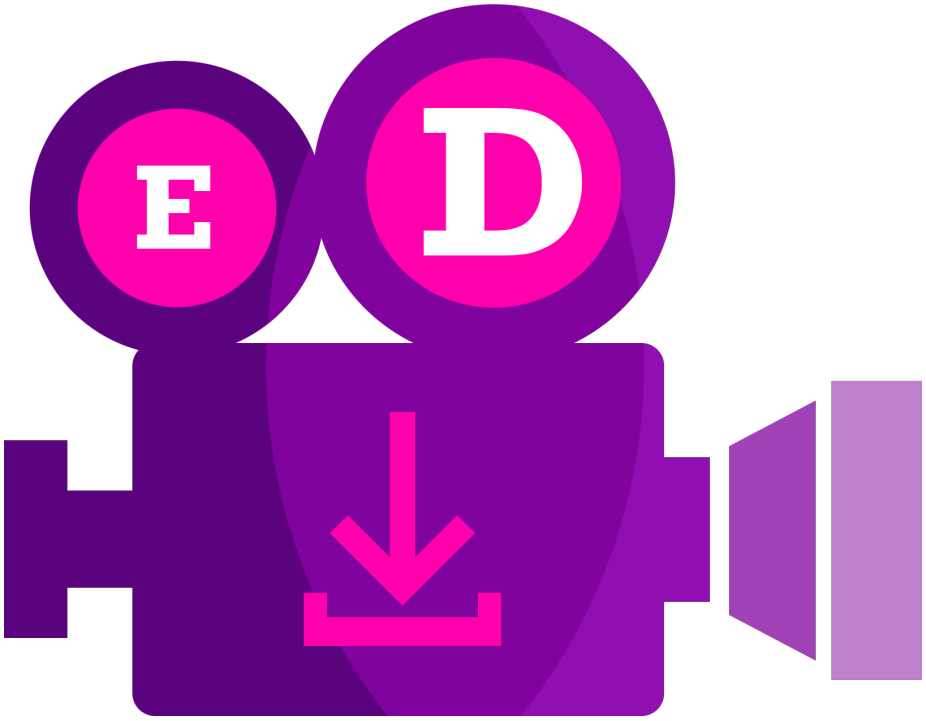
Reti di Calcolatori

Emanuele D’Ambrosio 0124002587
Domenico Zeno 0124002479

15 Febbraio 2024



1 Traccia

Creare un sito web che permetta la visualizzazione di film con i seguenti punti:
1. Creare una chat tra due o più utenti utilizzando websocket per mettere in comunicazione tutti gli utenti che si trovano sulla pagina dello stesso film:
- Lato js usare le websocket tramite libreria;
- Lato backend, che immagino sia in python, implementare manualmente le websockets (sono delle semplici sockets, ma con degli header http particolari);
Prendere come riferimento il seguente codice:
https://github.com/websocket-client/websocket-client/tree/master.

2. Invece di scaricare il video (o altro dato multimediale) direttamente tramite http classico, il browser ci fornirà un file testuale, che chiameremo metasource, in cui ci sono le informazioni per la connessione al server che rimarrà in ascolto con una socket.
A questo punto scriverete un piccolo client che legge le informazioni contenute nel file metasource, contatta il server ed avvia il download.
Usate il linguaggio che preferite, inoltre per l'invio del file, non usate l'invio diretto di bytes perché è più complesso, ma usate le funzioni per l'invio del file disponibili.

Ad esempio, in C avete il comando sendfile:
(https://man7.org/linux/man-pages/man2/sendfile.2.html)

In python lo stesso:
(https://docs.python.org/3/library/socket.html)

oppure tramite l'esempio che trovate qui:
(https://pythonassets.com/posts/send-file-via-socket/)

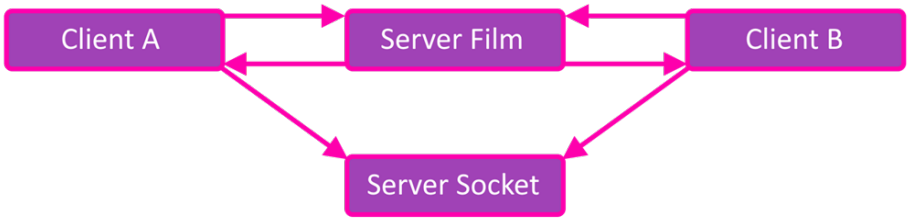
2 Punti Chiave

- Descrizione del Progetto
- Descrizione e Schema dell'Architettura
- Dettagli Implementativi dei Client/Server
- Parti Rilevanti del Codice Sviluppato
- Manuale Utente con le Istruzioni di Compilazione ed Esecuzione
- Video per Maggiori Chiarimenti

