

Re di Quadri

Pietro Lelli

`pietro.elli2@studio.unibo.it`

February 2023

Il progetto prevede la progettazione e l'implementazione del seguente gioco, che prende il nome di Re di Quadri, tratto dalla serie tv "Alice in Borderland", in versione online.

Ogni giocatore inizia la partita con 5 vite, ad ogni round deve scegliere un numero da 1 a 100, viene calcolata la media dei numeri inseriti da tutti i giocatori e moltiplicata per 0.8, chi si avvicina di più al numero calcolato vince e gli altri perdono una vita.

Quando un giocatore ha finito le sue vite ha perso e viene eliminato. Ad ogni eliminazione viene aggiunta una nuova regola e l'ultimo giocatore che rimane in vita ha vinto.

1 Goal/Requirements

Di seguito vengono riportate una serie di domande e risposte per spiegare più nel dettaglio gli aspetti del progetto relativi soprattutto alle iterazioni dell'utente con il sistema.

1.1 Questions/Answers

- *Come si accede al sistema?*
 - Per accedere al sistema è sufficiente inserire un nickname (che non risulti già in uso) che identifica univocamente il giocatore per l'intera sessione di gioco.
- *Una volta registrati cosa è possibile fare?*
 - Una volta effettuato l'accesso con il proprio nickname, nel caso in cui non esistano lobby disponibili se ne può creare una nuova, altrimenti è possibile scegliere se entrare in una delle lobby disponibili oppure crearne comunque una nuova.
- *Quanti giocatori devono essere presenti nella lobby per poter iniziare la partita?*
 - La partita potrà iniziare quando ad una lobby si sono uniti da 3 a 5 giocatori. Quando si sono connessi alla lobby 3 o 4 giocatori il creatore della lobby potrà scegliere se far iniziare la partita oppure aspettare che tutti e 5 i giocatori si siano connessi.
- *Com'è organizzata una partita?*
 - All'inizio di ogni partita, in base al numero dei giocatori, verranno visualizzate le regole di gioco. Successivamente ogni giocatore inserirà un numero e a questo punto il sistema calcolerà il vincitore del round, tutti gli altri giocatori perderanno una vita. Quando un giocatore termina le vite viene eliminato dalla lobby e viene aggiunta una nuova regola.
- *Come cambiano le regole durante lo svolgimento del gioco?*
 - In base al numero di giocatori le regole sono le seguenti:
 - 5 giocatori: Il sistema calcola la media dei numeri inseriti da tutti i giocatori e la moltiplica per 0.8, chi si avvicina di più al numero calcolato vince e gli altri perdono una vita.
 - 4 giocatori: oltre alla regola precedente si aggiunge che se due giocatori scelgono lo stesso numero viene detratta una vita a questi giocatori.
 - 3 giocatori: oltre alle precedenti si aggiunge che scegliere il numero esatto fa perdere 2 vite agli altri giocatori.
 - 2 giocatori: oltre alla regola iniziale dei 5 giocatori si aggiunge che se un giocatore sceglie 0 l'avversario vince se sceglie 100, se invece entrambi scelgono lo stesso numero nessuno dei due perde una vita e si passa al round successivo.

- *Cosa succede quando termina la partita di un giocatore?*
 - La partita di un giocatore termina quando ha terminato tutte le vite a disposizione. A quel punto verrà rimosso dalla lobby e potrà decidere se continuare a giocare, entrando in un'altra lobby, oppure uscire dal gioco.

1.2 Scenarios

All'avvio dell'applicazione viene richiesto all'utente di inserire un nickname per poterlo identificare durante la partita.

Dopo essersi connesso verranno mostrate all'utente le lobby disponibili e potrà scegliere se entrare in una di queste oppure crearne una nuova, se opta per quest'ultima scelta dovrà inserire il nome che identificherà la lobby. A questo punto rimane in attesa dell'arrivo di altri partecipanti.

Quando si connettono alla lobby 3 o 4 giocatori il creatore della lobby può scegliere se far iniziare la partita oppure aspettare l'arrivo di altri giocatori, se decide di aspettare, quando 5 giocatori si sono connessi la partita inizierà in automatico.

Una volta iniziata la partita l'utente deve scegliere un numero da 1 a 100, il sistema calcola il vincitore del round e tutti gli altri giocatori perdono una vita.

Quando un giocatore ha terminato le vite viene rimosso dalla lobby e può decidere se uscire dal gioco oppure continuare a giocare, in questo caso dovrà scegliere una nuova lobby.

1.3 Self-assessment policy

Il progetto sarà dotato di una serie di test automatici, realizzati tramite il framework JUnit.

I test sono stati realizzati per controllare più aspetti del sistema, come ad esempio i test per controllare la consistenza del sistema (test che controllano l'inserimento di due utenti con lo stesso nickname o due lobby con lo stesso ID), ma anche test che simulano una vera e propria partita, controllando ad esempio la logica e il calcolo del vincitore di un round oppure che testano quando un giocatore ha terminato le vite.

2 Requirements Analysis

Di seguito vengono riportati dei diagrammi dei casi d'uso per individuare le funzionalità che il sistema dovrà possedere, mostrando il comportamento del sistema per quanto riguarda la connessione dell'utente al sistema e lo svolgimento della partita.

