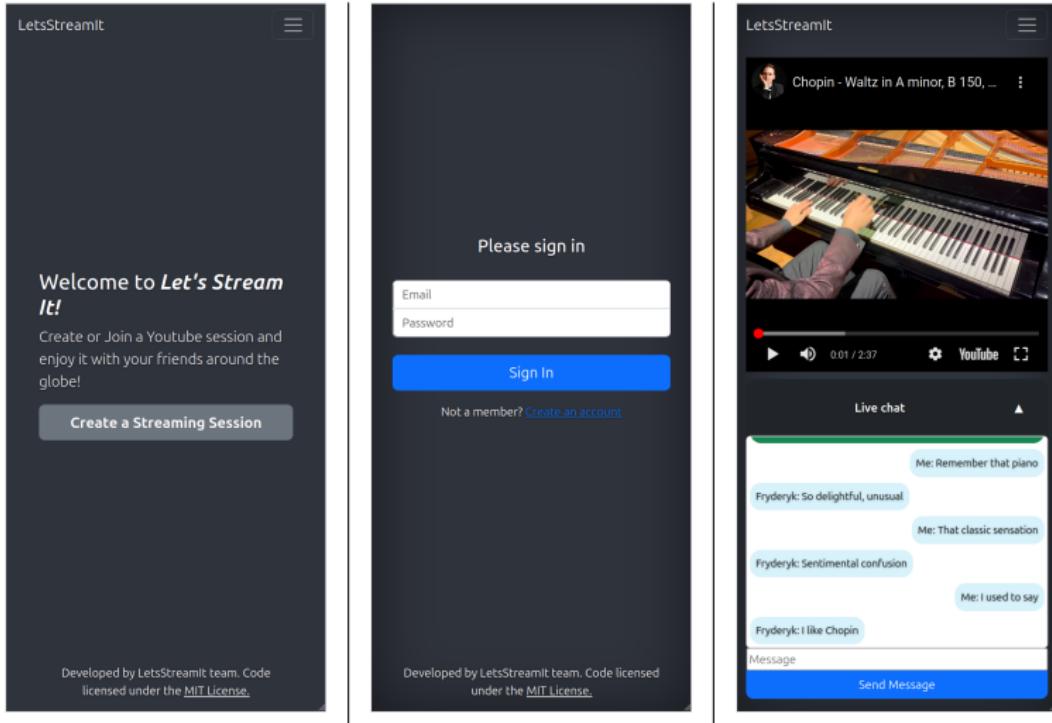
6 GUI

- Desktop
- Mobile

Desktop



Mobile



Un video di demo è disponibile al seguente [link](#)