Huesle

Report iniziale

Alberto Spadoni - 0000995952
(alberto.spadoni3@studio.unibo.it)

Gennaio 2024

1 Abstract

Il presente elaborato ha lo scopo di realizzare una versione digitale e distribuita del gioco da tavolo e di strategia Mastermind.

Tale versione rappresenta una variante rispetto a quella originale e prevede che i due sfidanti abbiano lo stesso ruolo e che debbano competere per la risoluzione di una partita. Per farlo, devono a turni cercare di indovinare un codice composto da una sequenza di 4 colori sulla base dei suggerimenti forniti a partire dai tentativi già effettuati. Tale codice viene generato casualmente dal server di gioco.

La modalità di gioco risulta asincrona in quanto i due giocatori non devono necessariamente essere online e collegati alla partita nello stesso momento per far si che il gioco possa andare avanti. L'interazione tra di essi avviene per mezzo della rete Internet e attraverso i client di gioco, che sono eseguiti sui singoli dispositivi dei giocatori.

L'implementazione in questione consentirà di creare due tipologie di partite: una pubblica in cui l'avversario viene scelto dal server con politica FIFO, e una privata alla quale è possibile accedervi esclusivamente tramite l'inserimento di un codice numerico fornito dal gioco. In seguito alla creazione, una partita potrà iniziare solo quando entrambi i giocatori si saranno uniti ad essa. Da quel momento in poi, il gioco fornirà 10 tentativi totali (5 per utente) entro i quali dovrà essere indovinata la sequenza di colori. Il giocatore che farà la prima mossa viene identificato dal gioco e poi si procederà a turni alterni. Vincerà la partita il primo che indovina la sequenza nelle mosse a sua disposizione. Al contrario, il match terminerà in pareggio se nessuno riesce a svelare il codice.

Oltre alle operazioni di gioco, l'applicazione permetterà agli utenti di registrarsi, autenticarsi, modificare alcune informazioni del profilo e visionare le partite in corso e quelle terminate.

2 Obiettivi e risultato atteso

L'obiettivo principale di questo progetto consiste nel rivedere un sistema già sviluppato e riscriverne completamente il lato backend, ponendo l'accento su tutti quegli aspetti che risultano fondamentali nel design di un sistema distribuito solido. Occorrerà quindi prevedere un'attenta fase di progettazione iniziale in cui si valuta la situazione attuale per capire come adattarla al meglio per garantire scalabilità, disaccoppiamento e robustezza alle disconnessioni.

Il punto di partenza della progettazione è un sistema avente un'architettura client-server formata da tre microservizi:

1. **Client** scritto in React sotto forma di Single Page Application
2. **Server** sviluppato in JavaScript ed eseguito per mezzo di Node.js
3. **Database** non relazionale MongoDB per la memorizzazione degli utenti e dei dati di gioco.

La comunicazione tra i microservizi avviene tramite chiamate HTTP ad una API RESTful e tramite il protocollo WebSocket per alcuni messaggi tra server e client.

L'idea è quella di mantenere l'architettura di partenza e di continuare ad orientarla ai microservizi. Il client rimarrà pressoché invariato, se non per alcuni piccoli adattamenti sul fronte della comunicazione che saranno necessari per assicurarne il corretto funzionamento con il nuovo backend. Il server, al contrario, verrà completamente riscritto in Java e suddiviso a sua volta in due microservizi:

- il primo gestisce gli utenti, le dinamiche di gioco e il database
- il secondo funge da interfaccia tra il client ed il backend Java. In particolare, esso implementerà un server web che riceverà le richieste HTTP dai client, le inoltrerà al backend e provvederà a recapitare le relative risposte.

Il microservizio appena citato verrà implementato con il supporto del framework Vert.x Web che fornisce una serie di strumenti per la realizzazione di un web server reattivo e scalabile. Inoltre, tale framework include una libreria che abilita la comunicazione tramite il protocollo WebSocket interagendo direttamente con l'event-bus di Vert.x.

L'API RESTful esposta dal backend seguirà la struttura di quella del sistema di partenza e la sua specifica formale sarà fornita mediante Swagger.

Per quanto riguarda la comunicazione tra i microservizi che compongono il backend, si pensa di sfruttare RabbitMQ, ovvero un Message Oriented Middleware che consente a diverse componenti eterogenee di comunicare tramite scambio di messaggi.

I microservizi, al loro interno, sfrutteranno gli appositi tool di build automation per effettuare la compilazione ed esecuzione dei sorgenti in modo automatizzato. Più nel dettaglio, il client sfrutterà npm, mentre tutto il backend sarà strutturato come progetto Gradle.

Nell'elaborato in oggetto sarà prevista la scrittura di appositi test, necessari per verificare il corretto funzionamento delle principali componenti. In particolare, il codice del backend verrà verificato mediante JUnit e i test copriranno sia le funzionalità relative agli utenti sia quelle inerenti le operazioni di gioco. Il frontend, invece, verrà testato grazie agli utenti che proveranno il sistema cercando di coprire tutti i flussi di utilizzo possibili.

Infine, verranno fornite le immagini Docker dei microservizi per facilitare e velocizzare il deploy dell'intero sistema.

3 Piano di lavoro

Il piano di lavoro previsto si aprirà con due fasi: studio e progettazione.

La prima attività sarà quella di studio e approfondimento delle tecnologie che si andranno ad utilizzare, quali Vert.x, RabbitMQ e i fondamenti di WebSocket.

In seguito, ci sarà la fase di progettazione e design del backend che ha l'obiettivo di identificare una corretta scomposizione in microservizi, capire come strutturare la gerarchia delle classi e definire un'efficace processo di comunicazione tra il client e il server e tra i microservizi stessi.

Solo a questo punto potrà iniziare il processo di scrittura del codice, che inizierà dal backend e, solo in un secondo momento, passerà al frontend. Più nel dettaglio, si prevede di:

1. identificare tutti i task da completare relativi al backend e ordinarli per priorità,
2. selezionare un sottoinsieme degli stessi e lavorare solo su quelli fino al loro corretto funzionamento,
3. infine, concentrarsi sul client per adattarlo perfettamente ai nuovi microservizi.

Di pari passo con lo sviluppo ci sarà la scrittura e l'esecuzione dei test per tenere sempre sotto controllo la correttezza del codice prodotto.