2.1 Connectivity functionality

- Registra nickname
- Creazione di una lobby
- Connessione ad una lobby
- Fa iniziare una partita

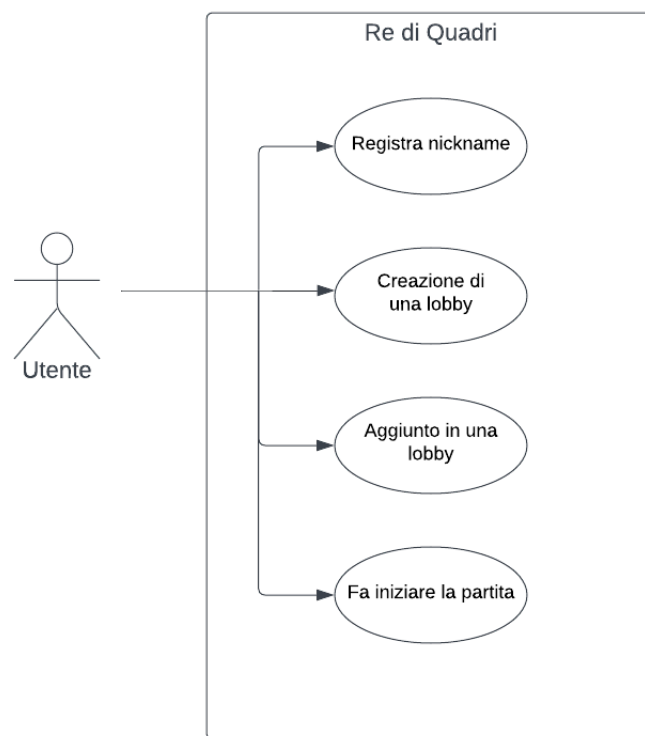


Figure 1: Initial logical Use Case

All'avvio dell'applicazione l'utente dovrà inserire il proprio nickname, se questo non è già presente l'utente sarà loggato. A questo punto gli saranno mostrate le lobby disponibili e potrà decidere se entrare in una di queste oppure crearne una nuova.

2.2 Game functionality

- Inserisce numero
- Effettuare una nuova partita
- Terminare gioco

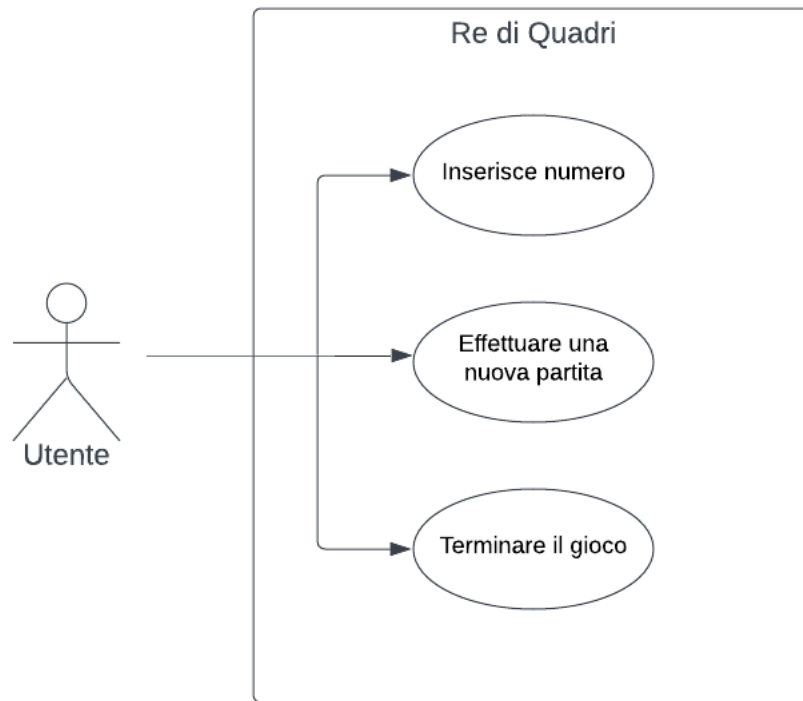


Figure 2: Game logical Use Case

L'utente, una volta iniziato ogni round, dovrà scegliere un numero tra 1 e 100 entro un tempo limite, successivamente rimane in attesa che tutti i giocatori abbiano scelto il proprio numero.

A questo punto il sistema calcolerà il vincitore e rimuoverà una vita a tutti gli altri. Quando un giocatore ha terminato le vite sarà rimosso dalla lobby e potrà decidere se continuare a giocare, scegliendo una nuova lobby di gioco oppure terminare e chiudere l'applicazione.

A seguito dell'analisi effettuata è stato deciso di implementare il sistema utilizzando un'architettura client-server, implementando il server come un Web Service, dotato di ReST API e che utilizza quindi HTTP (POST, GET, PUT, DELETE) come protocollo di comunicazione.

3 Design

Successivamente all'analisi dei requisiti si procede illustrando il design che si è scelto di adottare per la realizzazione del sistema del gioco Re di Quadri. La struttura del design scelto vuole mantenere un livello di astrazione del sistema elevato e vuole realizzare un'architettura che possa essere il più possibile indipendente dall'implementazione realizzata.

3.1 Structure

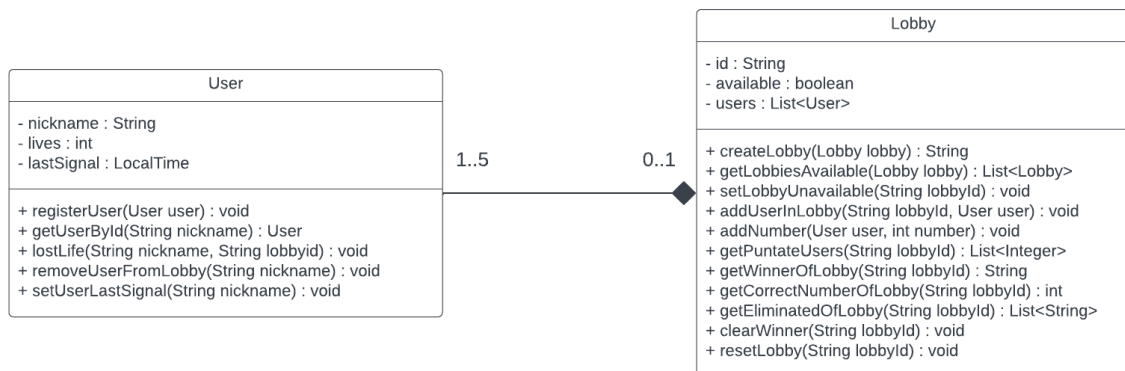


Figure 3: Class Diagram

In figura 3 è mostrata la struttura principale dei componenti del sistema, realizzata tramite diagramma delle classi UML.

