Real Time Collaboration Platform

Progetto finale di Architettura dei Sistemi Distribuiti

Ignazio Emanuele Piccichè
Mattia Castiello
Michela Di Simone

Anno accademico 2023/2024

Indice

1 Introduzione

Il progetto mira a sviluppare una piattaforma di collaborazione in tempo reale che non solo supporti la modifica condivisa dei documenti, ma che sia anche resiliente alle partizioni di rete, garantendo così un'esperienza utente affidabile.

1.1 Obiettivi del progetto

Il principale obiettivo del progetto è creare un editor collaborativo in grado di:

- Consentire a più utenti di modificare lo stesso documento simultaneamente, sfruttando l'utilizzo delle WebSocket
- Mantenere coerenza dei dati anche in presenza di partizioni di rete, utilizzando la tecnica *Conflict-free Replicated Data Types* (CRDTs)
- Gestire le partizioni di rete in modo da assicurare che tutte le modifiche vengano correttamente integrate una volta risolta la partizione
- Notificare in modo efficace la presenza degli utenti, informando chi è attualmente connesso al documento condiviso

1.2 Sfide

Per verificare la robustezza della piattaforma, sono stati condotti esperimenti mirati che includono:

- La simulazione delle partizioni di rete per testare la capacità del sistema di riconciliare le modifiche
- L'analisi dell'impatto delle diverse strategie di risoluzione dei conflitti sulla coerenza e la latenza del sistema

In conclusione, il progetto non solo esplorerà la complessità della collaborazione in tempo reale, ma cercherà anche di fornire soluzioni pratiche e innovative per garantire la continuità e l'integrità del lavoro condiviso in scenari di rete variabili.

2 Back-End

All'interno del *FileServer.py* che abbiamo sviluppato gestiamo la collaborazione in tempo reale su un documento condiviso tramite *WebSocket.* Consentendo a più client di connettersi e interagire simultaneamente con un file, aggiornandolo e mantenendo la consistenza tra le versioni sui diversi client.

L'applicazione utilizza un algoritmo di CRDT (*Conflict-free Replicated Data Type*) per risolvere conflitti e garantire che tutti i client visualizzino la stessa versione aggiornata del documento.

2.1 Gestione delle Connessioni

Questa sezione descrive come il server gestisce le connessioni WebSocket, mantenendo un registro dei client connessi e aggiornando gli altri client quando un nuovo utente si connette o si disconnette.

2.1.1 Metodo *file_server*

- Questo metodo gestisce la connessione e la comunicazione con i client. All'avvio della connessione, il server richiede il nome utente del client e lo aggiunge alla lista degli utenti connessi. In caso di disconnessione, il server rimuove il client dalla lista e informa gli altri utenti della partenza.

2.2 Gestione dei Messaggi

In questa sezione viene spiegato come il server gestisce e trasmette i messaggi tra i client. Questo include sia i messaggi di aggiornamento del documento che i messaggi di stato del sistema.

2.2.1 Metodo *send_to_all*

- Questo metodo invia un messaggio a tutti i client connessi. È utilizzato per notificare tutti gli utenti riguardo a eventi come l'ingresso o l'uscita di un utente.

2.2.2 Metodo *broadcast*

- Simile a *send_to_all,* ma specificamente progettato per inviare aggiornamenti del contenuto del file a tutti i client. Questo metodo

utilizza JSON per serializzare i messaggi, facilitando l'invio di dati strutturati.

2.2.3 Metodo *send_user_list*

- Questo metodo aggiorna tutti i client con la lista attuale degli utenti connessi, mantenendo la visibilità sui partecipanti.

2.3 Gestione del Documento

Questa sezione esplora come il server carica, salva e aggiorna il contenuto del file condiviso.

2.3.1 Metodo *load_file*

- Carica il contenuto del file condiviso dal file system, se esiste. Se il file non esiste, restituisce una stringa vuota.

2.3.2 Metodo *save_file*

- Salva il contenuto del file condiviso nel file system, aggiornando il file con le modifiche recenti.

2.3.3 Metodo *network_partition_consistency*

- In fase di riconnessione di un utente, risolve i conflitti tra versioni del file ottenute da diverse fonti (utenti), utilizzando il metodo *SequenceMatcher* della libreria *difflib* per allineare i cambiamenti.

2.4 Gestione delle Operazioni CRDT

Questa sezione tratta il modo in cui il server utilizza il CRDT per applicare e sincronizzare le modifiche al documento.

2.4.1 Metodo *crdt_operations*

- Calcola le operazioni necessarie per trasformare un testo vecchio in uno nuovo, generando una lista di operazioni che rappresentano aggiunte, cancellazioni o modifiche. Queste operazioni sono poi applicate alla classe CRDT per mantenere la coerenza tra i client.

3 Front-End

4 Docker

5 Compute Engine (GCP)

6 Conclusioni

7 Sviluppi futuri

Nonostante il progetto attuale rappresenti una solida base per una piattaforma di collaborazione condivisa, abbiamo identificato alcune aree di miglioramento ed eventuali funzionalità avanzate che potrebbero essere sviluppate in futuro al fine di migliorare l'esperienza dell'utente e la robustezza del sistema

1. **Tracciamento delle modifiche utente** Attualmente è consentito a ciascun utente online di modificare il documento del testo, senza tenere traccia di chi ha apportato una determinata modifica. Un possibile miglioramento potrebbe essere l'implementazione di sistema che tenga traccia delle modifiche effettuate da ciascun utente e permetta di:

    - Visualizzare chi sta modificando il documento in tempo reale
    - Utilizzare un colore specifico per ogni utente che evidenzi le modifiche che quell'utente ha apportato

2. **Possibilità di scaricare il documento** Un'ulteriore implementazione potrebbe essere la possibilità di scaricare il documento in vari formati (PDF, docx, txt), in modo tale da poterne usufruire anche offline e tenere traccia delle varie versioni del documento

3. Risoluzione dei bug di merge Quando un utente subisce una partizione della rete e poi si riconnette, il testo scritto offline subisce un merge con quello scritto dagli utenti online. Tuttavia, a volte possono verificarsi dei bug. Risulta quindi essenziale:

    - Implementare degli algoritmi di merge più robusti