Protocollo di Comunicazione

Baldelli Andrea, Branchini Alessandro, Benatti Nicolas
Gruppo AM27

17 giugno 2022

1 Struttura dei comandi inviati da Client e Server

Ogni comando inviato sarà un JSON.
IL JSON ha la struttura seguente:

```
{
        "command": "command_name",
        "param1": "param1",
        "param2": "param2"
}
```

Il parametro "command" indica il tipo di comando eseguito e deve essere sempre presente.
Per aiutare la leggibilità il nome dei comandi è molto simile se non uguale ai metodi del Modello chiamati dal Controller.
Dopo command c'è una lista di parametri riferiti al comando.

2 Connessione del Client al Server

Il Client richiede all'utente indirizzo IP e porta del Server e vi si connette.
Se il Server non viene trovato il Client richiede nuovamente IP e porta all'utente.
Una volta connesso, il Server chiede al Client di inserire un nickname. Quest'ultimo lo chiede all'utente e lo invia al Server.

Server → Client

```
{
        "command": "enterNickname",
}
```

Client → Server

```
{
        "command": "login",
        "nickname": "playerNickname"
}
```

Se il nickname è già in uso il Server lo comunica al Client che lo richiede nuovamente all'utente, altrimenti comunica che il login è avvenuto con successo

Server → Client

```
{
        "command": "nicknameAlreadyPresent"
}
```

Server → Client

```
{
        "command": "loginSuccessful"
}
```

3 Creazione/Unione alla partita

Il Client comunica al Server la volontà di unirsi a una partita specificando numero di giocatori e se vuole partecipare a una partita in variante esperta.
Il Server inserisce il Client nella prima partita disponibile avente le caratteristiche richieste. Se non la trova ne crea una nuova con quelle caratteristiche.

Client → Server

```
{
        "command": "joinMatch",
        "numPlayers": "numPlayers",
        "expertMatch": "true/false"
}
```

Il Server a questo punto comunica al Client l'avvenuto inserimento in una partita

Server → Client

```
{
        "command": "joinSuccessful"
}
```

4 Selezione mago e torre

Quando sono arrivati tutti i giocatori a turno ogni giocatore sceglie torre e mago.
Il server manda al primo giocatore le possibili scelte di maghi/torri.

Server → Client

```
{
        "command": "chooseWizardTower",
        "wizards": ["w1","w2", "..."],
        "towers": ["black", "white", "..."]
}
```

Il Client una volta scelti mago e torre manda i dati al Server

Client → Server

```
{
        "command": "addPlayer",
        "nickname": "playerNickname",
        "wizard": "wizardId",
        "tower": "towerColor"
}
```

Quando il controller riceve i dati del giocatore chiama GameManager.addPlayer();

5 Inizio partita

Quando tutti i giocatori hanno finito di scegliere, il server inizializza la partita con il metodo preparation();
Dopo di ciò viene inviato un JSON a tutti i giocatori con lo stato iniziale della partita e un messaggio che dice che la partita è stata inizializzata (la libreria GSON fornisce dei metodi per la conversione in JSON di interi oggetti).

Server → All Clients

```
{
        "command": "moveDone",
        "gameState": { "..." },
        "lastMove":
            {
                "command": "initialization"
            }
}
```

6 Mosse che un giocatore può compiere

- Giocare una carta assistente
- Spostare studenti entrata → sala, entrata → isola
- Muovere madre natura (specificando il numero di mosse)
- Scegliere una tessera nuvola e prendere tutti gli studenti su di essa
- Giocare una carta personaggio (specificando i parametri necessari)

Lista di comandi che svolgono queste mosse:

```
{
        "command": "playerMovePlayAssistant",
        "assistant": "assistantId"
}

{
        "command": "playerMoveMoveStudentFromEntranceToHall",
        "studentColor": "studentColor"
}

{
        "command": "playerMoveMoveStudentFromEntranceToIsland",
        "studentColor": "studentColor",
        "islandId": "islandId"
}
{
        "command": "playerMoveMoveMotherNature",
        "steps": "steps"
}
```

```
{
        "command": "playerMovePickStudentsFromCloud",
        "cloudId": "cloudId"
}

{
        "command": "playerMovePlayCharacter",
        "characterType": "characterType",
        "studentColor": "studentColor",
        "islandId": "islandId",
        "toExchangeFrom": ["studentColor1", "studentColor2", "..."],
        "toExchangeTo": ["studentColor1", "studentColor2", "..."],
        "toExchangeFromNumber": "numberOfExchanges",
        "toExchangeToNumber": "numberOfExchanges"
}
```

7 Risposta del Server a seguito di una mossa

A seguito della ricezione di un comando mossa da parte di un Client, il Server modifica lo stato del gioco e invia lo stato modificato a tutti i Client in formato JSON insieme a un messaggio contenente l'ultima mossa fatta.

Server → All Clients

```
{
        "command": "moveDone",
        "gameState": { "..." },
        "nickname": "playerOfTheMoveNickname",
        "lastMove":
            {
                "command": "...",
                "param1": "...",
                "param2": "..."
            }
}
```

8 Gestione degli imprevisti

8.1 Logout dell'utente

Siccome non gestiamo la resilienza alle disconnessioni, se un utente si disconnette volontariamente la partita deve terminare per tutti in qualunque fase di gioco.

Il Client manda un messaggio di logout al Server e il Server invia a tutti i Client un messaggio di chiusura forzata affinché si disconnettano.

Client → Server

```
{
        "command": "logout"
}
```

Server → All Clients

```
{
        "command": "forceEndMatch"
}
```

8.2 Server down

Il Server, dopo che un Client si è connesso, inizia ad inviargli dei beat ad intervalli regolari.
Se il Client non rileva più l'arrivo di suddetti beat considera il Server disconnesso e si chiude.

Server → Client

```
{
        "command": "beat"
}
```

8.3 Client down

Allo stesso modo, ogni Client, dopo essersi connesso, inizia ad inviare dei beat al Server ad intervalli regolari.
Se il Server non rileva più l'arrivo dei suddetti beat da un Client lo considera disconnesso. A questo punto la partita termina e il Server invia a tutti gli altri Client connessi un messaggio di chiusura forzata affinché si disconnettano.

Client → Server

```
{
        "command": "beat"
}
```

Server → All Clients

```
{
        "command": "forceEndMatch"
}
```

9 Fine della Partita

La partita si è conclusa correttamente con un vincitore.
Il Server invia normalmente il messaggio di moveDone che contiene lo stato del gioco.
I Client leggendo il gameState si accorgono che è nello stato di GAME_OVER. Sempre dal gameState reperiscono il vincitore e comunicano all'utente la fine della partita insieme con il vincitore.
A questo punto i Client si disconnettono.
Il Server si accorge della loro disconnessione ed esegue le operazioni di pulizia della partita.