Il sistema si compone di due entità: User e Lobby.

L'entità User rappresenta il concetto di utente, esso viene identificato da un nickname univoco, un attributo lives che rappresenta le vite a disposizione e dall'attributo lastSignal, che verrà settato ripetutamente durante l'esecuzione e che servirà ad individuare la disconnessione dell'utente stesso.

Inoltre contiene tutti i metodi necessari per il funzionamento del gioco, come quello per registrare l'utente (`registerUser`), quello per fargli perdere una vita (`lostLife`) e quello per terminare la sua partita quando ha finito le proprie vite (`removeUserFromLobby`).

L'entità Lobby invece rappresenta un gruppo di utenti che giocano insieme una partita. Questa è sempre composta da almeno un utente (colui che la crea) ed è considerata non disponibile quando è piena (5 giocatori) o quando la partita è cominciata. La lobby è identificata da un id, un booleano che indica se la lobby è disponibile e da una lista di utenti che vi fanno parte.

Questa entità contiene al suo interno una serie di metodi necessari per la gestione delle partite:

- createLobby(Lobby lobby): utilizzato per la creazione della lobby.
- getLobbiesAvailable(Lobby lobby): utilizzato per mostrare all'utente solo le lobby disponibili.
- setLobbyUnavailable(String lobbyId): utilizzato per settare una lobby a "Non disponibile".
- addNumber(User user, int number): utilizzato per aggiungere il numero scelto dall'utente.
- getWinnerOfLobby(String lobbyId): utilizzato per mostrare il vincitore del round a tutti gli utenti.
- getEliminatedOfLobby(String lobbyId) : utilizzato per mostrare i giocatori che hanno terminato le vite in quel round
- resetLobby(String lobbyId): utilizzato per resettare tutti i dati della lobby.

3.2 Behaviour

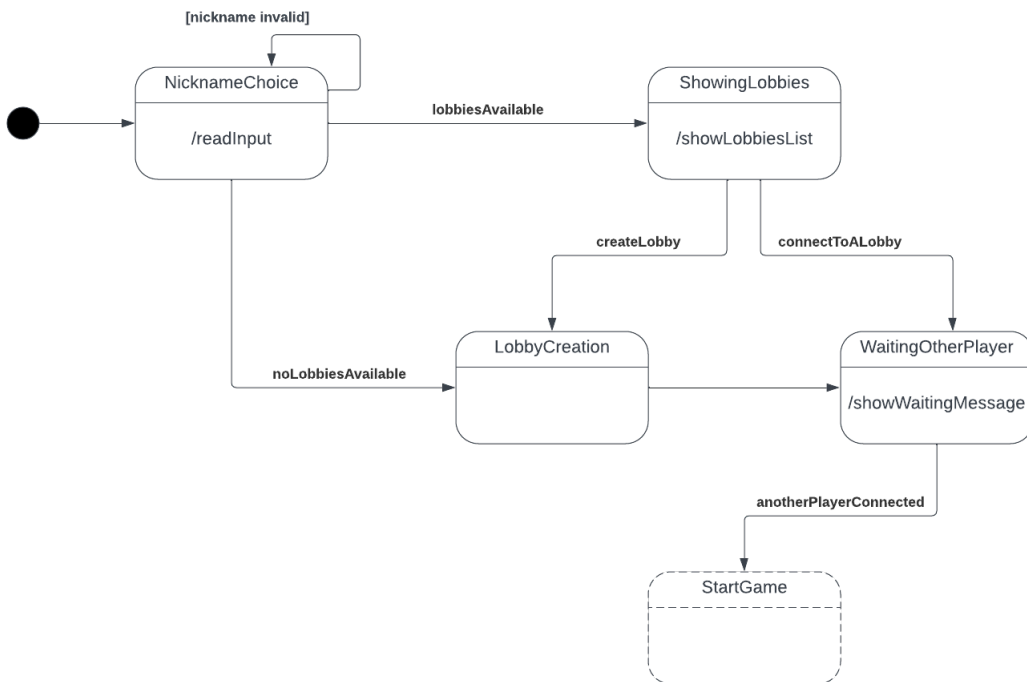


Figure 4: Lobby State Diagram

La figura 4 mostra il diagramma degli stati che rappresenta la fase iniziale del gioco, che comprende la registrazione del nickname dell'utente e la fase di accesso o creazione delle lobby.

Inizialmente viene richiesto all'utente di inserire un nickname valido, poi come descritto nei requisiti, se non sono presenti lobby disponibili l'utente ne creerà una nuova, mentre se ci sono lobby disponibili, queste saranno mostrate e l'utente potrà decidere se accedervi o crearne una nuova.

A questo punto l'utente rimarrà in attesa che altri giocatori si uniscano alla lobby.

