Riunione con IMOLA INFORMATICA

RAMtastic6

10 aprile 2024

email: ramtastic6@gmail.com

Informazioni sul documento

Redattori:	Filippo T.
Verificatori:	Riccardo Z.
Verificatori esterni:	

Indice

1 Partecipanti

NOME	DURATA
Alessandro (IMOLA INFORMATICA)	30m
Filippo T.	30m
Leonardo B.	30m
Michele Z.	30m
Riccardo Z.	30m
Samuele V.	30m

2 Riassunto dell'incontro

2.1 Presentazione del *PoC*

2.1.1 Presentazione e prime impressioni

La prima parte dell'incontro ha coinvolto la presentazione, da parte del gruppo, del *Proof of Concept* (PoC) sviluppato dal gruppo per la fase RTB del progetto, e che successivamente sarà soggetto a valutazione da parte del professor Cardin. Durante la presentazione sono state illustrate le seguenti funzionalità implementate: ricerca ristorante, prenotazione e ordinazione collaborativa.
Al termine, il commento di Alessandro di IMOLA INFORMATICA è risultato positivo, evidenziando in particolare l'apprezzamento per la realizzazione dell'ordinazione collaborativa. Difatti quest'ultima è stata una delle funzionalità più complesse, in particolar modo considerando l'implementazione del webserver socket.

2.1.2 Implementazione della *Pagination*

Tornando alla ricerca del ristorante, il proponente ha espresso dubbi riguardo al mancato utilizzo di una *Pagination* per la visualizzazione a schermo dei ristoranti presenti nel database. Il gruppo ha spiegato che al momento la feature non è stata implementata, e il proponente ha quindi proseguito ad illustrarne i vantaggi e due possibili approcci per implementarla: "in memoria" che risulta poco utile nel contesto del progetto da realizzare, e "lato chiamata". Quest'ultimo metodo consiste nel caricare gli elementi necessari al momento del caricamento della pagina. Ad esempio, questa implementazione prevede che se anche la query restituisse 1000 elementi da visualizzare, specificando nel link una "pagina 0" che debba mostrare i primi 20 elementi, allora si otterrebbero solo quest'ultimo numero di ristoranti da visualizzare nella "pagina 0"; specificando invece una "pagina 1", verrebbero visualizzati gli elementi dal 21° al 40° e così via.

2.2 Domande del gruppo

2.2.1 Chiamate al database ad ogni aggiornamento

A seguito del feedback del proponente sul *PoC*, si è passati a una seconda parte dell'incontro con alcune domande di natura tecnica poste dal gruppo all'azienda. Una prima domanda, posta da Leonardo, ha riguardato l'attuale implementazione dei socket. E' stato chiesto se ci fossero alternative all'approccio attuale considerando il potenziale problema legato alle molte chiamate al server che gestisce l'ordinazione collaborativa, dove ogni modifica comporta una chiamata al database.
Alessandro ha risposto che l'attuale sistema implementato è ottimo per la scalabilità orizzontale

argomentando che l'intero sistema può essere considerato *stateless*: con un *token* di autenticazione è possibile ricostruire interamente la situazione di un certo utente. Questo permette di replicare facilmente l'intero sistema, inclusi database, server e frontend. Nella pratica è stato spiegato che anziché utilizzare un unico server per gestire le richieste, sarebbe possibile utilizzarne più di uno. Se un server riuscisse a gestire 100 richieste, 10 di essi potrebbero gestirne 1000, permettendo, appunto, la scalabilità orizzontale.
Alessandro ha continuato aggiungendo un dettaglio importante: l'attuale implementazione del socket non permette la replicazione dello stesso. Pertanto, saranno necessari meccanismi con la stessa semantica *publisher-subscriber*, ma distribuiti. In questo contesto non ci sarà più il socket diretto tra client e server, ma sarà presente un terzo componente che fungerà da connettore. Questo componente pubblicherà su un connettore mentre gli altri riceveranno da quel connettore, che si occuperà di smistare i messaggi a tutti i subscriber ogni volta che viene trasmesso un messaggio. E' stato infine fornito un esempio di questa architettura, ovvero Kafka.

2.2.2 Backend e Socket su progetti diversi

Una seconda domanda è stata posta riguardo all'attuale struttura del progetto: attualmente, socket e backend sono implementati come due progetti *NestJS* separati: si è chiesto se sarebbe più opportuno collegarli.
Dopo un ragionamento iniziale, Alessandro ha esposto al gruppo che quello scelto, invece, è un approccio particolarmente interessante, facendo un parallelo con il paradigma dei microservizi. Ha spiegato che ogni componente di un'applicazione è un servizio a sé, con il suo database, le sue API e il suo server che gestisce l'interfaccia. Questo permette una maggiore scalabilità e manutenibilità del codice: se in futuro si dovesse modificare il modo in cui viene gestita l'ordinazione, sarebbe necessario modificare solo una parte più specifica del codice, riducendo il rischio di impattare negativamente sul resto dell'applicazione. Inoltre, tutti i componenti comunicano attraverso le API REST, che rappresentano un'interfaccia comune e standardizzata facilitando l'integrazione e la manutenzione.
Infine è stato consigliato di parlare e approfondire questa implementazione in sede di esame, in quanto parte non richiesta ma utile per fini di scalabilità del sistema.

2.3 Conclusioni

Sul finire dell'incontro, il proponente ha ribadito l'apprezzamento per aver già integrato nel *PoC* la feature di ordinazione collaborativa tramite il socket, e ha richiesto la possibilità di avere accesso al Repository del *PoC*. Il gruppo ha accettato e condiviso il link del repository GitHub sul gruppo Telegram in comune col proponente.