Real-time collaborative text editor

progetto di sistemi distribuiti - 2021/2022

Maria Mengozzi

Introduzione

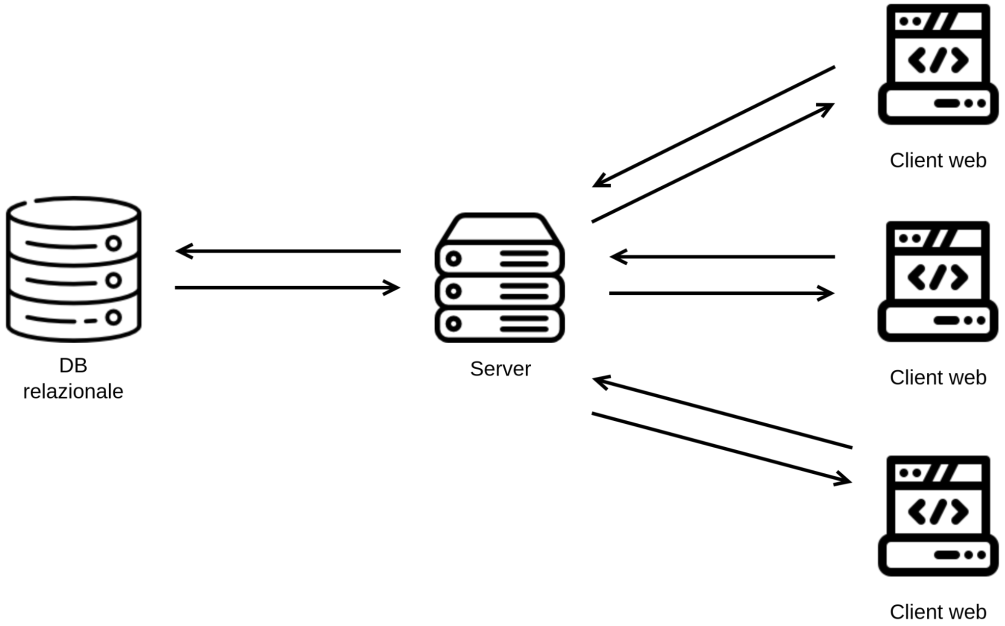
Si è pensato di realizzare un'applicazione che permetta agli utenti di connettersi da remoto e **creare ed editare in modo collaborativo documenti testuali**.

- un utente per poter iniziare ad utilizzare l'applicazione dovrà prima **registrarsi**, dopo di che avrà la possibilità di:
  - **creare** un nuovo documento, specificando titolo e lista di utenti che possono accedervi, scelti tra quelli già registrati all'applicazione
  - **editare** un documento in modo collaborativo, visionando sia la lista degli utenti attualmenti attivi nella sessione di lavoro che le modifiche apportate da ciascuno di essi
  - **compilare un planner** per gli incontri settimanali relativi ad un determinato documento

Analisi e modello del dominio

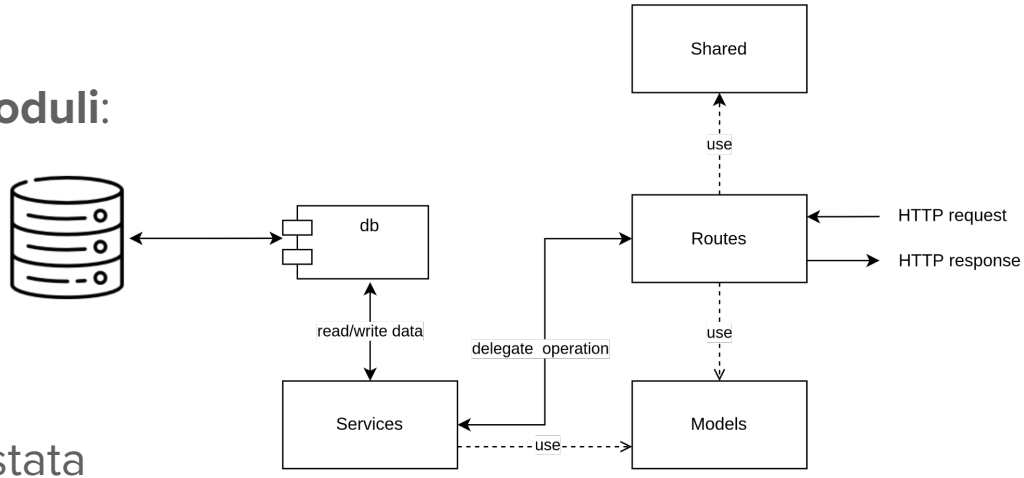
Le entità base del dominio sono:

- database relazionale
- server
- client web



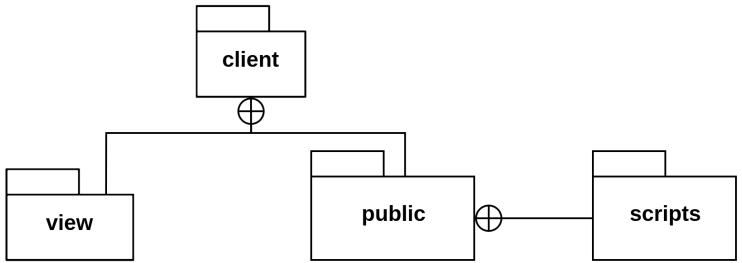
Server

- Il compito principale del server è **gestire le richieste** provenienti di client, **interagendo** con il database
- il server si compone di **quattro moduli**:
  - routes
  - models
  - services
  - shared
- l'interfaccia del servizio offerto è stata formalizzata tramite OpenAPI



Client

- Il client si occupa di gestire l'interazione con l'utente e di compiere le operazioni che quest'ultimo richiede, tramite l'invio di apposite richieste HTTP al server e la gestione delle risposte
- Le operazioni che l'utente effettua determinano il comportamento del client
- Il client è stato realizzato come servizio web, mediante l'uso di HTML, Javascript con il supporto delle librerie JQuery, Quill e Pusher Channels

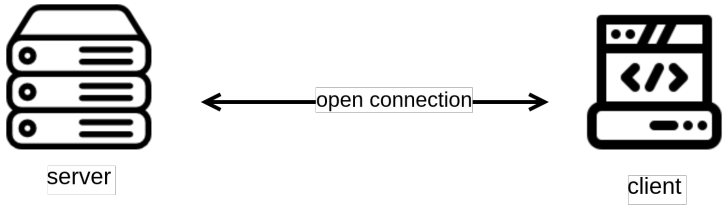


Sincronizzazione

La sincronizzazione tra client avviene per mezzo di **WebSocket**.

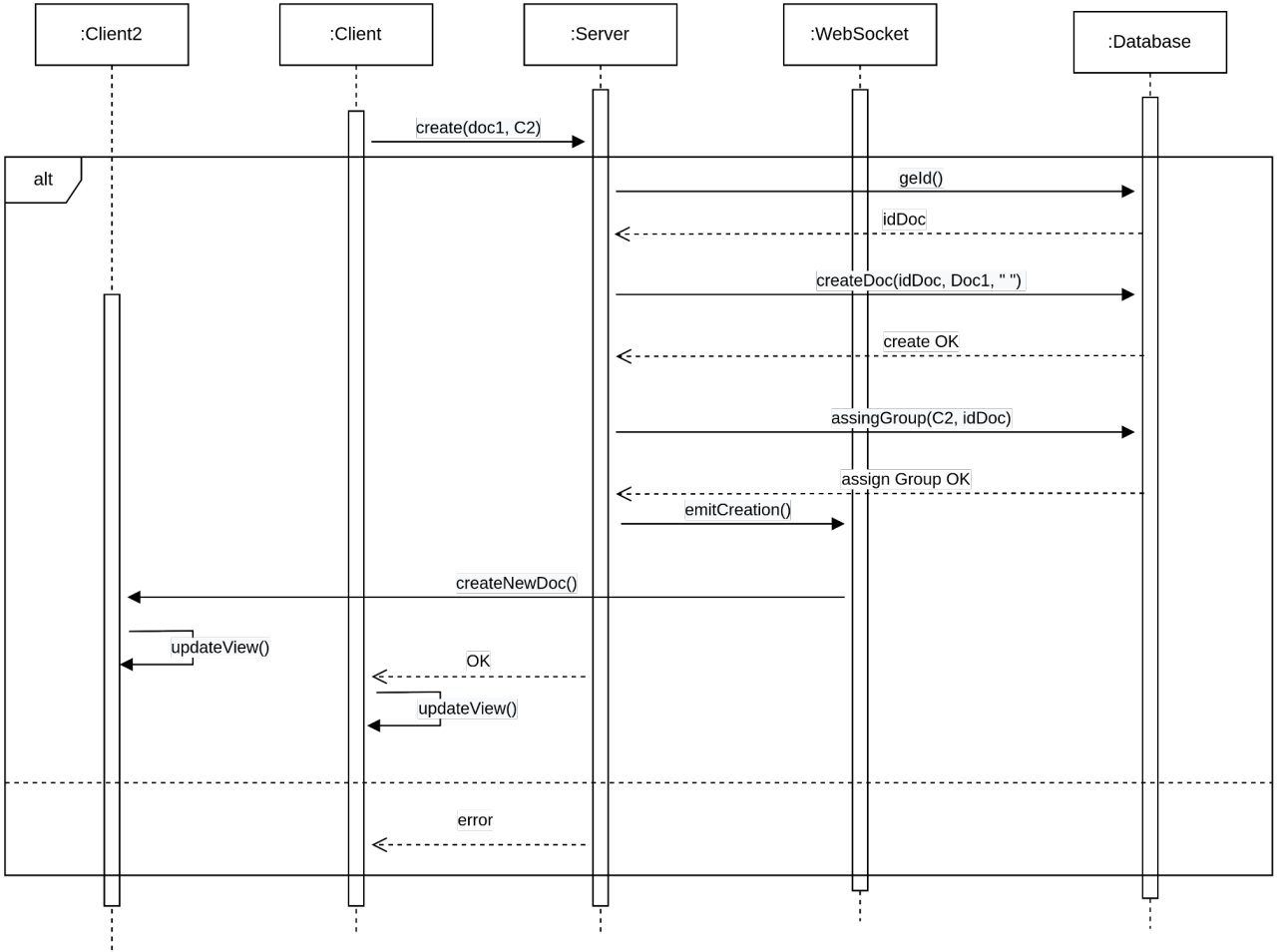
Caratteristiche delle WebSocket:

- bi-direzionale
- full duplex
- TCP based
- comunicazione in tempo reale
- connessione persistente



Nel progetto sono state utilizzate due librerie che mettono a disposizione il servizio di WebSocket: **Socket.io** e **PusherChannel**

Interazioni



Editing del documento

Librerie utilizzate:

- Quill
  - Quill-Delta
- PusherChannel

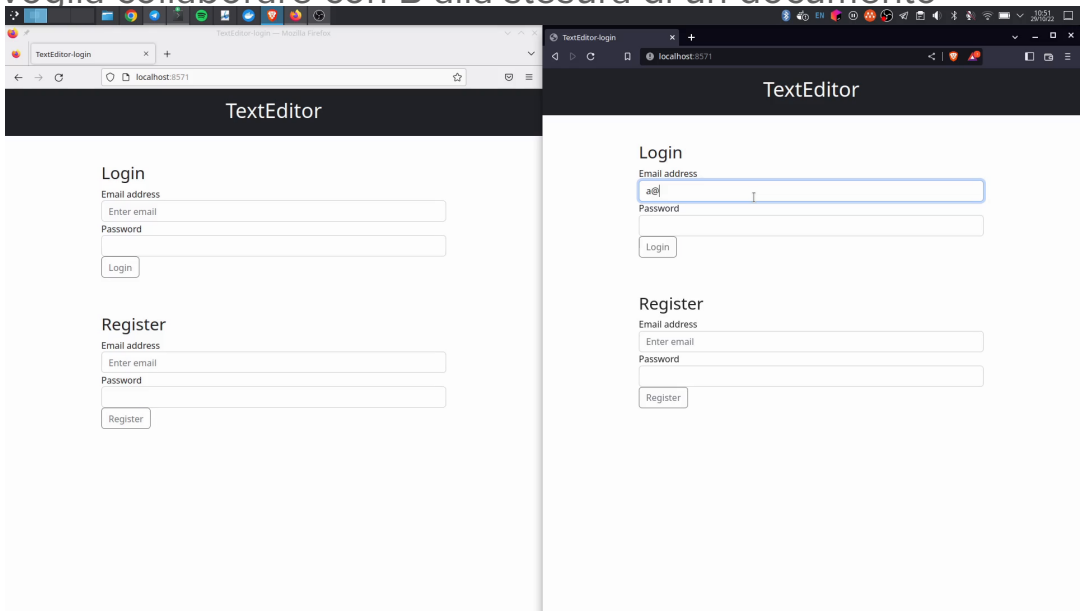
L'aggiornamento del documento avviene a seguito dell'evento **updateContent** e sfrutta 3 operazioni:

- **insert**, inserisce il nuovo contenuto
- **retain**, mantiene inalterato il contenuto
- **delete**, cancella il contenuto

- DEMO -

Esempi di utilizzo - 1

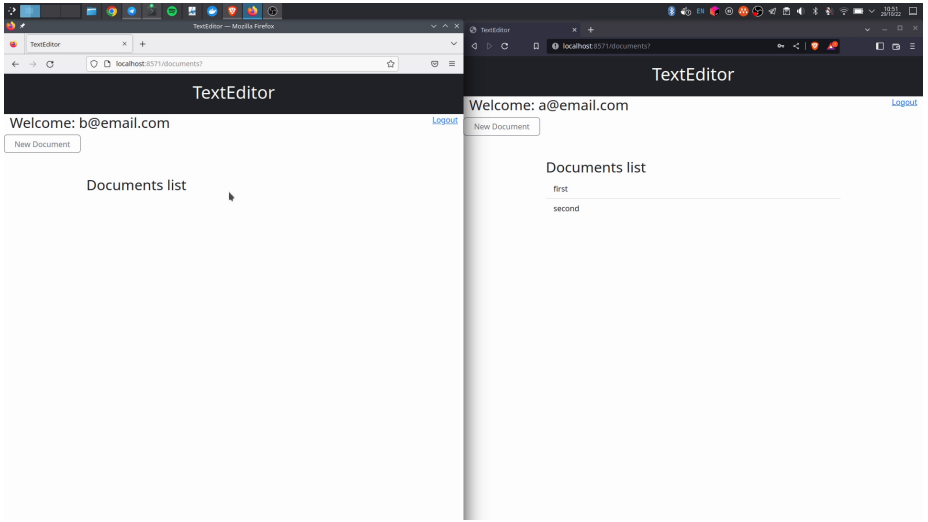
Supponiamo che A voglia collaborare con B alla stesura di un documento