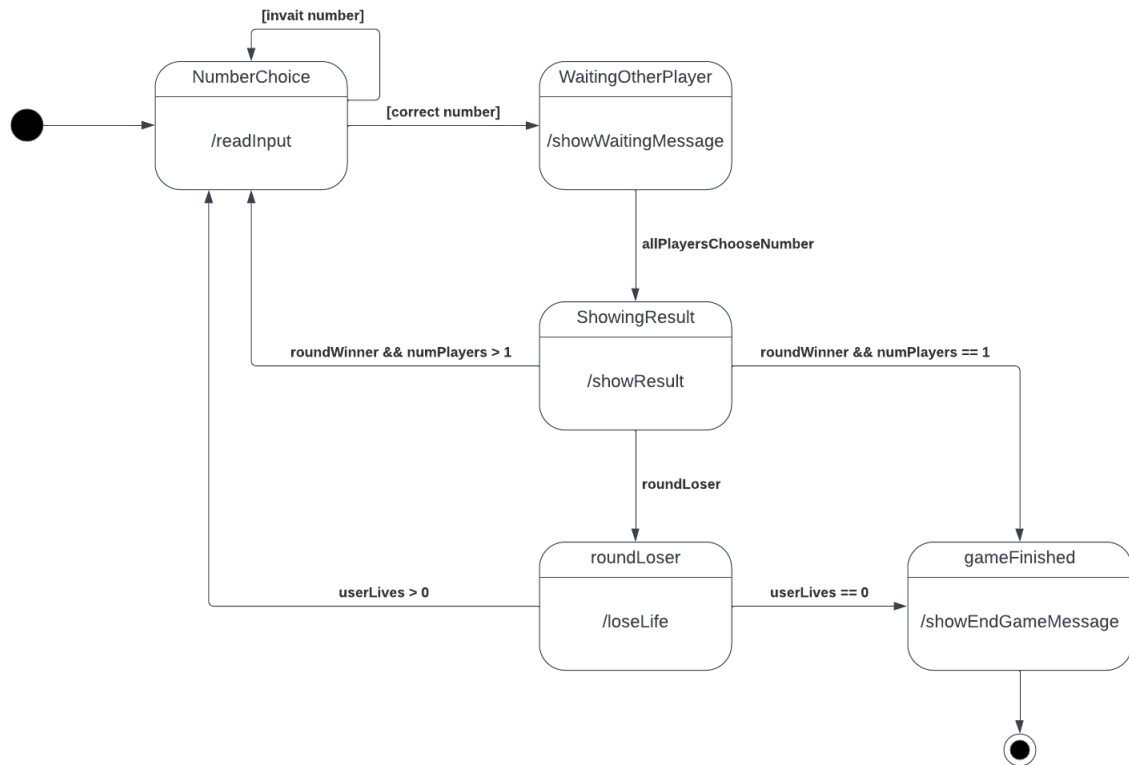


Figure 5: Game State Diagram

In figura 5 viene mostrato invece il diagramma degli stati che rappresenta lo svolgimento di una partita.

Inizialmente sarà richiesto all'utente di inserire un numero valido (compreso tra 0 e 100), poi rimarrà in attesa che tutti gli altri giocatori della lobby abbiano scelto il proprio numero. Verranno poi mostrati i risultati del round, quindi i numeri inseriti dagli utenti e il vincitore. A questo punto si possono verificare diversi scenari: se l'utente è il vincitore del round e tutti gli altri hanno terminato le vite tale utente è il vincitore della partita e il gioco termina, se l'utente è il vincitore del round e altri giocatori sono in vita la partita prosegue con il round successivo.

Se invece l'utente ha inserito il numero sbagliato perde una vita: nel caso in cui abbia ancora vite a disposizione la partita prosegue con il round successivo, se le sue vite sono terminate la sua partita termina.

3.3 Interaction

A seguire verrà analizzata la struttura delle iterazioni all'interno del sistema.

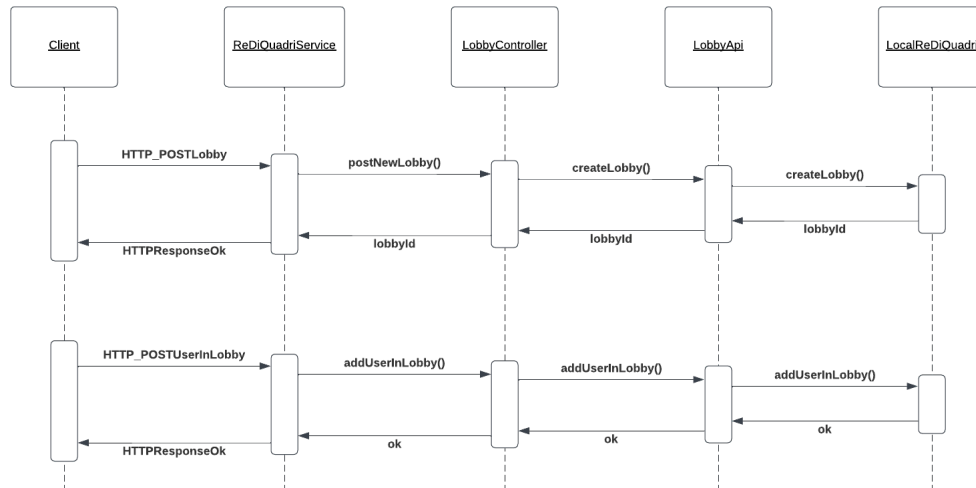


Figure 6: Lobby Sequence Diagram

In figura 6 viene mostrato il diagramma di sequenza che rappresenta la creazione di una lobby e il successivo inserimento di un utente al suo interno.

La lobby, come anche le altre risorse, possiede lato server un Controller e un insieme di Api che risponderanno alle richieste del Client. In questo caso, quindi, quando il Client effettua la richiesta POST al server per la creazione della lobby, essa viene indirizzata al LobbyController, che richiamerà l'API contenuta in LobbyApi che infine agirà sull'istanza locale di ReDiQuadri presente sul server.

Questa modalità di interazione vale per tutte le altre richieste, come nel caso della richiesta di aggiunta di un User all'interno della lobby che è mostrata sempre in figura 6.