3 Descrizione del Progetto

Si vuole realizzare una Pagina Web per lo sviluppo di una piattaforma che consente agli utenti di guardare film. Un utente all'interno della Pagina potrà effettuare varie operazioni cliccando sulle diverse impostazioni presenti nella parte alta del menu.
All'interno di esso saranno presenti diversi pulsanti che daranno la possibilità all'utente di visualizzare tutti i film presenti nella piattaforma, effettuare le operazioni di accesso e di poter ricercare un film attraverso il nome oppure applicando il filtro per generi.
Un utente dopo essersi registrato potrà usufruire di nuove impostazioni e vantaggi che la piattaforma fornirà. I vari utenti dopo aver eseguito l'autenticazione e selezionato il film altre operazioni aggiuntive:
1. Cliccare sul pulsante **Commenta** per scambiare idee e opinioni con altri utenti sul film visto;
2. Cliccare sul pulsante di **Download** per scaricare il film in modo da permettere all'utente di poter visionare il film anche in un secondo momento;

4 Schema dell'Architettura



5 Descrizione dell'Architettura

Il **Client A** dopo essersi connesso al **Server Film** potrà svolgere l'operazione di Login o Registrazione.
Dopo aver eseguito l'autenticazione, durante la visione del video (all'interno della scheda del film), ci sarà il pulsante "**Commenta**" che ci porterà al **Server Socket**.
Nel **Server Socket** potremo andare a visionare tramite la gestione dell'evento message e la route get-messages i diversi messaggi/commenti dei **Client** riguardanti il film selezionato che verrà salvato nei vari Chat Log (identificati dal Codice Film).
I **Client** autenticati potranno quindi partecipare ai commenti e discussioni attraverso un loro messaggio. All'interno della pagina di Chat, il Client potrà

usufruire del pulsante di Download per scaricare il film selezionato oggetto della “discussione”.
Lo scaricamento del film avviene attraverso la route metasource la quale invierà il file metasource.txt che conterrà i dati del film. Nel momento in cui il client ottiene i dati del film, avviene il Download vero e proprio tramite la route Download. Alla fine del caricamento, l’utente si ritroverà il file del film che è stato gestito come un file con estensione txt.

6 Dettagli Implementativi del Client/Server

Linguaggi usati:
Python, Html, Css, Javascript.

La decisione di concentrarsi su questi linguaggi è stata motivata da diversi fattori legati alla familiarità che abbiamo con questi ultimi.
Html ci permette di rappresentare in modo semplice e rapido le interfacce di cui abbiamo bisogno per la realizzazione del progetto insieme al **Css** che ci supporta nelle operazioni di abbellimento.
Il **Javascript** è stato incluso principalmente per la possibilità di offrirci supporto con le funzioni e per effettuare la connessione con l’API.
Il **Python** ci permette di implementare l’API, quindi si occupa di prendere le richieste fatte dal Javascript ed elaborarle insieme al database . Ovviamente la scelta è ricaduta su questo linguaggio di programmazione data la sua semplice sintassi e la sua elevata curva di apprendimento.

Framework:
Flask.

La scelta di Flask è stata dettata dal fatto che è un Framework Web leggero, flessibile e scritto in Python.
Flask ci offre inoltre, un supporto integrato per l’implementazione delle WebSocket.

Librerie:
Gevent.

Gevent è stato utilizzato dal momento che permette la programmazione concorrente in Python.

Moduli Standard di Python:
Os, Datetime, Json.

• Os è un modulo di Python per interagire con il sistema operativo;
• Datetime serve a prendere il momento dell’invio del messaggio;

• Json è un formato di dati leggero per lo scambio di dati;

La gestione della connessione è implementata utilizzando Flask-SocketIO. Flask-SocketIO è un'estensione di Flask che fornisce il supporto per WebSocket in applicazioni Flask. La gestione della connessione WebSocket è definita dalla funzione:
`handle_message`
Quando un client emette un messaggio WebSocket con il tipo 'message', questa funzione viene eseguita per gestire il messaggio e rispondere di conseguenza.