Se sono dei nuovi utenti dovranno effettuare la **registrazione**, altrimenti potranno procedere con il **login**

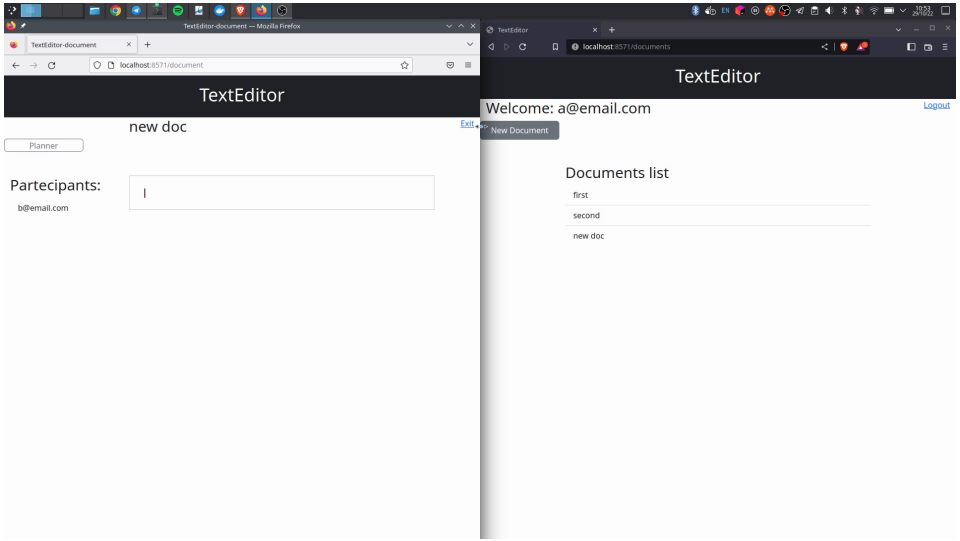
Esempi di utilizzo - 2

Una volta entrati nell'applicazione possono scegliere se **creare** un nuovo documento a cui collaborare, oppure **sceglierne uno** tra quelli già creati



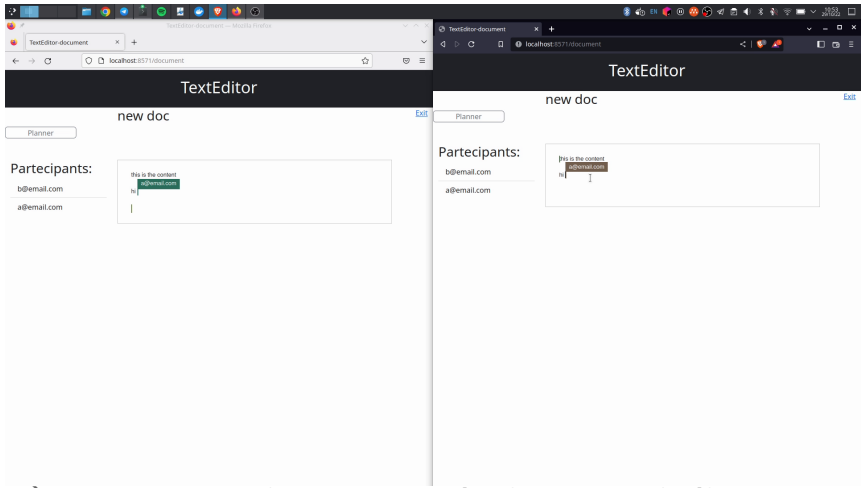
Esempi di utilizzo - 3

Una volta entrati sul documento A e B possono iniziare a **creare e modificare il suo contenuto** visualizzando le modifiche che sta apportando l'altro utente



Esempi di utilizzo - 4

Durante la sessione di editing i due collaboratori possono compilare un planner per segnare i giorni della settimana in cui sono disponibili per un meeting o per un'altra sessione di lavoro condivisa.



L'applicazione mostrerà un suggerimento sul giorno migliore per effettuare il meeting sulla base della disponibilità dei componenti del gruppo di lavoro