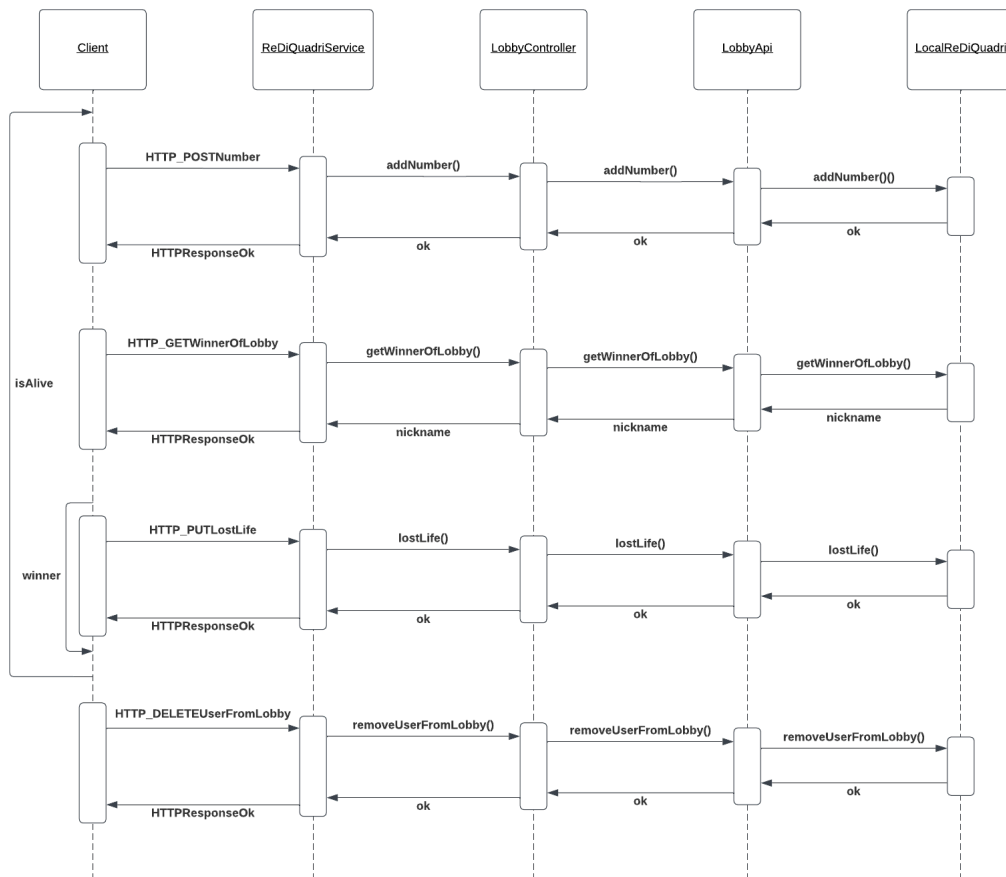


Figure 7: Game Sequence Diagram

In figura 7 viene invece mostrato il diagramma di sequenza che rappresenta lo svolgimento di una partita di una lobby.

In questo caso ogni utente, tramite richiesta POST al server, aggiungerà il proprio numero scelto; successivamente il sistema calcolerà il vincitore e ogni Client tramite la chiamata HTTP GETWinnerOfLobby otterrà il vincitore del round. A questo punto gli utenti che non hanno vinto effettueranno una chiamata HTTP PUT che avrà il compito di rimuovere loro una vita.

Se l'utente ha ancora vite a disposizione continua a giocare il round successivo, se invece ha terminato le vite viene rimosso dalla corrente lobby di gioco con una chiamata HTTP DELETE (removeUserFromLobby).

4 Implementation Details

In questa sezione verranno analizzate le strategie di implementazione utilizzate e descritte brevemente le tecnologie utilizzate.

Il progetto è stato interamente sviluppato in Java ed è composto principalmente in due parti: Client e ReDiQuadriService.

4.1 Client

Per quanto riguarda il Client, esso contiene tutti i metodi che consentono di effettuare le richieste HTTP al server attraverso i seguenti URL:

- */users*: fa riferimento a tutte le richieste che consentono la gestione degli utenti, quindi ad esempio la creazione di un utente o il decremento di una vita da quelle disponibili.
- */lobby*: fa riferimento a tutte le richieste che invece consentono la gestione delle lobby e del funzionamento delle partite, quindi ad esempio la creazione di una lobby, la richiesta GET del vincitore del round, ma anche la rimozione dei dati di una lobby per il round successivo.

In aggiunta, in alcuni casi viene aggiunta all'URL qualche informazione necessaria al funzionamento, sotto forma di *pathParam*; ad esempio passando l'id della lobby o il nickname dell'utente, per poter eseguire operazioni su una specifica Lobby o User.

All'interno del Client è inoltre presente un main che, in base alle interazioni dell'utente con l'interfaccia a linea di comando, consente di inviare le richieste HTTP al server.

4.2 ReDiQuadriService

Il server è stato sviluppato utilizzando il framework Javalin [2] ed è strutturato come web service, contiene infatti varie risorse dotate di un Controller e di un insieme di Api che, come già detto consentono di far interagire il Client con il Server.

In particolare per ogni risorsa viene registrato il relativo Controller all'indirizzo corrispondente, così facendo è possibile ricevere le richieste del Client e reindirizzarle al componente corretto.

