Sprint 3 Backlog **[26/07/2017 - Cesena]**

Revisione delle cose fatte:

Branch 'feature-controller'

Abbiamo completato la parte di controller, aggiungendo alcune funzionalità. In particolare abbiamo individuato almeno 2 tipi di controller:

- controllerUser:

Questo controlla tutto ciò che è inerente all'utente (login, caricamento dei risultati, …) e richiama direttamente i metodi del model.
Manca da gestire il logout dell'utente.

- controllerMatch:

Gestisce la configurazione della partita vera e propria, e prende la view come input.
Abbiamo aggiunto un metodo per richiedere al server i range disponibili, un altro per farsi dare la lista dei personaggi da scegliere, uno per scegliere il personaggio, che gestirà anche la possibilità che il personaggio non sia disponibile perchè già scelto da un altro giocatore. Questa informazione verrà passata dal server come risultato della richiesta di login.

Per quanto riguarda la scelta del playground non è importante che riceva il file di tutti i playground, ma solo quello scelto
Il metodo MatchResult invece viene chiamato alla fine della partita per andare a salvare il risultato sul server.

Quando il model andrà modificato dalla parte di comunicazione allora viene richiamato il metodo update.

Quando viene scelto un personaggio e la partita inizia ricevo la mia immagine, ma anche quelle di tutti gli altri giocatori, perchè dovrò disegnare anche quelle e me le dovrò andare a salvare da qualche parte. Per questo salvo le risorse e instanzio le relative entità.

Branch 'feature-clientside'

Parte di comunicazione dalla parte del client. Per comunicare abbiamo deciso di usare messaggi JSON ovunque.
Durante questa settimana abbiamo progettato e implementato lo schema ad attori che gestisce questa parte di comunicazione, che è sviluppato in modo tale da avere un attore che riceva messaggi dal server e un altro che invece glieli invii.
Inoltre è presente un attore che permette, tramite Inbox, di comunicare tra la parte ad oggetti e la parte ad attori del model.

branch 'feature-actor'

Abbiamo iniziato lo sviluppo della parte di Server direttamente su un altro progetto (e quindi su un altro repository (https://github.com/manuBottax/DPAC-Server)) per velocizzare lo sviluppo e la creazione di diversi jar. Questo è possibile perchè il Server avrà una struttura completamente ad attori, e i riferimenti ad entità già presenti nel client sono minimi.

Abbiamo progettato la struttura che dovranno avere gli attori, con un attore che riceve i messaggi dai client e li passa alla struttura interna ad attori che elaborano il messaggio. La risposta arriverà poi ad un attore che la invierà al client desiderato.

In questo sprint abbiamo sviluppato la struttura ad attori e lo scambio di messaggi all'interno del server stesso senza implementarne le funzionalità.

In questo modo abbiamo un primo prototipo con cui testare le comunicazioni in breve tempo, e tuttavia già strutturato come la sua versione definitiva.

branch 'feature-p2pCommunication'

Parte di comunicazione p2p tra i vari client verrà sviluppata usando la tecnologia RMI.

Alla fine della configurazione di una partita verrà ricevuta una lista di ip dei giocatori per quella partita dal server e verrà creata per ogni peer un 'server' e n-1 'client' (1 per ogni ip) per comunicare con gli altri peer della rete.

Inoltre sarà automatizzata anche la parte di configurazione di rmi e di gestione delle porte per impostare l'rmiregistry.

Ogni peer quindi avrà un thread che fa da server ed un thread per ogni giocatore che è collegato alla mia partita. Questo perchè ogni client dovrà ascoltare l'rmiregisty (il server) di uno dei giocatori. Infatti il suo server scriverà i valori del suo model su questo oggetto remoto, in modo che quando avvengono le modifiche il controller viene notificato.
Quando faccio una modifica faccio un rebind (una modifica) al valore che ho salvato nell'rmiregistry. La logica è la stessa del pattern observer.

Per ora abbiamo completato la configurazione dei peer, ora resta da finire la parte di comunicazione tra di essi e lo scambio di messaggi.