Peer-Review 2: Sequence Diagram

Pietro Agnoli, Arturo Amoretti, Daniel Carozzo, Pietro Benecchi

Gruppo AM26

Valutazione del sequence diagram delle classi del gruppo AM35.

# Introduzione

L’architettura proposta permette di gestire correttamente ogni fase della partita e di gestire contemporaneamente le varie richieste dei client. Il paradigma client server è correttamente rispettato e permette di semplificare l’implementazione. Tuttavia, sembra che in alcune transizioni abusiate di questo paradigma. Inoltre, sono presenti alcune scelte che rallentano le performance della vostra applicazione e vi consigliamo di rivalutarle.

Nella vostra architettura non è stato possibile analizzare la fase di scelta della carta iniziale, essendo completamente assente nel vostro documento. Probabilmente non è stata inserita per errore e proponiamo la nostra implementazione. Infine, suggeriamo alcune modifiche non strettamente richieste nelle regole del gioco o nei requirments, ma che potrebbero migliorare l’esperienza utente e conseguentemente la vostra applicazione.

# Starting Card

Ecco una possibile implementazione di StartingCard: A diagram of a server

Description automatically generated

Quando tutti i giocatori hanno scelto la carta obiettivo, il server invia a tutti i client la carta di partenza inviata come id. Il giocatore sceglie il lato di tale carta e comunica la scelta al server.

# Criticità

L’architettura complessivamente è ben fatta, tuttavia sono sorte varie perplessità su alcune implementazioni. In particolare, riteniamo siano state fatte scelte che complicano la realizzazione della view e un eccessivo utilizzo di richieste al server per cambiare fase di gioco.

# Initialize communications

Nel vostro documento voi scrivete: “Nella nostra applicazione quando un client apre il gioco deve inserire uno username e poi viene inserito da un server direttamente in una partita. In caso non ci siano partite disponibili allora il client deve creare una nuova partita inserendo il numero di giocatori”.

Nel caso di nuova partita, osserviamo che viene inviato nuovamente l’username, anche se è già stato richiesto. È questa richiesta necessaria? Avete già l’informazione del nome del giocatore, perciò potrebbe essere evitata.

# Start game

La fase di start game sembra molto laboriosa. Nella vostra implementazione un utente deve continuamente richiedere l’aggiornamento della partita per sapere se può iniziare a giocare. Questo rispetta chiaramente il paradigma Client-Server ma è molto operoso. Per rispettare il paradigma non è sempre necessario rispondere soltanto allo stesso client. Può succedere infatti che data una richiesta da uno specifico client, il server notifica quel client e anche gli altri client della modifica dello stato di gioco.

Ecco una nostra possibile soluzione: quando l’ultimo client si sarà connesso alla partita in fase di avvio, il server notifica a tutti i client connessi l’inizio della partita. Questa implementazione soddisfa il paradigma ed evita inutili richieste al server.

Si può utilizzare tale proposta sia per isFull che per StartGame, unendo il tutto in un unico messaggio.

Proponiamo la nostra implementazione, così possiate notare la differenza:

A diagram of a server

Description automatically generated

# Place card

La scelta di inviare la table area non è ottimale, in quanto la view deve ristampare continuamente tutta l’area di gioco, rallentando le performance della vostra applicazione.

È consigliabile sempre mandare ai client le informazioni di cosa è cambiato nel model, non continuamente tutto il model. Potreste modificare la vostra implementazione restituendo solo una conferma di posizionamento corretto. È infatti possibile conoscere direttamente lato client dove la nuova carta è stata posizionata e su che lato.

Consigliamo inoltre di introdurre più messaggi di errore/eccezione così da personalizzare maggiormente la view, introducendo le varie casistiche come l’assenza di risorse oppure una posizione scorretta di posizionamento.

# Richieste client non necessarie

In generale la vostra applicazione richiede molte mosse “artificiali” al client, che potrebbero essere evitate modificando i vari messaggi di ritorno del server. Ricordiamo infatti che a una richiesta di un client, il server può notificare tutti gli altri client della modifica dello stato di gioco. Procediamo ad elencare tutti i casi proponendo la nostra implementazione:

1. **Update Market**

Il comando updateMarket è possibile eliminarlo, modificando la risposta di drawCard. Quando un client pesca la carta, il server notifica tutti i client come è cambiato il tavolo mostrando la nuova carta. Si può trasmettere l’id della nuova carta sul tavolo e la posizione.

1. **requestObjectiveCard**

Dopo che l’ultimo player è entrato, il server notifica i client con le carte obiettivo da scegliere. Vengono inviati gli id delle due carte obiettivo.

1. **Start Game**

Quando tutti i player sono collegati con un username corretto e univoco, il server notifica a tutti i client la chiusura della partita e inizio del gioco. Si può notificare con un messaggio specifico con una unica trasmissione comunicando contemporaneamente entrambe le azioni. Controller e model lato server dovranno essere correttamente inizializzati.

1. **End game**

Le richieste di fine del gioco sono evitabili modificando i messaggi di ritorno per il metodo drawCard (caso in cui un utente superi i 20 punti) o placeCard (caso di fine mazzo). Inoltre, da vostra documentazione sembra che un client possa sempre richiedere la terminazione del gioco, anche quando non è consentito. Questa dovrebbe essere limitato a livello di controller Client oppure nel Model. Nel documento non è chiaro dove avvenga questo controllo.

Funzionalità aggiuntive

Elenchiamo ora una serie di funzionalità aggiuntive della vostra applicazione, le quali non sono strettamente necessarie dai requirments ma potrebbero migliorare l’esperienza utente.

Nella fase iniziale potreste fare scegliere ai client il colore della pedina voluta. Inoltre, invece di aggiungere automaticamente il player a una partita in fase di avvio, potreste far scegliere al giocatore se iniziare una nuova partita o aggiungersi a una già creata.

Successivamente, per tutto il gioco potreste inviare ai client i punteggi ottenuti durante il gioco e le risorse possedute.