Le API invece interagiscono con un'istanza locale del sistema.

lobbies	
POST	/rediquadri/v0.1.0/lobbies
GET	/rediquadri/v0.1.0/lobbies/
DELETE	/rediquadri/v0.1.0/lobbies/
PUT	/rediquadri/v0.1.0/lobbies//setUnavailable/{lobbyId}
GET	/rediquadri/v0.1.0/lobbies/{lobbyId}
PUT	/rediquadri/v0.1.0/lobbies/{lobbyId}
POST	/rediquadri/v0.1.0/lobbies/addUser/{lobbyId}
POST	/rediquadri/v0.1.0/lobbies/game/{number}
GET	/rediquadri/v0.1.0/lobbies/puntate/{lobbyId}
GET	/rediquadri/v0.1.0/lobbies/game/{lobbyId}
GET	/rediquadri/v0.1.0/lobbies/correctNumber/{lobbyId}
GET	/rediquadri/v0.1.0/lobbies/loser/{lobbyId}
DELETE	/rediquadri/v0.1.0/lobbies/reset/all

Figure 8: Swagger Lobby Doc

users	
POST	/rediquadri/v0.1.0/users
GET	/rediquadri/v0.1.0/users/{userId}
PUT	/rediquadri/v0.1.0/users/{lobbyId}
DELETE	/rediquadri/v0.1.0/users/{nickname}
PUT	/rediquadri/v0.1.0/users/lastSignal/{nickname}

Figure 9: Swagger User Doc

Tutte le rotte registrate nel Web Server sono state documentate tramite l'utilizzo di Swagger [4] e sono consultabili seguendo il percorso /doc/ui. Per quanto riguarda il livello di presentation, invece, è stata utilizzata la libreria Gson [1] per la serializzazione e deserializzazione degli oggetti.

4.3 Ensure reliability

Per evitare il blocco del gioco nel caso in cui un utente non inserisca il proprio numero è stato inserito un TimerTask che nel caso in cui un utente per più di 30 secondi non inserisca il proprio numero, questo viene rimosso dalla lobby, viene mostrato un messaggio agli altri giocatori della lobby e il gioco continua con il round successivo.

Un'altra precauzione, per evitare uno stallo nel caso in cui uno dei Client si disconnetta o crashi durante il gioco, consiste in un campo (lastSignal) presente in ogni utente che viene settata al TimeNow ad ogni interazione dell'utente col sistema.

Quindi il Server periodicamente controllerà che il tempo passato dall'ultima interazione non sia maggiore di 30 secondi, in questo caso l'utente viene rimosso dalla lobby e il gioco può continuare con il round successivo.

Grazie a questa funzionalità del Server anche in fase di creazione e formazione delle lobby se un giocatore è entrato in una lobby e il suo Client si disconnette o crasha, dopo 30 secondi viene eliminato dalla lobby.

Quindi se dopo 30 secondi la lobby è ancora in fase di attesa ai nuovi giocatori verrà fatto vedere il numero corretto di utenti, mentre se la partita è iniziata nell'arco di quei 30 secondi al primo round il giocatore disconnesso verrà eliminato.

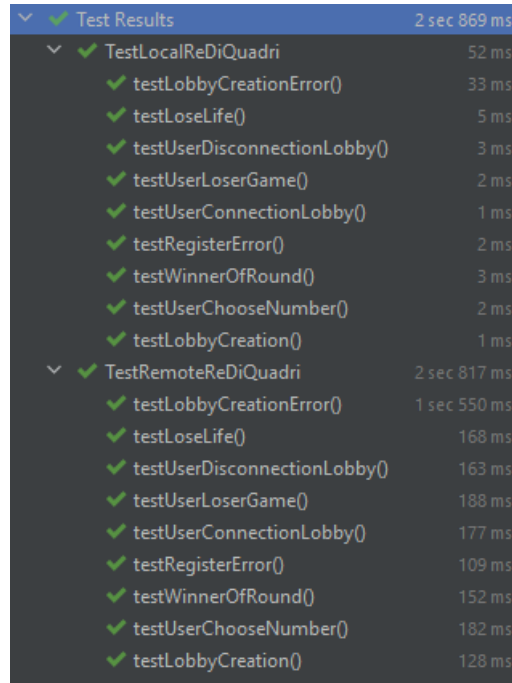
Oltre a questo il sistema lancia una serie di eccezioni con relativi messaggi output all'utente.

Di seguito gli errori rilevati:

- Nickname già presente.
- Id lobby già presente.
- Lobby inserita non esiste.
- Numero inserito non compreso tra 0 e 100.
- Lobby inserita non esiste.
- Input vuoto.

5 Self-assessment / Validation

Il testing è stato realizzato tramite test automatici utilizzando il framework JUnit [3], presenti all'interno del progetto nel modulo chiamato test.



✓ Test Results	2 sec 869 ms
✓ TestLocalReDiQuadri	52 ms
✓ testLobbyCreationError()	33 ms
✓ testLoseLife()	5 ms
✓ testUserDisconnectionLobby()	3 ms
✓ testUserLoserGame()	2 ms
✓ testUserConnectionLobby()	1 ms
✓ testRegisterError()	2 ms
✓ testWinnerOfRound()	3 ms
✓ testUserChooseNumber()	2 ms
✓ testLobbyCreation()	1 ms
✓ TestRemoteReDiQuadri	2 sec 817 ms
✓ testLobbyCreationError()	1 sec 550 ms
✓ testLoseLife()	168 ms
✓ testUserDisconnectionLobby()	163 ms
✓ testUserLoserGame()	188 ms
✓ testUserConnectionLobby()	177 ms
✓ testRegisterError()	109 ms
✓ testWinnerOfRound()	152 ms
✓ testUserChooseNumber()	182 ms
✓ testLobbyCreation()	128 ms

Figure 10: JUnit Test

La classe `AbstractTestReDiQuadri` contiene tutti i metodi di test per verificare il corretto funzionamento di tutte le funzionalità del sistema. Sono presenti infatti test che vanno dall'iniziale registrazione di un nuovo utente o nuova lobby, a casi più complessi che verificano l'effettivo svolgimento di un round di gioco, quindi la scelta dei numeri da parte degli utenti e il calcolo del vincitore.

Inoltre sono presenti test che controllano la gestione dei casi di errore in cui il sistema lancia un'eccezione.

