Peer-Review 2: Comunicazione

Marco Scarpelli, Davide Trapletti, Martina Viganò
Gruppo 55

9 maggio 2022

Valutazione del protocollo di comunicazione del gruppo 65.

1 Lati positivi

1.1 Struttura

A livello di classi che gestiscono la connessione client-server, il sistema pare ben progettato. La struttura è simile alla nostra, anche se noi abbiamo implementato una view a livello anche di server per mantenere separata la generazione di messaggi dallo scambio.

1.2 Protocollo

Il protocollo di comunicazione ci pare sensato, anche se i messaggi si sarebbero potuti accorpare in meno sottoclassi; per esempio, il messaggio per muovere potrebbe avere degli identificatori per source e destination, banalmente un enum, e si potrebbe far risolvere al controller le chiamate in modo da non dover testare ogni possibile sottoclasse di Message.

2 Lati negativi

2.1 Chiamate del controller

Abbiamo notato che il controller prende come parametro una stringa, che abbiamo immaginato essere il nickname del giocatore, per buona parte delle

azioni, ma poi anche il model ha in sè una serie di funzioni che accettano una stringa come parametro. Potrebbe essere meglio far risolvere le reference del giocatore direttamente dal controller senza che sia match a farlo.

2.2 View

Per rendere il modello un po' più chiaro si sarebbe potuta creare una view a livello di server in modo che il Model non faccia la notify direttamente sulla connessione; in tal modo la view sul server si potrebbe occupare della conversione del model in JSON e mandare il messaggio già pronto, viceversa interpretando il JSON in arrivo dal client notificando il controller con un Message.

3 Confronto tra le architetture

Le architetture sono abbastanza simili. Noi non abbiamo implementato il requisito di partite multiple ma per il resto l'idea di avere un server che gestisce un certo numero di entry point legati ai client è la stessa.

Altra piccola differenza è che il nostro client presenta la stessa separazione che c'è sul server, cioè la View (che fa il rendering del modello e interpreta i comandi) ha la parte di rete che è una propria classe, ma è solo una scelta implementativa.

Invece una differenza abbastanza grossa è che noi abbiamo deciso di creare delle classi-scheletro serializzabili per la comunicazione da server a client, mentre qui è stato utilizzato il JSON; in più, i nostri messaggi dal client al server hanno sì un tipo individuato da un enum, ma sono accorpati in un'unica classe che abbiamo cercato di rendere il più completa possibile.