Nel nostro gioco abbiamo sviluppato:

- RMI
- Socket
- GUI
- CLI
- 2 funzionalità avanzate : ripristino sessione giocatore, lobby e in parte la gestione degli utenti (una volta effettuato l'accesso la prima volta i giocatori rimangono salvati sul server con la propria password e possono riaccedervi, in questo modo nessun'altro potrà usare il loro nome anche se sono offline)

Abbiamo suddiviso la struttura del gioco inizialmente in 2 parti, client e server.
Il server si divide essenzialmente in questi componenti:

- Model che salva sia giocatori collegati e il loro stato (online, in game ecc..) sia l'elenco di partite presenti con relativo stato.
- Logica di gioco (GameManager) che modifica il model di un gioco in base alle scelte dei giocatori.
- Controller (RMIControllerServer) che si occupa di soddisfare le richieste dei client (utilizzando la logica di gioco per richieste relative ad azioni di gioco).
- Connessione verso il client (Connection implementata da 2 classi) per permettere al server di contattare il client (attraverso metodi per RMI o mandando messaggi con un protocollo per Socket).
- Connessione ricevente (SocketServer e RMIControllerServer, quest'ultimo ha 2 ruoli) che permette al server di essere contattato dal client (attraverso metodi per RMI o mandando messaggi con un protocollo per Socket).
- Prepara dati da inviare al client (PrepareDataForClient) è una classe che legge lo stato del gioco dal model e lo passa al client utilizzando tipi di Java elementari (principalmente String, int). In questo modo passiamo al client solo i dati necessari ossia quelli che sono cambiati e non tutto lo stato di gioco. Inoltre il client non avrà bisogno di conoscere tutto il complesso model del server ma soltanto sapere come sono organizzati i dati nelle costanti o vettori che gli vengono passati dal server.

Mentre il client si divide in:

- Model che salva:
  1. In locale i dati della partita necessari al singolo giocatore per giocare.
  2. In quale schermata si trova il giocatore.
  3. Alcune infomazioni riguardanti il giocatore (nome e tipo di connessione scelta)
- View divisa in CLI e GUI che interagisce con l'utente (visualizzando informazioni e leggendo i comandi inseriti).
- Controller che in base alla schermata in cui si trova il giocatore e all'opzione scelta eseguira certe funzioni.
- Connessione verso il server (ClientConnection implementata da 2 classi) che permette al client di contattare il server (attraverso l'invocazione di metodi RMI o mandando messaggi con lo stesso protocollo usato dal server per Socket).
- Connessione ricevente (SocketClient e RMIClient) che permette al client di essere contattato dal server (attraverso l'invocazione di metodi RMI o mandando messaggi con lo stesso protocollo usato dal server per Socket).

Per rendere la View indipendete dal controller abbiamo utilizzato il pattern Observe-Observable.
Per rendere invece il Controller indipendete dal tipo di Connessione (sia nel client che nel server) abbiamo creato un interfaccia (Connection e ClientConnection) implementata poi in due differenti modi per RMI e Socket.