6 Deployment Instruction

Essendo il progetto realizzato in Java, è necessario eseguirlo su una macchina dotata di JVM.

Per prima cosa è necessario avviare il server, posizionandosi nella directory del progetto con un terminale ed eseguendo il comando: `./gradlew --console plain rq-server:run`

A seguire, per simulare la partecipazione dei giocatori è necessario eseguire il comando su un nuovo terminale: `./gradlew --console plain rq-client:run`

7 Usage Example

Una volta avviato il Client viene mostrato in output un'immagine e successivamente viene chiesto all'utente di inserire un nickname.

Nel caso in cui l'utente inserisca un nickname già presente viene visualizzato un messaggio di errore e viene chiesto all'utente di inserirne un altro.

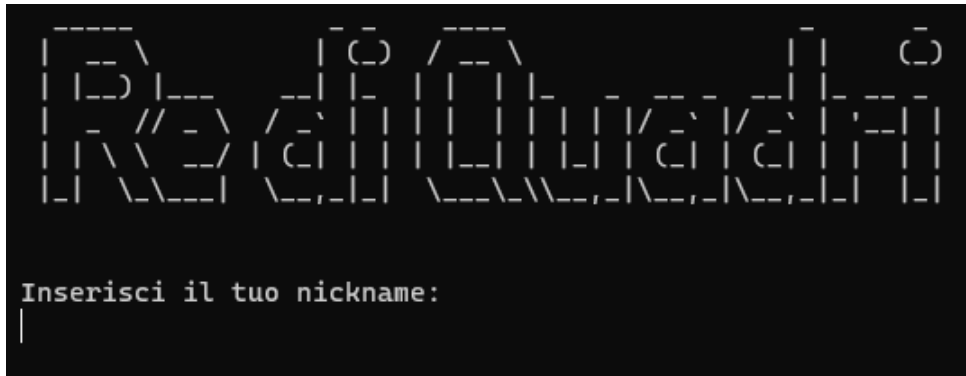


Figure 11: Choose nickname

Dopo aver effettuato l'accesso vengono mostrate le lobby disponibili, è possibile accedere digitando il nome della lobby oppure scegliere di crearne una nuova digitando prima "c" e successivamente inserendo il nome per la propria lobby oppure aggiornare le lobby disponibili cliccando "a".

```
Elenco delle Lobby disponibili:
lobbyL [1/5]

Digita il nome di una lobby per entrarci, digita c per crearne una nuova
oppure digita a per aggiornare le lobby disponibili
lobbyL
Benvenuto nella lobby lobbyL!

Attendi l'arrivo degli altri partecipanti...
|
```

Figure 12: Lobbies available

Se invece, una volta effettuato l'accesso, non ci sono lobby disponibili viene chiesto all'utente di crearne una nuova oppure aggiornare le lobby disponibili.

```
Non ci sono lobby disponibili, digita c per creare una nuova lobby oppure a per
aggiornare le lobby disponibili.
c

Nome per la nuova lobby:
lobbyL
lobby: lobbyL creata con successo!

Attendi l'arrivo degli altri partecipanti...
Elenco delle Lobby disponibili:
lobbyL [0/5]

Digita il nome di una lobby per entrarci, digita c per crearne una nuova oppure
digita a per aggiornare le lobby disponibili
lobbyL
Benvenuto nella lobby lobbyL!
```

Figure 13: Lobby creation

```
Non ci sono lobby disponibili, digita c per creare una nuova lobby oppure a per
aggiornare le lobby disponibili.
a
Elenco delle Lobby disponibili:
lobbyL [1/5]

Digita il nome di una lobby per entrarci, digita c per crearne una nuova oppure
digita a per aggiornare le lobby disponibili
|
```

Figure 14: Lobby update

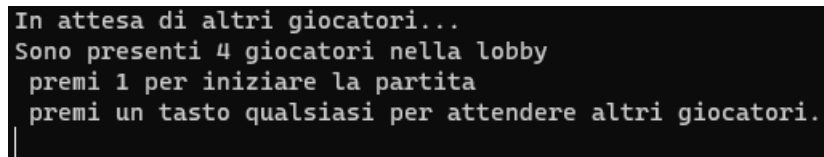
Una volta entrati nella lobby scelta si resta in attesa che altri utenti vi accedano. Quando sono presenti in una lobby 3 giocatori viene chiesto all'utente creatore della lobby se vuole iniziare la partita oppure se vuole attendere altri giocatori. Anche in questo caso se il giocatore non risponde entro 30 secondi viene eliminato dalla lobby.

```
Nome per la nuova lobby:
lobby2
lobby: lobby2 creata con successo!

Attendi l'arrivo degli altri partecipanti...
Sono presenti 3 giocatori nella lobby:
premi 1 per iniziare la partita
premi un tasto qualsiasi per attendere altri giocatori.
|
```

Figure 15: Three players message

La stessa cosa accade quando 4 giocatori hanno effettuato l'accesso ad una lobby. In questo caso se il creatore decide di aspettare altri giocatori la partita inizierà automaticamente quando il quinto giocatore accederà alla lobby.