7 Codice

7.1 Codice Sock.py

```
film_connections = {}

async def send_message(websocket, cod_film, username, message):
    payload = {'action': 'message', 'cod_film': cod_film, 'username': username, 'message': m
    await websocket.send(json.dumps(payload))

async def save_to_file(cod_film, username, message):
    timestamp = datetime.datetime.utcnow().strftime('%Y-%m-%d %H:%M:%S')
    filename = f'chat_log{cod_film}.txt'
    with open(filename, 'a') as file:
        file.write(f'{username} {timestamp} {message}\n')

async def handle_connection(websocket, path):
    try:
        async for message in websocket:
            data = json.loads(message)
            username = data.get('username', 'UnknownUser')
            cod_film = data.get('cod_film', None)

            if data['action'] == 'join':
                if cod_film not in film_connections:
                    film_connections[cod_film] = set()

                film_connections[cod_film].add(websocket)

            elif data['action'] == 'message':
                message = data['message']
                await save_to_file(cod_film, username, message)
                await asyncio.gather(
                    *[send_message(ws, cod_film, username, message) for ws in film_connectio
```

```
                )

    except websockets.exceptions.ConnectionClosed:
        print(f"Connection closed: {websocket.remote_address}")
    finally:
        for cod_film, connections in film_connections.items():
            if websocket in connections:
                connections.remove(websocket)
                if not connections:
                    del film_connections[cod_film]

start_server = websockets.serve(handle_connection, "localhost", 8080)

asyncio.get_event_loop().run_until_complete(start_server)
asyncio.get_event_loop().run_forever()
```

7.2 Codice mysocket.py

```
CHAT_LOG_DIR = "chat_logs"
app.config["CHAT_LOG_DIR"] = CHAT_LOG_DIR

@app.route("/join")
def join():
    cod_film = request.args.get("cod_film")
    return jsonify({"status": "success", "message": f"Joined film {cod_film}"})

@app.route("/request_metasource")
def request_metasource():
    cod_film = request.args.get("cod_film")
    metasource_filename = f'metasource{cod_film}.txt'
    if os.path.exists(metasource_filename):
        with open(metasource_filename, 'r') as metasource_file:
            metasource_content = metasource_file.read()

        print(f"Metasource received: {metasource_content}")
        return jsonify({"status": "success", "data": metasource_content})
    else:
        return jsonify({"status": "error", "message": f"Metasource not found for cod_film: {

@app.route("/download")
def download_file():
    file = request.args.get("file")
    print(file)
    path = f"chat_log{file}.txt"
    print(path)
```

```
    try:
        return send_file(path, as_attachment=True)
    except FileNotFoundError:
        return jsonify({"status": "error", "message": "File not found"})

@app.route("/get_chat_log")
def get_chat_log():
    cod_film = request.args.get("cod_film")
    filename = f"chat_log{cod_film}.txt"
    try:
        with open(filename, 'r') as file:
            chat_log_content = file.read()
        return jsonify({"status": "success", "content": chat_log_content})
    except FileNotFoundError:
        return jsonify({"status": "error", "message": "Chat log not found"})

# Avvia l'app Flask
if __name__ == "__main__":
    if not os.path.exists(CHAT_LOG_DIR):
        os.makedirs(CHAT_LOG_DIR)

    app.run(port=5000)
```

7.3 Codice html

```
async function initWebSocket() {
    await new Promise(async (resolve) => {
        socket.addEventListener('open', async (event) => {
            getcod_film();

            if (cod_film) {
                console.log(cod_film);

                // Invia il messaggio di join al server WebSocket
                socket.send(JSON.stringify({ action: 'join', cod_film }));

                // Invia il messaggio di join al server Flask
                await fetch(`http://localhost:5000/join?cod_film=${cod_film}`);

                // Aggiorna il log della chat lato client
                await fetchAndDisplayChatLog();

            }
            resolve();
        });
```

```
    });

    // Event listener per gestire i messaggi in arrivo durante l'esecuzione
    socket.addEventListener('message', async (event) => {
        const data = JSON.parse(event.data);
        console.log('Received message:', data);

        // Aggiorna il log della chat lato client ogni volta che arriva un nuovo messaggio
        await fetchAndDisplayChatLog();
    });
}

// Funzione per richiedere e visualizzare il log della chat dal server Flask
async function fetchAndDisplayChatLog() {
    const response = await fetch(`http://localhost:5000/get_chat_log?cod_film=${cod_film}`);
    const responseData = await response.json();
    if (responseData.status === "success") {
        const chatLogContent = responseData.content;

        const chatMessages = document.getElementById('chatMessages');
        chatMessages.innerHTML = chatLogContent
            .split('\n')
            .map(line => `<p>${line}</p>`)
            .join('');

    }
}

function sendMessage() {
    const messageInput = document.getElementById('messageInput');
    const message = messageInput.value;
    const payload = { action: 'message', cod_film, username, message };

    socket.send(JSON.stringify(payload));

    messageInput.value = '';
}

document.addEventListener('DOMContentLoaded', () => {
    initWebSocket();

    const messageInput = document.getElementById('messageInput');
    messageInput.addEventListener('keydown', (event) => {
        if (event.key === 'Enter') {
            sendMessage();
        }
```

```
    });
})
async function requestMetasource() {
            // Invia una richiesta al server per ottenere il metasource
            const response = await fetch(`http://localhost:5000/request_metasource?cod_film=
            const responseData = await response.json();

            if (responseData.status === "success") {
                console.log('Metasource received:', responseData.data);
                downloadFile(responseData.data);
            }
        }
        async function downloadFile(dati) {
            const response = await fetch(`http://localhost:5000/download?file=${dati}`);
            const blob = await response.blob();
            // Crea un oggetto URL dal blob e crea un link invisibile per il download
            const url = window.URL.createObjectURL(blob);
            const a = document.createElement("a");
            a.style.display = "none";
            a.href = url;
            a.download = `chat_log${dati}.txt`;
            document.body.appendChild(a);
            a.click();

            // Rimuovi il link dal documento
            document.body.removeChild(a);

            // Rilascia l'URL dell'oggetto
            window.URL.revokeObjectURL(url);
        }
```