A screenshot of a game lobby message displayed in a black box with white text. The text is in Italian and provides instructions for starting the game when four players are present.

```
In attesa di altri giocatori...  
Sono presenti 4 giocatori nella lobby  
premi 1 per iniziare la partita  
premi un tasto qualsiasi per attendere altri giocatori.  
|
```

Figure 16: Four players message

Giunti a questo punto la partita può iniziare. Verrà mostrato all'utente un messaggio di inizio partita dove verranno spiegate le regole di gioco, che variano in base al numero di giocatori, come già descritto.

```
INIZIO PARTITA!  
Queste sono le regole:  
-Ogni giocatore inizia la partita con 10 vite,  
ad ogni round ogni giocatore deve scegliere un numero da 1 a 100.  
Il sistema calcola la media dei numeri inseriti da tutti i giocatori,  
la moltiplica per 0.8, chi si avvicina di piu' al numero calcolato vince  
e gli altri perdono una vita.  
  
Regola aggiuntiva 4 giocatori:  
-Se due giocatori scelgono lo stesso numero gli viene detratta una vita.
```

Figure 17: Game start

Successivamente verrà richiesto all'utente di inserire il proprio numero e gli verrà mostrato un messaggio per segnalargli l'attesa degli altri giocatori.

```
ROUND 1  
Inserisci il tuo numero:  
33  
Hai scelto il numero: 33  
Attendi gli altri giocatori...
```

Figure 18: Choose number

Quando tutti i giocatori hanno inserito il proprio numero il sistema calcolerà il vincitore e rimuoverà una vita a tutti gli altri giocatori della lobby. In ogni Client verrà quindi visualizzato un messaggio che indica il vincitore, i numeri scelti dagli altri giocatori nel round corrente e le vite rimaste.

```
Hai scelto il numero: 4  
  
Il vincitore e' lollo  
  
I giocatori hanno scelto i seguenti numeri:  
55 4 37  
  
Non ti sei avvicinato abbastanza al numero corretto, perdi una vita!  
  
4 vite rimaste
```

Figure 19: Round loser

Nel caso in cui un giocatore abbia perso una vita a causa di una delle regole aggiuntive verrà stampato un ulteriore messaggio che spiega il motivo della vita persa.

```
Hai scelto 0 e il tuo avversario ha scelto 100, hai perso una vita  
I giocatori hanno scelto i seguenti numeri:  
100 0  
3 vite rimaste
```

Figure 20: Additional rule

Ogni giocatore ha a disposizione 30 secondi di tempo per inserire il proprio numero, scaduti questi avrà perso e verrà eliminato dalla lobby.

```
ROUND 1  
Inserisci il tuo numero:  
Tempo scaduto, sarai eliminato dalla lobby
```

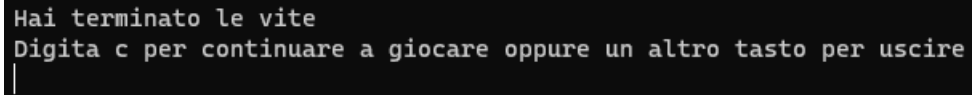
Figure 21: Time expired

In questo caso il round per gli altri giocatori verrà annullato e, dopo aver mostrato un messaggio che indica l'utente che è stato eliminato, si procederà con il round successivo.

```
gianlu12 ha abbandonato la partita!  
I giocatori sopravvissuti sono 2.  
  
Viene aggiunta una nuova regola oltre a quella iniziale:  
-se un giocatore sceglie 0 l'avversario vince se sceglie 100.
```

Figure 22: Time expired

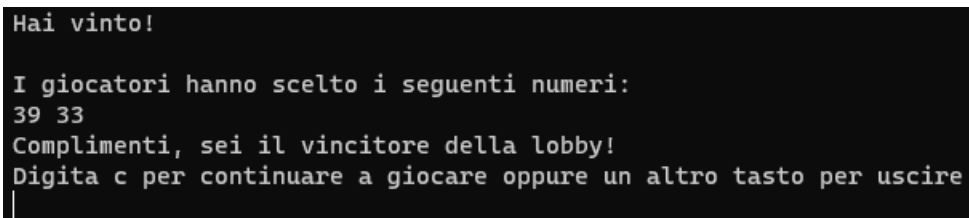
La partita di un giocatore termina quando ha finito le proprie vite, a questo punto potrà decidere se uscire o scegliere una nuova lobby con la quale iniziare una nuova partita.



```
Hai terminato le vite
Digita c per continuare a giocare oppure un altro tasto per uscire
|
```

Figure 23: Game over

Allo stesso modo l'ultimo giocatore che resterà in vita avrà vinto, gli sarà mostrato un messaggio e potrà decidere di uscire o iniziare una nuova partita in una nuova lobby.



```
Hai vinto!

I giocatori hanno scelto i seguenti numeri:
39 33
Complimenti, sei il vincitore della lobby!
Digita c per continuare a giocare oppure un altro tasto per uscire
|
```

Figure 24: Game winner

8 Conclusions

Con la progettazione e l'implementazione di Re di Quadri in versione online, è stato realizzato un sistema distribuito basato su un'architettura Client-Server che utilizza HTTP come protocollo di comunicazione e utilizzando le ReST API per quanto riguarda l'interazione con il servizio web.

Il sistema è stato sviluppato in modo da garantire consistenza e scalabilità, ed è in grado di gestire un numero variabile di utenti.

8.1 Future Works

Pensando ad una versione futura con più funzionalità si potrebbe implementare un meccanismo di autenticazione più complesso che può rendere possibile il mantenimento dei dati relativi all'utente all'interno di un database. In questo modo si potrebbe tener traccia delle partite passate di ogni utente e visualizzare alcune statistiche relative.

Si potrebbe inoltre realizzare un'interfaccia grafica per garantire una User Experience più semplice e coinvolgente.

8.2 What did we learned

Durante lo sviluppo del progetto sono riuscito a mettere in pratica gli argomenti trattati durante il corso, soprattutto per quando riguarda il funzionamento e l'implementazione di un Web Service per i sistemi distribuiti. Inoltre, ho potuto mettere in pratica anche gli aspetti principali del ciclo di vita di un software, partendo dalla parte di progettazione del sistema e sviluppo, e seguite da una fase di testing.

References

- [1] *Gson* - <https://github.com/google/gson>.
- [2] *Javalin* - <https://javalin.io>.
- [3] *JUnit 5* - <https://junit.org/junit5/>.
- [4] *Swagger* - <https://swagger.io>.