7.4 Spiegazione Codice

Il file HTML contiene del codice JavaScript.
Il Javascript si occupa di interagire con il server WebSocket utilizzando l'API WebSocket in modo da effettuare richieste HTTP al server Flask.
Le funzioni JavaScript nel file HTML includono initWebSocket, fetchAndDisplayChatLog, sendMessage, requestMetasource e downloadFile.

Queste funzioni sono responsabili dell'inizializzazione della connessione WebSocket, del recupero e della visualizzazione dei registri di chat, dell'invio di messaggi, della richiesta e del download di contenuti metasource.

8 Manuale

Manuale Avvio
Per avviare il server bisogna utilizzare l'istruzione "**Python mysocket.py , python -m http.server e python sock.py**" da terminale di comando in modo da avviare l'API.

Manuale Utente
L'utente una volta arrivato alla Pagina Home potrà decidere se guardare film senza Registrarsi/Loggarsi oppure accedere e commentare il film insieme a tutti i visitatori del sito.
La Home Page è la pagina di benvenuto su cui l'utente si ritroverà al suo arrivo. La pagina si apre con uno SlideShow e una Navigation Bar.

La Navigation Bar contiene i seguenti pulsanti:
• Home;
• Film;
• Genere;
• Accesso;
• Barra di Ricerca e Pulsante di Ricerca;

Il pulsante **Home** permette all'utente di ritornare alla pagina di Benvenuto.
Il pulsante **Film** permette all'utente di visionare a griglia tutti i film contenuti all'interno del database. A questo punto, basterà cliccare sulla copertina del film designato per aprire la schermata di visione del film.
Il pulsante **Genere** permette all'utente di applicare una Ricerca per Categoria, spuntando le varie caselle con il genere selezionato, l'utente potrà usare i filtri per svolgere l'operazione di **Ricerca Dettagliata**.
Il pulsante **Accesso** porterà l'utente in una pagina contenente 2 pulsanti con le relative intestazioni: **Login** e **Registrazione**.
Il pulsante **Registrazione** porterà l'utente ad una formbox da compilare per potersi registrare alla piattaforma, basterà quindi inserire i campi specificati per completare l'operazione.
Il pulsante **Login** porterà l'utente alla formbox dell'autenticazione, ovvero ad una tabella molto simile a quella di registrazione ma con un numero di campi ridotti, quindi l'utente dovrà solo inserire lo username e la password per farsi riconoscere dalla piattaforma.
La **Barra di Ricerc**a e il pulsante di **Ricerca** permetteranno all'utente di effettuare l'operazione di Ricerca per nome di un determinato film all'interno della piattaforma.

Comandi Supplementari
Solo dopo aver effettuato l'operazione di autenticazione, gli utenti potranno utilizzare queste altre funzionalità offerte dalla piattaforma. Le funzioni sono state posizionate al di sotto della schermata dov'è presente il titolo del film:

• Vota;
• Commenta;
• Download;

Il Pulsanti **Vota** (ovvero quei cinque pulsanti a forma di stella), permettono all'utente di rilasciare un breve feedback sul film che ha appena visionato.
Tuttavia, nel caso in cui l'utente volesse lasciare un commento più elaborato, allora gli basterà cliccare sul pulsante **Commenta** per confrontarsi con gli altri utenti.
Il pulsante **Commenta** permette all'utente di finire all'interno della Pagina di Chat per poter conversare e scambiare opinioni sul film visionato.
Il pulsante **Download** permette all'utente di scaricare il film selezionato.

9 Video per Maggiori Chiarimenti

Questo video aiuta a comprendere il risultato finale della nostra pagina web: https://youtu.be/mArhnPWjiFM